Online k-means Clustering

Vincent Cohen-Addad
CNRS
vcohenaddad@gmail.com

Benjamin Guedj
Inria & University College London
benjamin.guedj@inria.fr

Varun Kanade
University of Oxford
varunk@cs.ox.ac.uk

Guy Rom

guy.rome@gmail.com

Abstract

We study the problem of learning a clustering of an online set of points. The specific formulation we use is the k-means objective: At each time step the algorithm has to maintain a set of k candidate centers and the loss incurred by the algorithm is the squared distance between the new point and the closest center. The goal is to minimize regret with respect to the best solution to the k-means objective in hindsight. We show that provided the data lies in a bounded region, learning is possible, namely an implementation of the Multiplicative Weights Update Algorithm (MWUA) using a discretized grid achieves a regret bound of $\widetilde{O}(\sqrt{T})$ in expectation. We also present an online-to-offline reduction that shows that an efficient no-regret online algorithm (despite being allowed to choose a different set of candidate centers at each round) implies an offline efficient algorithm for the k-means problem, which is known to be NP-hard. In light of this hardness, we consider the slightly weaker requirement of comparing regret with respect to $(1+\epsilon)$OPT and present a no-regret algorithm with runtime $O\left(T\text{poly}(\log(T), k, d, 1/\epsilon)^{O(kd)}\right)$. Our algorithm is based on maintaining a set of points of bounded size which is a coreset that helps identifying the *relevant* regions of the space for running an adaptive, more efficient, variant of the MWUA. We show that simpler online algorithms, such as *Follow The Leader* (FTL), fail to produce sublinear regret in the worst case. We also report preliminary experiments with synthetic and real-world data. Our theoretical results answer an open question of Dasgupta (2008).

1 Introduction

Clustering algorithms are one of the main tools of unsupervised learning and often form a key part of the data analysis pipeline. Unlabeled data is ubiquitous and discovering structure in such data is essential in several online applications. This work focuses on the *online* setting where data arrive one at a time and need to be assigned to a cluster (either new or existing) without the benefit of having observed the entire sequence, or knowing the "correct" labels of the elements that have already arrived. Hence, our setting captures fully *unsupervised* scenarios where the user does not have to provide any feedback to the decisions made by the algorithm at any point in time. We model the accuracy of a given clustering using the popular k-means objective, although our results can be generalized to most center-based objectives.

The analysis of online algorithms comes in two flavors involving bounding either the *competitive ratio*, or the *regret*. An online algorithm makes irrevocable decisions and its performance is measured by the value of an objective function. The competitive ratio is the ratio between the objective value achieved by the online algorithm and the best offline solution (for minimization problems). In the case of clustering, without strong assumptions on the instance, no algorithms with non-trivial bounds on the competitive ratio can be designed. In *regret analysis*, the difference between the objective value of the online algorithm and the best offline so-

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

lution (in hindsight) is sought to be bounded by a function that grows sublinearly with the number of data elements. We focus on the latter, seeking to answer the following fundamental question, also asked by Dasgupta (2008): *Can an online algorithm learn the optimal clustering of a dataset?*

More precisely, in the case of online k-means clustering, the online algorithm at time t produces a set of k candidate cluster centers, $C_t = \{c_{t,1}, \ldots, c_{t,k}\}$ before observing the datum x_t that arrives at time t. The loss incurred by the algorithm at time t is $\ell_t(C_t, x_t) = \min_{c \in C_t} \|x_t - c\|_2^2$. The *regret* is the difference between the cumulative loss of the algorithm over T time steps and the optimal fixed solution in hindsight, i.e. the optimal k-means solution for the entire input. More formally, the regret is defined as,

$$\text{Regret} = \sum_{t=1}^{T} \ell(C_t, x_t) - \min_{C:|C|=k} \sum_{t=1}^{T} \min_{c \in C} \|x_t - c\|_2^2.$$

Our Contributions: We consider the setting where the data x_t all lie in the unit box $[0,1]^d \subseteq \mathbb{R}^d$ (any bounded box suffices). We summarize our principal contributions:

1. We show that a multiplicative weight-update algorithm (MWUA) over size k sets of candidate centers drawn from a grid over $[0,1]^d$ achieves expected regret $\widetilde{O}(\sqrt{T})$; the $\widetilde{O}(\cdot)$ notation hides factors that are poly-logarithmic in T and polynomial in k and d. The algorithm and its analysis are along standard lines; although the algorithm is computationally inefficient, it shows that information-theoretically achieving $\widetilde{O}(\sqrt{T})$ regret is possible.
2. We observe that an online-to-offline reduction that shows that any online algorithm that runs in time $f(t,k,d)$ at time t, yields an offline algorithm that solves the k-means problem to additive accuracy ϵ in time polynomial in n, k, d, $1/\epsilon$, $f(n,k,d)$. In particular, for an offline instance of k-means with n points in a bounded region with $\mathsf{OPT} \geq 1/\text{poly}(n)$, an online algorithm with polynomial running time would yield a fully poly-time approximation scheme. We note that there exist such instance families for k-means which are known to be APX-hard (Awasthi et al., 2015). This suggests we must relax performance requirements for efficient algorithms.
3. As our main result, in light of the above hardness result, we consider a weaker notion called $(1+\epsilon)$-regret: Let OPT denote the loss of the best solution in hindsight and L_T the cumulative loss of the algorithm; define $(1+\epsilon)$-regret of the algorithm as $L_T - (1+\epsilon)\mathsf{OPT}$. We provide an algorithm that achieves $(1+\epsilon)$-regret $\widetilde{O}(\sqrt{T})$ and runs in time $T^{1+o(1)} \cdot \epsilon^{-O(k^3 d^3)}$, which is near-linear in the total number of samples, and fixed-parameter tractable in d and k.
4. As a means to achieve the above algorithmic guarantees, we introduce two techniques that we believe may be of independent interest. (i) An algorithm that maintains an insertion only coreset for k-means for any substream. (ii) A variant of the MWUA algorithm that allows an adaptive set of experts, that is not known in advance.
5. Finally, we consider online algorithms that have oracle access to a k-means solver. For instance, this allows us to implement the *Follow the Leader* (FTL) algorithm. We show that there exists data sequences for which FTL has *linear* regret. We show that this construction indeed results in linear regret in simulations, but also observe that FTL (using k-means++ as a proxy for oracle) works rather well on *random stream* real-world data.

Related Work: Clustering has been studied from various perspectives, e.g. combinatorial optimization, probabilistic modeling, and there are several widely used algorithms such as Lloyd's local search algorithm with k-means++ seeding, spectral methods, the EM algorithm, etc. A popular way to model clustering as a combinatorial optimization problem that we follow in this paper is through the k-means objective. The k-means objective is one of the family of center-based objectives which uses the squared distance to the center as a measure of variance. Optimizing the k-means objective is NP-complete, and assuming the Exponential Time Hypothesis of Impagliazzo et al. (1998), Cohen-Addad et al. (2018) showed that there is no exact algorithm running in time $f(k)n^{o(k)}$ even for 4-dimensional Euclidean inputs, for any computable function f. As a result, theoretical work has focused on approximation algorithms. Reviewing the entire literature on the approximation algorithms for k-means clustering is beyond the scope of this paper. For Euclidean inputs, the most important results are the best known polynomial time algorithm of Ahmadian et al. (2017) which produces a 6.43-approximation and the $(1+\varepsilon)$-approximation algorithm of Cohen-Addad et al. (2019) which runs in time $f(k,\varepsilon)n^{O(1)}$. Designing approximation algorithm with bounded competitive ratio for k-means in the online setting is not possible, thus Liberty et al. (2016) have proposed a *bicriteria* algorithm for the k-means problem in the online setting: the algorithm produces a clustering with $O(k \log n)$ centers and achieves a constant approximation ratio with respect to the best k-means clustering using k centers (see also Charikar et al. (2003); Meyerson (2001) for similar algorithms for the k-median and facility location problems in the online setting). For the regret analysis setting, Dasgupta (2008) asked whether one could derive bounded regrets bounds for the online k-

means: we thus answer the question in the affirmative in this paper. Designing algorithms for streaming data is closely related to online algorithms. As in the online setting, the data is received one at a time. The focus in the streaming setting is to have extremely low memory footprint and the algorithm is only required to propose a solution once the stream has been exhausted. In contrast, in the online setting the learning algorithm has to make a decision at each time step and incur a corresponding loss.

Coresets are widely used in computational geometry to obtain approximation algorithms. A coreset for k-means is a mapping of the original data to a subset of the data, along with a weight function, such that the k-means cost of partitions of the data is preserved up to some small error using the given mapping and weights.

There is also a vast literature on online learning with experts and related problems have been widely studied, and we refer to e.g. (Cesa-Bianchi and Lugosi, 2006) and references therein. The *Follow the Leader* (FTL) algorithm always predicts using a solution that would have been optimal in hindsight at any given time. FTL achieves low regret for some problems, typically with strongly convex loss functions (Shalev-Shwartz et al., 2012). Variants of FTL that optimize a regularized objective have been successfully utilized for a wider range of settings. Recent work of Dudik et al. (2017) has successfully modified the follow-the-perturbed leader (FTPL) framework to design oracle efficient algorithms for some combinatorial problems in the online setting. However, although they can deal with an exponentially large number of experts, the *experts* need to be known in advance. This is not easy in our case, as the number of experts is either too *large* if a very fine grid is used, or not known in advance, if we choose centers adaptively using the locations of historical data. In the former case, the separability condition required in their work, which very loosely stated says that the randomness needs to be chosen so that each expert behaves somewhat differently on the "fake data" created by this randomness, cannot be met. The interplay between additive and multiplicative approximations required to derive approximate regret bounds makes applying existing frameworks to our problem challenging.

