Online Active Learning with Dynamic Marginal Gain Thresholding

Mariel A. Werner, Anastasios Angelopoulos, Stephen Bates, Michael I. Jordan

January 26, 2022

Abstract

The blessing of ubiquitous data also comes with a curse: the communication, storage, and labeling of massive, mostly redundant datasets. In our work, we seek to solve the problem at its source, collecting only valuable data and throwing out the rest, via active learning. We propose an online algorithm which, given any stream of data, any assessment of its value, and any formulation of its selection cost, extracts the most valuable subset of the stream up to a constant factor while using minimal memory. Notably, our analysis also holds for the federated setting, in which multiple agents select online from individual data streams without coordination and with potentially very different appraisals of cost. One particularly important use case is selecting and labeling training sets from unlabeled collections of data that maximize the test-time performance of a given classifier. In prediction tasks on ImageNet and MNIST, we show that our selection method outperforms random selection by up to 5-20%.

1 Introduction

It is often infeasible to store and label large, real-world datasets in their entirety due to constraints on memory capacity, annotation resources, and communication patterns. Consequently, methods for extracting high-value subsets from large collections of data that incur low selection and memory cost are increasingly indispensable. Within a learning framework, one natural method is to assign high value to those subsets of the training data which improve a classifier's performance at test time. In practice, however, identifying such high-value training sets is challenging. Real-world datasets are often highly redundant (e.g., monotonous camera footage from self-driving vehicles or sequential video frames), such that labels are costly to obtain and of little marginal value.

Furthermore, in many settings (e.g., video streams, sensor data), the scale, velocity, and topology of data generation make it infeasible to store data at a central site for post-processing. Indeed, data sets may need to be processed in a distributed fashion across several machines or edge devices, necessitating a federated approach to set selection. Simply extracting a subset at random is often a bad solution; in particular, in data-imbalanced settings, rare classes will be under-represented in the training set. In many high-stakes settings these are classes for which the classifier's performance matters most. Consider medical imaging data or footage from autonomous vehicles, where it is unlikely that training data will contain many of the types of instances for which test-time accuracy is most critical (e.g., rare tumors or dangerous traffic events). Consider also face-recognition systems for which the available data may be severely imbalanced along demographic axes, yet which must perform fairly during deployment.

To address such challenges, we propose a general framework for selecting high-value subsets from a stream of data (or a distributed set of streams) at low selection and memory cost. Considering first the case with a single stream, let $\mathcal{D} = \{x_1, ..., x_n\} \in \mathcal{X}^n$ denote a sequence of n points, where $\mathcal{X}$ is a feature space, and let f be a function that takes as input a subset of $\mathcal{D}$ and outputs its value as a positive real number.[1] Points from $\mathcal{D}$ arrive sequentially and we must make an irrevocable decision at each time step to select the current point or not.

[1] Henceforth, for ease of expression, we will employ set syntax for our sequential framework. In particular, by 'subset' we will mean subsequence, by $A \subseteq B$, we will mean that A is a subvector of sequence B, and by 2^A, we will mean the set of all subvectors of sequence A.

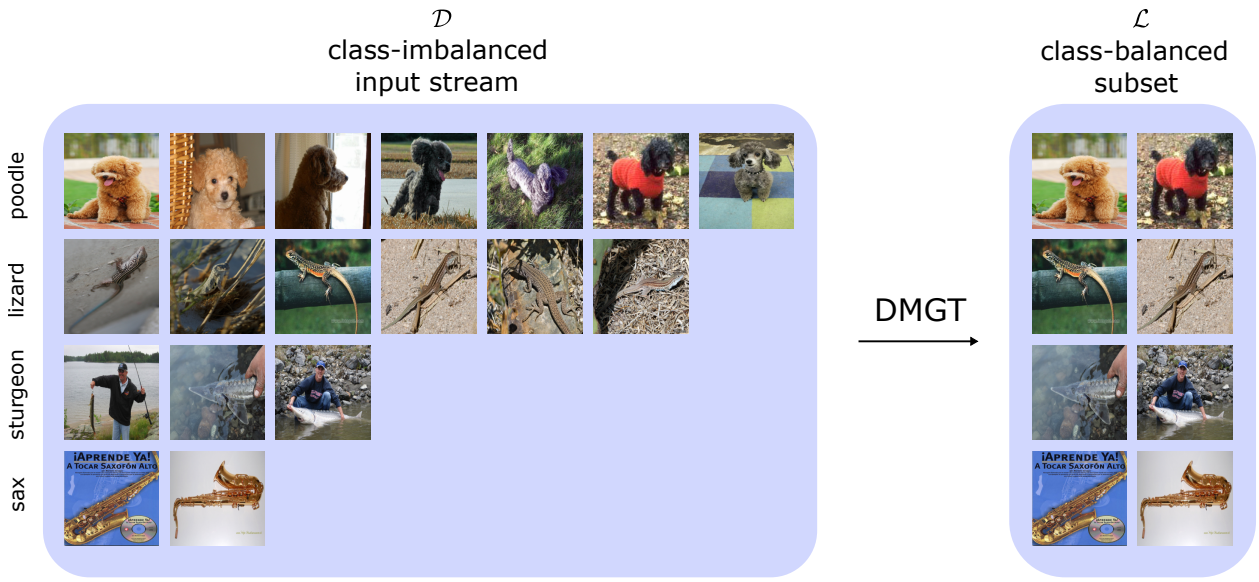


Figure 1: `DMGT` **correcting class imbalance on ImageNet**: In the imbalanced input stream $\mathcal{D}$ (left), poodles and lizards are common, while sturgeons and saxophones are rare. `DMGT` selects a class-balanced subset $\mathcal{L}$ (right).

Our goal is to select a subset $\mathcal{L} \subseteq \mathcal{D}$ whose value is close to

$$\max_{S \subseteq \mathcal{D}: \mathcal{C}} f(S), \tag{1}$$

where $\mathcal{C}$ denotes an arbitrary cost constraint. Suppose we define $\mathcal{C} : |S| = k$. If we had access to all data points in $\mathcal{D}$ simultaneously and if we had infinite computing resources, the optimal approach would be to search over all feasible subsets of $\mathcal{D}$ according to $\mathcal{C}$ and select the value-optimal subset:

$$\mathrm{OPT}(f, \mathcal{D}, k) \triangleq \operatorname*{arg\,max}_{S \subseteq \mathcal{D}: |S| = k} f(S).$$

Unfortunately, finding OPT requires solving a generally intractable combinatorial problem. We introduce a dynamic-thresholding algorithm that achieves almost the same performance without storing $\mathcal{D}$, simply by selecting a point if its marginal value exceeds a time-specific threshold set by the analyst—we call the algorithm *dynamic marginal gain thresholding* (`DMGT` — see Algorithm 1). Formally, at time step t, given the currently selected set $\mathcal{L}_{t-1}$, `DMGT` selects the current point x_t if

$$f(\mathcal{L}_{t-1} \cup \{x_t\}) - f(\mathcal{L}_{t-1}) > \tau_t,$$

where τ_t is a threshold chosen by the analyst at time t or any time prior. Under this selection rule, given any submodular, nonnegative, monotone increasing value function f and schedule of thresholds, $\mathcal{T} = \{\tau_t\}_{t \in |\mathcal{D}|}$, `DMGT` returns a set $\mathcal{L} \subseteq \mathcal{D}$ such that

$$f(\mathcal{L}) \geq \frac{\tau_{\min}}{\tau_{\min} + \tau_{\max}} f(\mathrm{OPT}(f, \mathcal{D}, |\mathcal{L}|)), \tag{2}$$

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum elements of $\mathcal{T}$ respectively. In words, `DMGT` selects a set whose value is *at least* a $\tau_{\min}/(\tau_{\min} + \tau_{\max})$-approximation to the value of the optimal, same-sized set. To exemplify this result, consider the case in which $\mathcal{T}$ is chosen to be a uniform sequence of thresholds such that $\tau_{\min} = \tau_{\max}$. Then the multiplicative factor in (2) becomes $1/2$. Consequently, in an uniform-threshold setting, our method performs at least $1/2$ as well as OPT. Finally, running `DMGT` requires only $O(|\mathcal{L}|)$ memory, the minimum amount needed for any online selection algorithm, as it is simply the memory needed to store the selected set $\mathcal{L}$. The generality of our method (namely that for any choice of $\mathcal{T}$, the threshold-dependent

bound in (2) holds) is particularly useful in distributed settings. Consider a scenario in which multiple agents (or nodes in the distributed system) individually aim to select a high-value, cost-sensitive subset from their own data streams. Furthermore, suppose the agents have different labeling resources: some agents have a large capacity for selecting and labeling data, and can afford to set low thresholds, while others are more cost-constrained and thus must set their thresholds higher. Even in this highly uncoordinated setting, our method returns a pooled subset from the agents whose value is at least a constant factor of the value of the optimal same-sized set; we develop an explicit counterpart to the bound in (2) for the distributed setting later in this work.

The simple thresholding technique of `DMGT` can apply to a large range of practical problems in which an analyst wants to extract high-value subsets from streams of data at low cost. In Section 2.3.1, we demonstrate its application to a class-imbalance correction problem, in which the goal is to extract and label a class-balanced training set for a classifier from a class-imbalanced, unlabeled data stream. Figure 1 previews the results on ImageNet. We set f to assign high value to class-balanced sets. Given an imbalanced input data stream $\mathcal{D}$ with an abundance of common classes and paucity of rare ones, our algorithm selects a high-value (i.e. class-balanced) training set $\mathcal{L}$ in a low-cost manner. That is, only the highest-value points which facilitate class balance are selected, saving selection costs on low-value points, with the entire procedure requiring minimal $O(|\mathcal{L}|)$ memory.

1.1 Related work

Solving (1) effectively and efficiently is a longstanding problem in submodular optimization since finding an exact solution is NP-hard for most choices of submodular f [1]. For monotone increasing, nonnegative, and submodular f, along with a cardinality constraint on the feasible sets, [2] proposed a greedy algorithm which obtains a $(1-1/e)$-approximation, and which has subsequently been augmented to several computationally efficient variants [3–6]. However, these algorithms are non-streaming and require that all of $\mathcal{D}$ be maintained in memory. In the streaming setting, [7] achieved a $(1/2-\epsilon)$-approximation for any $\epsilon > 0$ in $O(k \log k/\epsilon)$ memory (where k is an upper bound on cardinality of feasible subsets). Subsequently, [8] achieved the same approximation using only $O(k/\epsilon)$ memory. When $\mathcal{D}$ is a random rather than arbitrary (and thus potentially adversarial) stream, tighter constant factors are achievable. In particular, [9] demonstrated the possiblity of a greater-than-1/2-approximation in expectation, [10] precisely achieved a $(1-1/e-\epsilon)$-approximation in $O(k2^{\text{poly}(1/\epsilon)})$ memory, and most recently [11] achieved the same constant-factor approximation in improved $O(k/\epsilon)$ memory. Motivated by applications to team formation, [12] introduced a simple and efficient streaming algorithm which approximates an unconstrained, cost-diminished version of (1) for arbitrary $\mathcal{D}$ and submodular, nonnegative f in $O(k)$ memory. Though our optimization objective is fundamentally different from theirs (we optimize a submodular function while their objective is the difference between a submodular value function and an additive cost function), our selection rule and theoretical guarantees are inspired by their approach. Online greedy selection rules—whereby points are selected if their marginal gain under some function exceeds a given threshold—have been known and used in streaming submodular optimization for a long time. Indeed, all the aforementioned works use threshold schedules of various design to obtain $\mathcal{L}$. However, our algorithm is more flexible, gives constant-factor approximations for a wider class of thresholds.

Another line of work we build on is distributed submodular maximization. Parallelization of submodular optimization algorithms across different machines in the MapReduce framework is the focus of [4, 6, 13–17]. Closer to our setting of interest is [18] which explores the multi-agent regime. In particular, they show that for non-streaming, k-cardinality-constrained optimization of a submodular, nonnegative objective and M agents, its possible to achieve a $O(1/\min(M,k))$ constant approximation to the value of the optimal subset from the agents' pooled original sets. All the aforementioned distributed selection approaches are non-streaming. To the best of our knowledge, we present the first proposal for a *streaming* distributed submodular maximization algorithm, accompanied by theoretical guarantees. Furthermore, we attain the same scaling in M even though we are restricted to the streaming setting.

We also draw inspiration from the extensive literature on submodular optimization [19] for real-world applications, such as representation, summarization, and network modeling [20–35]. In particular, [36, 37] introduced a general framework for pool-based active learning with submodular information measures [38, 39]

(combinations of submodular functions which generalize information-theoretic quantities such as mutual and conditional information and conditional entropy beyond random variables to arbitrary sets). Importantly, they focused on correcting real-world cases of class imbalance, presence of out-of-distribution samples, and redundancy in unlabeled datasets with active labeling. However, their framework is non-streaming and requires that the entire unlabeled dataset be stored in random-access memory, which is infeasible for many real-world applications.

1.2 Our contribution

We present a general online, threshold-based selection procedure that (1) given any arbitrary (and possibly adversarial) data stream $\mathcal{D}$, (2) any submodular, nonnegative, monotone increasing value function f, and (3) any set of thresholds $\mathcal{T}$ chosen by the analyst, returns a subset $\mathcal{L} \subseteq \mathcal{D}$ whose value is at least a fraction $\tau_{\min}/(\tau_{\min} + \tau_{\max})$ of the value of the optimal same-sized set. Our algorithm has no memory requirements beyond what is needed to store $\mathcal{L}$, and it gives a lower bound on the value of $\mathcal{L}$ in terms of $\mathcal{T}$, for any $\mathcal{T}$ that the analyst chooses to implement. We highlight that our theoretical analysis gives constant-factor guarantees for a broad set of selection algorithms, substantially enlarging the space of labeling rules that the analyst can deploy with confidence. We obtain similar constant-factor optimality guarantees for federated selection, in which multiple uncoordinated agents select separately from different data streams with different threshold sets. To our knowledge, this is the first proposal for a distributed submodular maximization algorithm in the streaming setting. Finally, we conduct experiments for the problem of class-imbalance correction in both the standard and federated settings. When training a convolutional neural network on sets selected by our algorithm versus randomly selected same-size subsets, we demonstrate up to 20% prediction accuracy gains on MNIST and up to 5% gains on ImageNet. The code for all of our experiments is available on GitHub.

2 Theory and Methods

In this section, we present our proposed methodology and supporting theory. Section 2.1 establishes setting, notation, and fundamental properties of our value function and selection rule, Section 2.2 introduces our online selection method, `DMGT`, Section 2.3 demonstrates how we can use `DMGT` to correct for class imbalance in data streams, and finally, Section 2.4 introduces a federated version of `DMGT` with associated theoretical guarantees.

2.1 Submodularity

Let $\mathcal{D} = \{x_1, ..., x_n\} \in \mathcal{X}^n$ denote a sequence of data with $\mathcal{X}$ a feature space. Our objective is to extract in an online fashion a high-value subset, $\mathcal{L}$, of $\mathcal{D}$ at low cost, where we let $f : 2^{\mathcal{D}} \to \mathbb{R}^+$ quantify the value of subsets of $\mathcal{D}$.

The *value* of a set of points depends on the application or the objectives of the analyst. For instance, if the objective is to extract a class-balanced subset, then f would assign high value to class-balanced subsets of the data stream $\mathcal{D}$. If the objective is to maximize diversity in selected sets, such that the data are separated in the feature space, subsets of $\mathcal{D}$ with this property have high value. Since our selection decisions boil down to determining whether, given the currently selected set, the value gained from adding the current point exceeds a particular threshold, the first quantity of relevance to our setting is *marginal gain*, essentially a measure of how much one set adds to another under a specified function.

Definition 1 (Marginal gain)**.** *The* marginal gain *under function g of set S given set T is*

$$g(S|T) \triangleq g(S \cup T) - g(S).$$

Regardless of how large the already selected set is, if a new point adds sufficient value, it will be included. However, for most reasonable value functions, adding a point to a set should always count more than adding

it to a *superset* of that set. This diminishing returns property, wherein the amount of gain from adding new points to a set changes inversely with the size of that set, can be formalized by taking the value function to be *submodular* [19, 38].

Definition 2 (Submodularity)**.** *A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is* submodular *if for all sets $S \subseteq T \subseteq \mathcal{X}$ and $x \in \mathcal{X} \backslash T$, we have*

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T).$$

Importantly, appraising value with submodular functions helps to regulate selection costs. Namely, the diminishing returns property controls the size of the selected subset and penalizes the addition of points that don't contribute sufficient value.

2.2 DMGT: Dynamic Marginal Gain Thresholding

We now present our main algorithm along with its optimality guarantee. The algorithm is simple: at each time step, we compare the marginal value of the incoming point, given the already selected set, to a threshold value which can either be set at the current time step or any time in advance of it. We then select the point if its marginal value exceeds the corresponding threshold. Our framework and analysis can handle any threshold schedule, even those set adaptively in an online fashion.

We denote the threshold schedule as $\mathcal{T} = \{\tau_t\}_{t \in |\mathcal{D}|} \in \mathbb{R}^{|\mathcal{D}|}$, where τ_t may depend on any information observed in the algorithm by the analyst prior to time step t. Formally, $\tau_t = \mathcal{A}(x_1, ..., x_t, \tau_1, ..., \tau_{t-1})$, where $\mathcal{A}$ is a threshold-setting routine that, at the current time step, is allowed to depend on the current point, all prior points in the stream, and all prior thresholds, and outputs a threshold value for the current time step. When DMGT is used to decide points to label (e.g. Section 2.3), then $\mathcal{A}$ may also depend on prior queried labels. After discussing Theorem 1, we present one canonical way to design $\mathcal{A}$.

Algorithm 1 DMGT: Dynamic Marginal Gain Thresholding

Input Stream of arbitrary data $\mathcal{D} \subset \mathcal{X}$; value function $f : 2^{\mathcal{D}} \to \mathbb{R}^+$

Set of currently selected points $\mathcal{L}_0 = \emptyset$
Set of thresholds $\mathcal{T} = \emptyset$
for $t = 1, ..., |\mathcal{D}|$ **do**
    Receive point x_t and set threshold $\tau_t = \mathcal{A}(x_1, ..., x_t, \tau_1, ..., \tau_{t-1})$.
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau_t\}$
    if $f(\mathcal{L}_{t-1} \cup \{x_t\}) - f(\mathcal{L}_{t-1}) > \tau_t$ **then**
        $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1} \cup \{x_t\}$
    else
        $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1}$
return $\mathcal{L} = \mathcal{L}_{|\mathcal{D}|}, \mathcal{T}$

Theorem 1 (Near optimality of DMGT)**.** *Suppose* DMGT *takes as input an arbitrary data stream $\mathcal{D}$, a submodular, nonnegative, monotone increasing value function f, and constructs threshold schedule $\mathcal{T}$. Then*

$$f(\mathcal{L}) \geq \underbrace{\frac{\tau_{\min}}{\tau_{\min} + \tau_{\max}} f(\mathrm{OPT}(f, \mathcal{D}, |\mathcal{L}|))}_{\text{constant-factor optimal}} + \underbrace{\frac{\tau_{\min}\tau_{\max}}{\tau_{\min} + \tau_{\max}} |\mathrm{OPT}(f, \mathcal{D}, |\mathcal{L}|) \cap \mathcal{L}|}_{\geq 0}, \tag{3}$$

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum elements of $\mathcal{T}$ respectively, and $\mathcal{L}$ is the set selected by DMGT.

The first term in the bound of (3) is the important one for understanding the performance of DMGT. It bounds the value of $\mathcal{L}$ in terms of the value of the same-sized optimal set. If, in advance of running DMGT, the analyst constrains thresholds to fall in a bounded interval $[\tau_{\min}, \tau_{\max}]$, then DMGT ensures an a priori constant-factor approximation to the value of the same-sized optimal set. One special case is when

the values in $\mathcal{T}$ are set to be uniform, i.e., $\tau_{\max} = \tau_{\min} = \tau^*$. Then the constant factor $\tau_{\min}/(\tau_{\min} + \tau_{\max})$ reduces to $1/2$. Therefore, if aiming for a $1/2$ approximation, the analyst may simply set a uniform threshold across the selection horizon. If they want a large volume of points to be selected, they should set τ^* to a low value; otherwise, they should set it higher. Suppose the analyst sets $\mathcal{T}$ adaptively (rather than in advance), based on information acquired from the data stream. Then `DMGT` still gives an instance-specific bound based on the threshold values chosen during its implementation. Finally, the second term in the bound of (3), a relic of the proof, is always positive and only improves the optimality of $f(\mathcal{L})$.

We now briefly discuss an intuitive way to design $\mathcal{T}$ and an extension of `DMGT` to the batch setting.

Setting thresholds based on cost. `DMGT` is a general framework for threshold-based online selection that guarantees the bound in (3), *regardless* of how $\mathcal{T}$ is chosen. One intuitive way to set $\mathcal{T}$, given the motivating themes of *value* and *cost* in this paper, is as a sequence of marginal costs. To understand this, let us first define *cost* in a more formal sense as a function $c : 2^{\mathcal{D}} \to \mathbb{R}^+$ which takes in a subset of $\mathcal{D}$ and assigns a cost to that subset in the form of a real nonnegative number. For the practitioner who wants to train a classifier on the selected data, one intuitive choice for c would be cardinality, or a positive-constant scaling of cardinality. In this setting, larger sets should be more costly to select because they contain more points which require expensive labeling. Given this formal notion of cost, we can define a sequence of thresholds. One reasonable approach might be as follows: at time step t, given the currently selected set $\mathcal{L}_{t-1}$, only select the current point x_t if the marginal value of doing so outweighs the marginal cost; that is, if

$$f(\mathcal{L}_{t-1} \cup \{x_t\}) - f(\mathcal{L}_{t-1}) > c(\mathcal{L}_{t-1} \cup \{x_t\}) - c(\mathcal{L}_{t-1}).$$

Deploying this rule, the analyst obtains a set whose value obeys (3), while ensuring that at every time step, they never pay more for selecting a point than the value it contributes. In this case, the threshold-setting routine $\mathcal{A}$ depends on the prior-observed data and an external cost function decided by the analyst. While marginal costs are one intuitive way to construct $\mathcal{T}$, we again emphasize that the guarantee of `DMGT` in (3) holds for *any* choice of $\mathcal{T}$.

`DMGT` in batch mode In practice, we might want to implement `DMGT` in batch mode in order to update f with information from previously selected points (see the example in Section 2.3.1). In this case, f changes across batches, and the guarantee in Theorem 1 only holds per batch; i.e., for the value function and selected set in each batch. To cover this setting, we present an extension of `DMGT` in Appendix B, along with a guarantee on the value of the cumulative selected set (i.e., a union of selected sets over batches). In short, a constant-factor threshold-dependent approximation that scales as $O(1/B)$ is still achievable in the batch setting, where B is the number of batches.

2.3 `DMGT` for learning

Up to this point, we have been concerned with the abstract task of selecting a subset from some feature space $\mathcal{X}$ that has maximal value. We now show how `DMGT` can be used for learning. Given a feature space $\mathcal{X}$, a set of classes $\mathcal{Y} = \{1, ..., K\}$, a classifier $\hat{\pi}$, and $\mathcal{D} = \{(X_i, Y_i)\}_{i \in [n]} \in \mathcal{X}^n \times \mathcal{Y}^n$ a set of feature-class *pairs*, we'd ideally like to select the subset of $\mathcal{D}$ on which to train $\hat{\pi}$ that maximizes the predictive performance of $\hat{\pi}$ on test-time data. Intuitively, the more closely the selected training set $\mathcal{L} \subseteq \mathcal{D}$ resembles the test-time data, the better we'd expect the test-time accuracy of $\hat{\pi}$ to be. Therefore, within our value-function framework, we assign more value to sets which resemble the expected test-time data. We now present a detailed example to show how `DMGT` can be deployed for learning in this type of setting. This example also provides the theoretical basis of our experiments in Section 3.

2.3.1 Example: Correcting class imbalance

We consider an example where the training data, $\mathcal{D}$, is significantly class-imbalanced, and we'd nonetheless like $\hat{\pi}$ to predict well on *all* classes at test time. We assume that feature-class pairs from $\mathcal{D}$ arrive sequentially.

Furthermore, we assume that X_i in each pair can be observed upon arrival, but its label Y_i can only be known if actively queried (e.g., sought out by a human). Selecting a sequence for annotation at random from $\mathcal{D}$ preserves class imbalance, leaving $\hat{\pi}$ under-exposed to rare classes in training and thus a poor classifier of them at test time. We can deploy `DMGT` in this setting to select a high-value (i.e. class-balanced) subsequence $\mathcal{L}$ from $\mathcal{D}$, whose Y coordinates we then query to form a class-balanced training set for $\hat{\pi}$.

To do this, we split $\mathcal{D}$ into B batches, run `DMGT` sequentially on each batch, and between batches update $\hat{\pi}$ on the newly selected points. This induces a sequence of value functions $\{f_{\hat{\pi}^b}\}_{b\in[B]}$, where $\hat{\pi}^b$ denotes $\hat{\pi}$ at the start of batch b. In batch b at the current time step, given the currently labeled set, $\mathcal{L}_b$, we define the value of adding a new point and its class label (x, y) to $\mathcal{L}_b$ to be

$$f_{\hat{\pi}^b}(\mathcal{L}_b \cup \{(x,y)\}) \triangleq \sum_{k\in[K]} \hat{\pi}^b_k(x) g\left(\sum_{z\in\mathcal{L}_b\cup\{(x,y)\}} \hat{\pi}^b_k(z_x)\right). \tag{4}$$

We allow g to be any concave increasing function, which ensures submodularity of $f_{\hat{\pi}^b}$ (see Appendix A), and use the notation z_x to denote the first coordinate of the tuple $z = (x, y)$. With this choice of value function, we see that the label, y, of x is queried by `DMGT` if

$$\sum_{k\in[K]} \hat{\pi}^b_k(x) \left[g\left(\sum_{z\in\mathcal{L}_b\cup\{(x,y)\}} \hat{\pi}^b_k(z_x)\right) - g\left(\sum_{z\in\mathcal{L}_b} \hat{\pi}^b_k(z_x)\right)\right] > \tau^b_{(x,y)}. \tag{5}$$

Here we take $\tau^b_{(x,y)}$ to be the threshold corresponding to the arrival of point x in batch b. For ease of notation, we drop reference to b, though it is to be assumed. Since g is concave, the left-hand side of (5) will be small (assuming sufficient model accuracy) when the currently labeled set already contains many instances with the same label as the current point x, and large when it doesn't, thus favoring class balance. In practice, we observe that assuming access to labels of previously selected points in our value function improves performance. (Note that having access to labels of points we've previously elected to label is a very light assumption.) Therefore, in our experiments in Section 3, given currently labeled set $\mathcal{L}$, we label the current point x if

$$\sum_{k\in[K]} \hat{\pi}_k(x)\big[g(1 + |\{(x,y)\in\mathcal{L} : y = k\}|) - g(|\{(x,y)\in\mathcal{L} : y = k\}|)\big] > \tau_{(x,y)}. \tag{6}$$

The value function in this formulation, $f_{\hat{\pi}}(\mathcal{L}\cup\{(x,y)\}) = \sum_{k\in[K]} \hat{\pi}_k(x) g(1 + |\{(x,y)\in\mathcal{L} : y = k\}|)$, is also submodular (see Appendix A).

2.3.2 Setting thresholds for class balance

Depending on g, we can leverage the form of $f_{\hat{\pi}}$ to construct a threshold set $\mathcal{T}$ that controls how many instances of each class are selected for labeling. Note that with perfect model accuracy (i.e., for every input, the softmax vector encoding $\hat{\pi}$ has a 1 in the correct class entry and 0 everywhere else), the marginal gain quantity on the left-hand side of (6) simplifies to

$$f_{\hat{\pi}}(\mathcal{L}\cup\{(x,y)\}) - f_{\hat{\pi}}(\mathcal{L}) = g(1 + |\{z\in\mathcal{L} : z_y = y\}|) - g(|\{z\in\mathcal{L} : z_y = y\}|),$$

where z_y is the second coordinate of the tuple $z = (x, y)$. We consider the particular case where $g(x) = \sqrt{x}$, a natural scaling. Suppose we'd like to have n samples from each class in our training set and there are K classes. We can approximately achieve this desired class balance in the training set, regardless of any imbalance in the input stream, by setting a uniform threshold τ^* across the selection horizon such that τ satisfies

$$\sqrt{n+1} - \sqrt{n} = \tau. \tag{7}$$

Figure 2 displays how closely `DMGT` achieves this ideal balance for different threshold values on ImageNet after running for six rounds of point selection and classifier updates. We see that for most threshold values in the given range the number of selected rare instances and selected common instances are nearly equal.

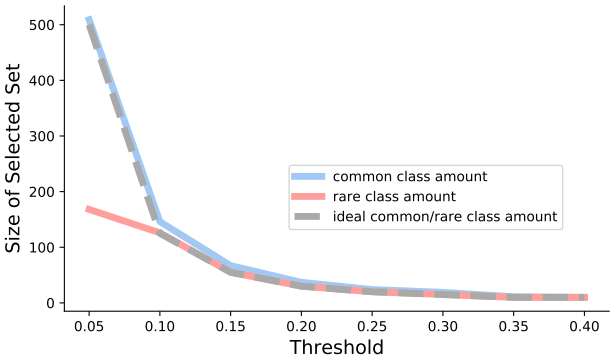


Figure 2: **Rare/Common class balance on ImageNet as a function of threshold values**: For thresholds 0.1 and greater, selected rare classes achieve the ideal amount, while selected common classes converge to the correct level slightly more slowly.

2.4 FED-DMGT: DMGT for federated selection

We now show the utility of DMGT for federated selection. Consider a scenario in which there are M agents or devices (e.g. phones, laptops) each receiving their own streams of data. If each agent separately deploys DMGT on their stream, we show that the value of pooled selected subsets from the agents has a $O(1/M)$-approximation to the value of the optimal same-sized set from the pooled original streams. To obtain such a subset, we introduce FED-DMGT, which simply runs DMGT separately for each agent and then combines the agents' selected subsets.

Algorithm 2 FED-DMGT: Federated Dynamic Marginal Gain Thresholding

Input M streams of data $\{\mathcal{D}_j\}_{j\in[M]}$, with $\mathcal{D}_j \subset \mathcal{X}$; value function $f : 2^{\mathcal{D}} \to \mathbb{R}^+$ (where $\mathcal{D} = \cup_{j\in[M]}\mathcal{D}_j$)

1: **for** $j = 1, ..., M$ **do**
2: $\quad \mathcal{L}_j, \mathcal{T}_j \leftarrow \texttt{DMGT}(\mathcal{D}_j, f)$
3: **return** $\mathcal{L} = \cup_{j\in[M]}\mathcal{L}_j, \mathcal{T} = \cup_{j\in[M]}\mathcal{T}_j$.

Theorem 2 (Near optimality of FED-DMGT)**.** *Suppose* FED-DMGT *runs on M separate arbitrary data streams $\{\mathcal{D}_j\}_{j\in[M]}$, each with submodular, nonnegative, monotone increasing, value function f, and constructs threshold sets $\{\mathcal{T}_j\}_{j\in[M]}$. Define*

$$\mathcal{D} \triangleq \cup_{j\in[M]}\mathcal{D}_j.$$

Then

$$f(\mathcal{L}) \geq \frac{\tau_{\min}}{M(\tau_{\min}+\tau_{\max})} f(\mathrm{OPT}(f, \mathcal{D}, |\mathcal{L}|)) + \frac{\tau_{\min}\tau_{\max}}{M(\tau_{\min}+\tau_{\max})} |\mathrm{OPT}(f, \mathcal{D}, |\mathcal{L}|) \cap \mathcal{L}|, \tag{8}$$

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum elements respectively of $\mathcal{T}$.

The bound in (8) holds regardless of how different agents choose their individual threshold sets. Indeed, agents may have very different cost functions or resources for labeling which inform how they set their thresholds. Given any such divergent choices, FED-DMGT still guarantees a constant-factor approximation which depends only on the number of agents and the minimum and maximum values observed in the agents' combined threshold sets.

If $M = 1$, FED-DMGT reduces to DMGT and consequently (8) reduces to (3). As the number of agents increases, the potential for subsets to be selected by agents that individually have high value but collectively have low value arises (e.g., consider the extreme case in which all agents individually select very similar sets), thus potentially decreasing the value of the pooled labeled set relative to the optimal set and weakening the bound in (8). The scaling in M here is the same as that observed in offline distributed submodular maximization.

3 Experiments

In this section, we report results from a variety of image classification experiments for the class-imbalance correction example from Section 2.3.1. We take $\mathcal{X}$ to be images from MNIST and ImageNet, and we let $\mathcal{Y}$ be the corresponding labels for those images. We define our value function $f_{\hat{\pi}}$ as in equation (6), setting $g(x) = \sqrt{x}$, and we take $\mathcal{T}$ to be a uniform sequence of thresholds for all experiments. For MNIST, we initialize $\hat{\pi}$ to be an untrained ResNet50 classifier from the `torchvision` [40] repository, and for ImageNet, we use the same base model, but pre-train it on SimCLR feature embeddings [41] for efficiency. For training $\hat{\pi}$, we use an SGD optimizer with learning rate 10^{-3}, momentum 0.9, and weight-decay 5^{-4}. We follow the same general recipe for all experiments:

1. Partition the set of classes $[K]$ into $\mathcal{R}$, rare classes, and $\mathcal{C}$, common classes. For all experiments on MNIST, we set $\mathcal{R}$ to be classes $\{0, 1, 2, 3, 4\}$ and $\mathcal{C}$ to be classes $\{5, 6, 7, 8, 9\}$. For ImageNet, we populate $\mathcal{R}$ and $\mathcal{C}$ with 5 randomly selected classes each.

2. Set an imbalance parameter β and from $\mathcal{X}^n \times \mathcal{Y}^n$, sample $\mathcal{D}$, an $\beta\times$-imbalanced sequence of image-label pairs. By a $\beta\times$-imbalanced sequence, we mean that $\mathcal{D}$ should contain β times as many points whose $\mathcal{Y}$ coordinate belongs to $\mathcal{C}$ as to $\mathcal{R}$.

3. Warm-start train $\hat{\pi}$ on a randomly chosen, 1000-length subsequence, $\mathcal{D}_0$, of $\mathcal{D}$, and set the currently labeled set $\mathcal{L} = \mathcal{D}_0$.

4. In 1000-point batches, $\{\mathcal{D}_b\}_{1:B}$, pass the remainder of $\mathcal{D}$ to `DMGT`, which in each batch b selects a subsequence $\mathcal{L}_b$ of $\mathcal{D}_b$ whose labels are to be queried. Before the next batch commences, update $\hat{\pi}$ on $\mathcal{L}_b$ over 200 epochs or until training accuracy achieves 0.99, and add $\mathcal{L}_b$ to the set of selected points $\mathcal{L}$. Between batches, calibrate $\hat{\pi}$ with isotonic regression on held-out validation data. For this purpose, on ImageNet we use half of the validation set, and on MNIST half of the test set. The other halves respectively are held back for prediction experiments.

5. Finally we demonstrate performance of `DMGT` on a variety of metrics like prediction accuracy and class-balance convergence, reporting results both for all classes and just rare classes.

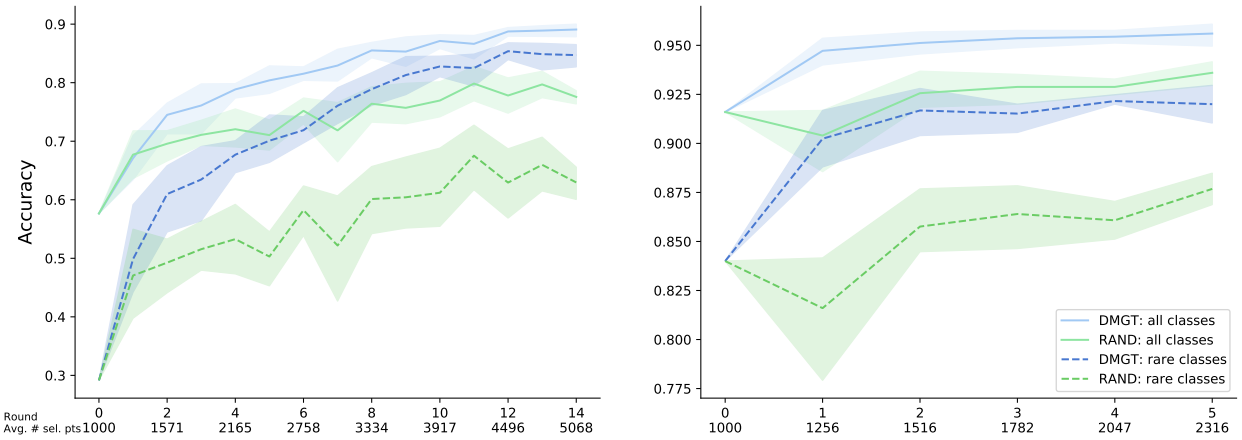
One hopes that after a few applications of `DMGT`, we will obtain a classifier which predicts well on all classes at test time and a value function which selects class-balanced subsets from a continually class-imbalanced stream. We demonstrate experimentally that this is indeed the case—and moreover that the number of iterations needed to achieve these results can be as small as 1.

3.1 Experiment 1: Accuracy of `DMGT` vs. `RAND` on MNIST and ImageNet

In Figure 3, we benchmark `DMGT`'s performance against an algorithm `RAND`, which randomly selects a training subset of $\mathcal{D}$ to label of the same cardinality as `DMGT`'s selected set. Setting the imbalance factor $\beta = 5$ and taking a uniform threshold $\tau = 0.1$, we track the prediction accuracy of `DMGT` and `RAND` over multiple rounds (1000 points streamed per round) on held-out test data from MNIST and ImageNet. The first row of the x-axis on our plots indicates the round, and the second row indicates the cardinality of $\mathcal{L}$ after the corresponding round. Precisely, in Figure 3a, the procedure runs for 14 selection rounds plus the initial warm-start training round, so $\mathcal{D} = 15K$. We see that that by the end of the entire procedure, only about a third of those points have been selected for labeling, while the prediction accuracy of `DMGT` on rare classes exceeds that of `RAND` by around 20% on MNIST and 5% on ImageNet.

3.2 Experiment 2: Convergence to class balance on MNIST and ImageNet

In Figure 4, we compare how quickly `DMGT` and `RAND` start selecting class-balanced subsets from continuously class-imbalanced streams of MNIST and ImageNet data. We run this experiment for the same imbalance factor ($\beta = 5$) and uniform threshold value ($\tau = 0.1$) as before. Given these parameters, the fact that

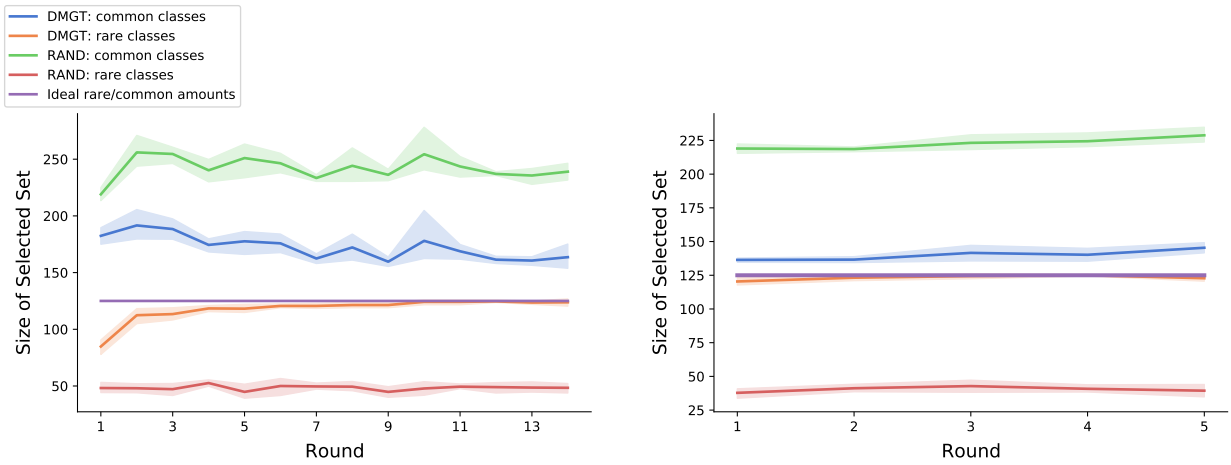


(a) MNIST: prediction accuracy on all and rare classes

(b) ImageNet: prediction accuracy on all and rare classes

Figure 3: **Prediction accuracy on MNIST and ImageNet with** `DMGT`: We show prediction accuracy of `DMGT` vs. `RAND` after running both algorithms for 14 selection rounds on MNIST and 5 selection rounds on ImageNet. Plots depict 95% confidence intervals around the mean accuracy value over 5 random permutations of $\mathcal{D}$.

$|\mathcal{R}| = |\mathcal{C}| = 5$ and $g(x) = \sqrt{x}$ in our setting, and that our objective is class-balanced selection, we'd like ~ 125 rare-class instances and ~ 125 common-class instances to be selected during a pass over the stream (see (7)). After approximately six selection and model-updating rounds on MNIST and just one on ImageNet by `DMGT`, the selected rare-class amount achieves and sustains this desired level. Common classes converge to 125 instances more slowly than rare classes for both datasets, but are significantly less abundant under selection by `DMGT` than by `RAND`.



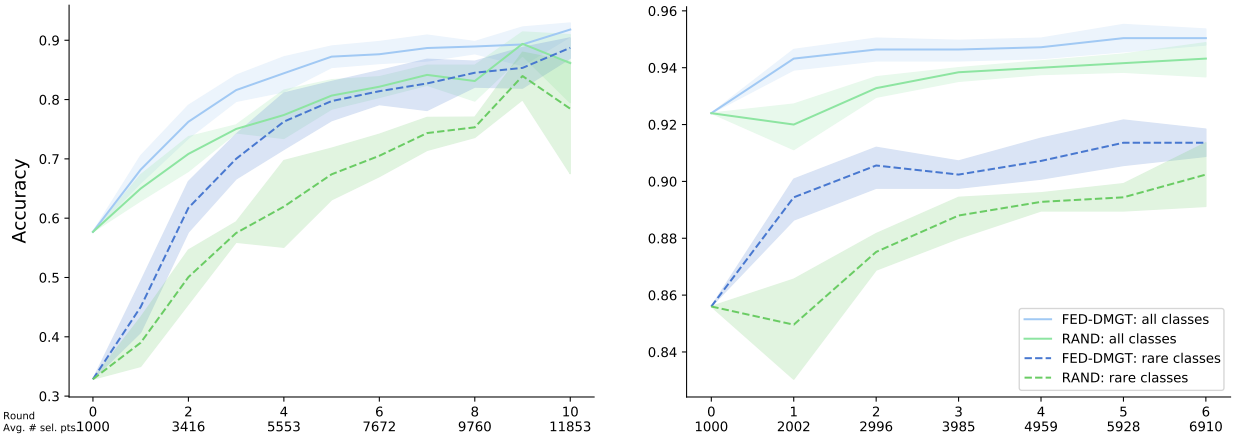
(a) MNIST class balance over rounds

(b) ImageNet class balance over rounds

Figure 4: **Class Balance**: Rare and common class amounts converge to the ideal amount of 125, given $g(x) = \sqrt{x}$, $\tau = 0.1$, and $|\mathcal{R}| = |\mathcal{C}| = 5$. Plots depict 95% confidence intervals around the mean amounts over 5 random permutations of $\mathcal{D}$.

3.3 Experiment 3: Accuracy of FED-DMGT vs. RAND on MNIST and ImageNet

In Figure 5, we run FED-DMGT with three agents, whose streams each have different imbalance factors ($\beta = 2, 5,$ and 10) and corresponding uniform threshold values ($\tau = 0.15, 0.1,$ and 0.05). In each selection round, each agent runs DMGT on their individual stream and then sends their selected set to a global model, $\hat{\pi}$, which updates on the pooled selected sets from the agents. The updated model is then broadcast back to the agents to be used for their selection decisions in the next round of DMGT. At each round, Figure 5 compares the prediction accuracy of $\hat{\pi}$ trained on the pooled DMGT-selected sets from the agents and a randomly selected, same-sized set from the agents' combined original streams. We see that despite the differences in stream balance and threshold sets amongst the agents, DMGT still outperforms RAND. With a training set whose size is only $\sim 30\%$ of the original pooled data, DMGT achieves 10% accuracy gains when predicting on rare MNIST classes and 1% gains on rare ImageNet classes.



(a) MNIST: accuracy on all and rare classes

(b) ImageNet: accuracy on rare and all classes

Figure 5: **Prediction accuracy on MNIST and ImageNet with FED-DMGT**: We show prediction accuracy of FED-DMGT vs. RAND after running both algorithms for 10 selection rounds on MNIST and 6 selection rounds on ImageNet. Plots depict results for 3 agents selecting from differently imbalanced streams with different threshold sets. We show 95% confidence intervals around the mean accuracy value over 5 random permutations of $\mathcal{D}$.

4 Discussion

We propose an algorithm, DMGT, which selects high-value subsets from streams of data at low selection and memory cost. DMGT decides to select each point in the stream if its marginal value given the currently selected set exceeds a threshold decided by the analyst at that time step. One natural choice for the thresholds, $\mathcal{T}$, across the selection horizon are the marginal costs of the points in the stream under an appropriate cost function. However, DMGT guarantees that, *regardless* of how $\mathcal{T}$ is set, the value of the selected set can always be lower bounded by a $\mathcal{T}$-dependent scaling of the value of the optimal same-sized set.

Our selection method is simple and efficient, passing only once over the data stream $\mathcal{D}$ and requiring only enough memory to store the selected set $\mathcal{L}$. A principal contribution of our algorithm is to the federated setting in which multiple agents individually aim to select a high-value, cost-sensitive subset from their own data streams, each according to different, uncoordinated threshold sets. For this setting our algorithm also guarantees a constant-factor approximation to the optimal same-sized set. Finally, we experimentally demonstrate the practical utility of our method for online active learning in a class-imbalanced regime: a classifier trained on sets selected by our method in this context significantly outperforms one trained on randomly selected training sets.

Acknowledgements

We wish to thank Kirthevasan Kandasamy and Sayna Ebrahimi for helpful discussions early on. M. W. was supported by the Chancellor's Fellowship. A. A. was supported by the Berkeley Fellowship and the NSF GRFP. S. B. was supported by the Foundations of Data Science Institute Postdoc Fellowship.

References

[1] U. Feige, "A threshold of ln n for approximating set cover," in *Journal of the ACM*, vol. 45, 1998, pp. 634–652. [Online]. Available: https://doi.org/10.1145/285055.285059.

[2] G. Nemhauser and L. Wolsey, "An analysis of approximations for maximizing submodular set functions," in *Mathematical Programming 14*, 1978, pp. 265–294.

[3] A. Badanidiyuru and J. Vondrak, "Fast algorithms for maximizing submodular functions," in *Proceedings of the 25th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014, pp. 1497–1514.

[4] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani, "Fast greedy algorithms in MapReduce and streaming," in *ACM Transactions on Parallel Computing*, vol. 2, 2015, pp. 1–22. [Online]. Available: https://doi.org/10.1145/2809814.

[5] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization Techniques*, 1978, pp. 234–243.

[6] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause, "Lazier than lazy greedy," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 2015, pp. 1812–1818.

[7] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming submodular maximization: Massive data summarization on the fly," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2014, pp. 671–680. DOI: 10.1145/2623330.2623637.

[8] E. Kazemi, M. Mitrovic, M. Zadimoghaddam, S. Lattanzi, and A. Karbasi, "Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, 2019, pp. 3311–3320. [Online]. Available: https://proceedings.mlr.press/v97/kazemi19a.html.

[9] A. Norouzi-Fard, J. Tarnawski, S. Mitrović, A. Zandieh, A. Mousavifar, and O. Svensson, "Beyond 1/2-approximation for submodular maximization on massive data streams," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 3829–3838. [Online]. Available: https://proceedings.mlr.press/v80/norouzi-fard18a.html.

[10] S. Agrawal, M. Shadravan, and C. Stein, "Submodular secretary problem with shortlists," 2018, arXiv:1809.05082.

[11] P. Liu, A. Rubinstein, J. Vondrak, and J. Zhao, "Cardinality constrained submodular maximization for random streams," 2021, arXiv:2111.07217.

[12] S. M. Nikolakaki, A. Ene, and E. Terzi, "An efficient framework for balancing submodularity and cost," in *The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2021, pp. 1256–1266. DOI: 10.1145/3447548.3467367. [Online]. Available: https://doi.org/10.1145/3447548.3467367.

[13] V. Mirrokni and M. Zadimoghaddam, "Randomized composable core-sets for distributed submodular maximization," in *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, 2015, pp. 153–162. [Online]. Available: https://doi.org/10.1145/2746539.2746624.

[14] R. d. P. Barbosa, A. Ene, H. L. Nguyen, and J. Ward, "The power of randomization: Distributed submodular maximization on massive datasets," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 1236–1244. [Online]. Available: https://proceedings.mlr.press/v37/barbosa15.html.

[15] ——, "A new framework for distributed submodular maximization," in *IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016, pp. 645–654.

[16] B. Mirzasoleiman, M. Zadimoghaddam, and A. Karbasi, "Fast distributed submodular cover: Public-private data summarization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[17] P. Liu and J. Vondrak, "Submodular optimization in the MapReduce model," in *2nd Symposium on Simplicity in Algorithms(SOSA 2019)*, vol. 69, 2018, 18:1–18:10. [Online]. Available: `http://drops.dagstuhl.de/opus/volltexte/2018/10044`.

[18] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," in *Advances in Neural Information Processing Systems (NeurIPS*, vol. 26, 2013, pp. 2049–2057. [Online]. Available: `https://proceedings.neurips.cc/paper/2013/file/84d2004bf28a2095230e8e14993d398d-Paper.pdf`.

[19] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Foundations and Trends in Machine Learning*, vol. 6, pp. 145–373, 2013. DOI: `10.1561/2200000039`.

[20] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2003, pp. 137–146. [Online]. Available: `https://doi.org/10.1145/956750.956769`.

[21] D. Dueck and B. J. Frey, "Non-metric affinity propagation for unsupervised image categorization," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007.

[22] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2007, pp. 420–429. [Online]. Available: `https://doi.org/10.1145/1281192.1281239`.

[23] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, 2009, pp. 5–14. [Online]. Available: `https://doi.org/10.1145/1498759.1498766`.

[24] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin, "Turning down the noise in the blogosphere," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2009, pp. 289–298. [Online]. Available: `https://doi.org/10.1145/1557019.1557056`.

[25] H. Lin and J. Bilmes, "Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies," vol. 1, Association for Computational Linguistics, 2011, pp. 510–520.

[26] A. Das and D. Kempe, "Submodular meets spectral: Greedy algorithms for subset selection, sparse, approximation and dictionary selection," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 1057–1064.

[27] K. El-Arini, G. Veda, and C. Guestrin, "Beyond keyword search: Discovering relevant scientific literature," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2011, pp. 439–447. [Online]. Available: `https://doi.org/10.1145/2020408.2020479`.

[28] H. Lin and J. Bilmes, "Optimal selection of limited vocabulary speech corpora," in *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 2012, pp. 479–490.

[29] S. Tschiatschek, R. Iyer, H. Wei, and J. Bilmes, "Learning mixtures of submodular functions for image collection summarization," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014.

[30] A. Das, A. Dasgupta, and R. Kumar, "Selecting diverse features via spectral regularization," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1592–1600.

[31] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in *ACM Transactions on Knowledge Discovery from Data*, vol. 5, 2012, pp. 1–37.

[32] R. Sipos, A. Swaminathan, P. Shivaswamy, and T. Joachims, "Temporal corpus summarization using submodular word coverage," in *21st ACM International Conference on Information and Knowledge Management (CIKM)*, 2012.

[33] A. Dasgupta, R. Kumar, and S. Ravi, "Summarization through submodularity and dispersion," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 1014–1022. [Online]. Available: https://aclanthology.org/P13-1100.

[34] R. B. Bairi, R. Iyer, G. Ramakrishnan, and J. Bilmes, "Summarization of multi-document topic hierarchies using submodular mixtures," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL—IJCNLP)*, vol. 1, 2015, pp. 553–563.

[35] J. Zheng, Z. Jiang, R. Chellappa, and J. P. Phillips, "Submodular attribute selection for action recognition in video," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 1341–1349. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/b056eb1587586b71e2da9acfe4fbd19e-Paper.pdf.

[36] S. Kothawade, N. Beck, K. Killamsetty, and R. Iyer, "Similar: Submodular information measures based active learning in realistic scenarios," 2021.

[37] S. Kothawade, V. Kaushal, G. Ramakrishnan, J. Bilmes, and R. Iyer, "Prism: A rich class of parameterized submodular information measures for guided subset selection," 2021, arXiv:2103.00128.

[38] R. Iyer, "Submodular optimization and machine learning: Theoretical results, unifying and scalable algorithms and applications," Ph.D. dissertation, University of Washington, 2015.

[39] R. Iyer, N. Khargonkar, J. Bilmes, and H. Asnani, "Submodular combinatorial information measures with applications in machine learning," in *Proceedings of Machine Learning Research and 32nd International Conference on Algorithmic Learning Theory*, vol. 132, 2021, pp. 722–754. [Online]. Available: https://proceedings.mlr.press/v132/iyer21a.html.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 8026–8037.

[41] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, vol. 19, 2020, pp. 1597–1607. [Online]. Available: https://proceedings.mlr.press/v119/chen20j.html.

A Proofs

Theorem 1. Theorem 1 is a consequence of Theorem 2, taking $M = 1$. □

Submodularity, nonnegativity, and increasing monotonicity of $f_{\hat{\pi}}$. As in (4), we define the value of adding a new point (x, y) to the currently labeled set $\mathcal{L}$ to be

$$f_{\hat{\pi}}(\mathcal{L} \cup \{(x,y)\}) = \sum_{k\in[K]} \hat{\pi}_k(x) g\left(\sum_{z\in\mathcal{L}\cup\{(x,y)\}} \hat{\pi}_k(z_x)\right).$$

Nonnegativity and increasing monotonicity of $f_{\hat{\pi}}$ are obvious. Submodularity, which we now show, follows from the concavity of g.

Let $S \subseteq T \subseteq \mathcal{X} \times \mathcal{Y}$ and $x \in \mathcal{X} \times \mathcal{Y} \backslash T$. Since g is a concave function, its secant lines have decreasing slope. Therefore, $\forall\ \hat{\pi}_{k\in[K]}(x) > 0$,

$$\frac{g\Big(\sum_{z\in S\cup\{(x,y)\}} \hat{\pi}_k(z_x)\Big) - g\Big(\sum_{z\in S} \hat{\pi}_k(z_x)\Big)}{\hat{\pi}_k(x)} \geq \frac{g\Big(\sum_{z\in T\cup\{(x,y)\}} \hat{\pi}_k(z_x)\Big) - g\Big(\sum_{z\in T} \hat{\pi}_k(z_x)\Big)}{\hat{\pi}_k(x)}.$$

It then follows, using the definition of $f_{\hat{\pi}}$ for the equalities, that

$$\begin{aligned}
f_{\hat{\pi}}(S \cup \{(x,y)\}) - f_{\hat{\pi}}(S) &= \sum_{k\in[K]} \hat{\pi}_k(x) \left[g\left(\sum_{z\in S\cup\{(x,y)\}} \hat{\pi}_k(z_x)\right) - g\left(\sum_{z\in S} \hat{\pi}_k(z_x)\right)\right] \\
&\geq \sum_{k\in[K]} \hat{\pi}_k(x) \left[g\left(\sum_{z\in T\cup\{(x,y)\}} \hat{\pi}_k(z_x)\right) - g\left(\sum_{z\in T} \hat{\pi}_k(z_x)\right)\right] \\
&= f_{\hat{\pi}}(T \cup \{(x,y)\}) - f_{\hat{\pi}}(T),
\end{aligned}$$

which proves submodularity of $f_{\hat{\pi}}$. (Note that if $\hat{\pi}_k(x) = 0$ for any $k \in [K]$, we simply remove that term from the sums above and the argument still holds). Using the same sequence of steps, the slightly-modified value function we use for our experiments,

$$f_{\hat{\pi}}(\mathcal{S} \cup \{(x,y)\}) = \sum_{k\in[K]} \hat{\pi}_k(x) g(1 + |\{(x,y) \in S : y = k\}|),$$

can be shown to be submodular. □

Theorem 2. For each $j \in [M]$, let $\mathcal{D}_j$ be the j'th agent's unlabeled data stream, let $\tau_{j,x}$ be the j'th agent's threshold value corresponding to arbitrary point $x \in \mathcal{D}_j$, let

$$\mathcal{L}_j = \{l_{j,1}, ..., l_{j,s_j}\}$$

be agent j's selected set under `DMGT`, let

$$\mathcal{L} = \cup_{j\in[M]} \mathcal{L}_j$$

and let

$$\mathcal{V}_j = (\mathrm{OPT}(\mathcal{D}, |\mathcal{L}|) \cap \mathcal{D}_j) \backslash \mathcal{L}_j = \{v_{j,1}, ..., v_{j,m_j}\}.$$

Following an initial proof strategy used for Theorem 5.1 in [12],

$$0 > \sum_{j\in[M]}\sum_{i\in[m_j]} f(\mathcal{L}_{j<v_{j,i}} \cup \{v_{j,i}\}) - f(\mathcal{L}_{j<v_{j,i}}) - \tau_{j,v_{j,i}} \qquad \text{Alg. 1(line 6); def. of } \mathcal{V}$$

$$\geq \sum_{j\in[M]}\sum_{i\in[m_j]} f(\mathcal{L}_j \cup \mathcal{V}_{j<v_{j,i}} \cup \{v_{j,i}\}) - f(\mathcal{L}_j \cup \mathcal{V}_{j<v_{j,i}}) - \tau_{j,v_{j,i}} \qquad \text{submod. of } f$$

$$\geq \sum_{j\in[M]} f(\mathcal{L}_j \cup \mathcal{V}_j) - f(\mathcal{L}_j) - \tau_{\max}|\mathcal{V}_j| \qquad \tau_{\max} \geq \tau_t \ \forall t \in [|\mathcal{D}|]\text{; telescope sum}$$

$$\geq f(\cup_{j\in[M]}\mathcal{L}_j \cup \mathcal{V}_j) - \tau_{\max}(|\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)| - |\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|) \cap \mathcal{L}|) - \sum_{j\in[M]} f(\mathcal{L}_j) \qquad \text{subadd. of nonneg., submod. } f$$

$$\geq f(\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)) - \tau_{\max}(|\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)| - |\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|) \cap \mathcal{L}|) - \sum_{j\in[M]} f(\mathcal{L}_j). \qquad \text{inc. mon. of } f \qquad (9)$$

Note also that for all $j \in [M]$, we have

$$\begin{aligned} f(\mathcal{L}_j) &= \Big[\sum_{i\in[s_j]} f(\mathcal{L}_{<l_{j,i}} \cup \{l_{j,i}\}) - f(\mathcal{L}_{<l_{j,i}})\Big] \\ &> \Big[\sum_{i\in[s_j]} \tau_{j,l_{j,i}}\Big] && \text{Alg.1(line 6)} \\ &\geq \tau_{\min}|\mathcal{L}_j|. && \tau_{\min} \leq \tau_t \ \forall t \in [|\mathcal{D}|] \qquad (10) \end{aligned}$$

Combining (9) and (10),

$$\begin{aligned} \sum_{j\in[M]} f(\mathcal{L}_j) &\geq \tau_{\min}|\mathcal{L}| && \text{by (10)} \\ &= \frac{\tau_{\min}}{\tau_{\max}}\tau_{\max}|\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)| \\ &\geq \frac{\tau_{\min}}{\tau_{\max}}\Big[f(\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)) + \tau_{\max}|\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|) \cap \mathcal{L}| - \sum_{j\in[M]} f(\mathcal{L}_j)\Big] && \text{by (9),} \end{aligned}$$

which implies that

$$\sum_{j\in[M]} f(\mathcal{L}_j) \geq \frac{\tau_{\min}}{\tau_{\min}+\tau_{\max}} f(\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)) + \frac{\tau_{\min}\tau_{\max}}{\tau_{\min}+\tau_{\max}}|\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|) \cap \mathcal{L}|. \qquad (11)$$

Finally, let $j^* = \arg\max_{j\in[M]} f(\mathcal{L}_j)$. Then

$$\begin{aligned} f(\mathcal{L}) &\geq f(\mathcal{L}_{j^*}) && \text{inc. mon. of } f \\ &\geq \frac{1}{M}\sum_{j\in[M]} f(\mathcal{L}_j) && \text{def. of } j^* \\ &\geq \frac{\tau_{\min}}{M(\tau_{\min}+\tau_{\max})} f(\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|)) + \frac{\tau_{\min}\tau_{\max}}{M(\tau_{\min}+\tau_{\max})}|\mathrm{OPT}(f,\mathcal{D},|\mathcal{L}|) \cap \mathcal{L}|, && \text{by (11)} \end{aligned}$$

concluding the proof. □

B Batch-DMGT

DMGT can be extended to a batch setting in which the value function changes between batches. This extension is useful when the analyst wants to update the value function with information from past selection rounds.

For instance, suppose f depends on a model which is being continually updated on selected data from earlier batches, inducing a sequence of value functions over batches. We discuss such a setting in Section 2.3.

Algorithm 3 `Batch-DMGT`

Input Number of batches B; for each batch: data stream $\mathcal{D}_b \subset \mathcal{X}$, value function $f_b : 2^{\mathcal{D}_b} \to \mathbb{R}^+$

Set of selected points $\mathcal{L} = \emptyset$
for $b = 1, .., B$ **do**
    $\mathcal{L}_b, \mathcal{T}_b \leftarrow \texttt{DMGT}(\mathcal{D}_b, f_b)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}_b$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_b$
return $\mathcal{L}, \mathcal{T}$

Corollary 1 (Near-optimality of `Batch-DMGT`)**.** *Suppose* `Batch-DMGT` *runs on B arbitrary data streams $\{\mathcal{D}_b\}_{b\in[B]}$ with accompanying value functions $\{f_b\}_{b\in[B]}$, and constructs per-batch threshold sets $\{\mathcal{T}_b\}_{b\in[B]}$. Then for each $b \in [B]$*

$$f_b(\mathcal{L}_b) \geq \frac{\tau^b_{\min}}{\tau^b_{\min} + \tau^b_{\max}} f(\mathrm{OPT}(f_b, \mathcal{D}_b, |\mathcal{L}_b|)) + \frac{\tau^b_{\min}\tau^b_{\max}}{\tau^b_{\min} + \tau^b_{\max}} |\mathrm{OPT}(f_b, \mathcal{D}_b, |\mathcal{L}_b|) \cap \mathcal{L}_b|, \tag{12}$$

where $\tau^b_{\min}$ and $\tau^b_{\max}$ are the minimum and maximum elements of $\mathcal{T}_b$ respectively, and

$$f_B(\mathcal{L}) \geq \frac{\tau_{\min}}{B(\tau_{\min} + \tau_{\max})} f(\mathrm{OPT}(f_B, \mathcal{D}, |\mathcal{L}|)) + \frac{\tau_{\min}\tau_{\max}}{B(\tau_{\min} + \tau_{\max})} |\mathrm{OPT}(f_B, \mathcal{D}, |\mathcal{L}|) \cap \mathcal{L}|, \tag{13}$$

where $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum elements of $\mathcal{T}$ respectively.

Corollary 1. Inequality (12) is a direct consequence of Theorem 1. The proof for inequality (13) is identical to the proof of Theorem 2, with j indexing the batches $[B]$ instead of agents $[M]$. □

C Relevant Definitions and Theorems

Definition 3 (Monotonicity)**.** *A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is* monotone increasing *if* $\forall S \subseteq T$ *with* $S, T \subseteq \mathcal{X}$

$$f(S) \leq f(T)$$

and monotone decreasing *if* $\forall S \subseteq T$ *with* $S, T \subseteq \mathcal{X}$

$$f(S) \geq f(T)$$

Definition 4 (Sub-additivity)**.** *A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is* sub-additive *if* $\forall S, T \subseteq \mathcal{X}$

$$f(S \cup T) \leq f(S) + f(T)$$

Definition 5 (Equivalent definition of submodularity)**.** *A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is* submodular *if* $\forall S, T \subseteq \mathcal{X}$

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

Theorem C.1 (Sub-additivity of nonnegative submodular functions)**.** *All nonnegative submodular functions are sub-additive.*

Proof of Theorem C.1. Let $f : 2^{\mathcal{X}} \to \mathbb{R}^+$ be a nonnegative submodular function. Then, by Definition 5, $\forall S, T \subseteq \mathcal{X}$, we have

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \geq f(S \cup T).$$

□