Choromanska and Monteleoni (2012) address a relaxed version of the problem at hand – a set of batch clustering algorithms calculate sets of k cluster centers at each step, and forward the cluster center closest to the next data point in the stream to an aggregation algorithm, that in turn obtains an approximate regret guarantee of $\log(T)$ for the stream. The relaxation lies in the fact that the auxiliary batch algorithms must observe the next data point in the stream prior to committing to the clustering at step t in order to forward the cluster center *closest to the next point in the stream*. Li et al. (2018) provide a generalized Bayesian adaptive online clustering algorithm (built upon the PAC-Bayes theory). They describe a Gibbs Sampling procedure of $O(k)$ centers and prove it has a minimax sublinear regret. They present a reversible jump MCMC process to sample these centers with no theoretical mixing time analysis. Moshkovitz (2019) studies a related problem where the competitive ratio is sought to be bounded by a constant. However, in this setting the online algorithms is allowed to open more centers than k and the measure of performance is the number of candidate centers opened by the online algorithm, which could vary between $O(k)$ and $\Omega(Tk)$, depending on the instance.

2 Preliminaries and Notation

We consider data in the unit box $[0,1]^d \subseteq \mathbb{R}^d$.[1] The datum arriving at time t is denoted by x_t and we use $X_{1:t-1}$ to denote the data received before time t. The learning algorithm must output a set $C_t \subseteq [0,1]^d$ of k candidate centers using only $X_{1:t-1}$. The loss incurred by the algorithm at time t is $\ell(C_t, x_t) = \min_{c \in C_t} \|x_t - c\|_2^2$. The total loss of an algorithm up to time step t is defined as $L_t = \sum_{\tau=1}^{t} \ell(C_\tau, x_\tau)$. The loss of the best k-means solution in *hindsight* after T steps is denoted by OPT. The regret is defined as $L_T - \mathsf{OPT}$ and, for any $\epsilon > 0$, the $(1+\epsilon)$-approximate regret as $L_T - (1+\epsilon)\mathsf{OPT}$. The algorithms we design pick cluster centers from a constrained set, whose elements are referred to as *sites*, and multisets of k such sites as *experts*. The $\widetilde{O}$ notation hides factors polynomial in $k, d, \log(T)$.

We denote the best k-means solution, i.e. best k centers, for $X_{1:t-1}$ by $C_t^\star$, hence $C_{T+1}^\star$ is the best k-means solution in hindsight. In this case, the *Follow-The-Leader* (FTL) algorithm simply picks $C_t^\star$ at time t.

The *weighted* k-means loss of a set of cluster centers $\boldsymbol{\mu}$, with respect to data X, given a weight function $\omega : X \to \mathbb{R}_+$, is defined as $\ell(\boldsymbol{\mu}, X) = \sum_{x \in X} \omega(x) \min_{\mu \in \boldsymbol{\mu}} \|x - \mu\|_2^2$.

3 The MWUA Algorithm and Lower Bounds

3.1 MWUA with Grid Discretization

As a warm-up, we provide a simple but inefficient approach for the problem based on the multiplicative weights update algorithm (MWUA). While the MWUA is very widely applicable, some difficulties arise when

[1] A B-bounded box is sufficient for our results to hold, but we will get bounds that are worse by a factor polynomial in B.

applying it to our problem. In order to obtain a finite set of experts, we consider any choice of k *sites* obtained by a δ-grid of $[0,1]^d$. In order to obtain meaningful regret bounds, i.e. sublinear in T, then for any $0 < \alpha \leq 1/2$, obtaining a regret of $\widetilde{O}(T^{1-\alpha})$ requires $\delta = T^{-\alpha/2}$. Thus the number of experts is at least $T^{\Omega(kd)}$, i.e. exponential in k and d. However, as the regret of MWUA only has logarithmic dependence on the number of experts, this results in a computational, rather than information-theoretic cost.[2] Of course the standard lower bound of $\Omega(\sqrt{T})$ regret still applies as the usual worst-case example can easily be formulated in our setting.

In Section 5, we apply the MWUA to our problem in a more *data-adaptive* manner, which allows us to significantly reduce the number of *sites* required—we don't put sites in location where there is no data—but also requires a much more intricate analysis. The price we pay for adaptivity and *computational efficiency* is that we are only able to get regret bounds on the $(1+\epsilon)$-regret. A simple result in Section 3.2 shows that this is unavoidable, assuming RP $\neq$ NP. The proof of the following is provided in Appendix A.1.

Theorem 1. *Let $S = \{i\delta \mid 0 \leq i \leq \delta^{-1}\}^d \subset \mathbb{R}^d$ be the set of* sites *and let $\mathcal{E} = \{C \subset S \mid |C| = k\}$ be the set of experts (k-centers chosen out of the sites). Then, for any $0 < \alpha \leq 1/2$, with $\delta = T^{-\alpha/2}$, the* MWUA *with the expert set $\mathcal{E}$ achieves regret $O(kd \ln(T) T^{1-\alpha})$; the per round running time is $O(T^{\alpha kd/2})$.*

3.2 Lower Bound

Given the disappointing runtime of the (grid) MWUA algorithm, one may wonder whether there is a way to avoid explicitly storing a weight for each of the exponentially many experts and speed-up the MWUA algorithm. The following result gives evidence that it is unlikely that a significant speed-up is possible, assuming RP $\neq$ NP. In particular, a consequence of Theorem 2 is that for instances of k-means with data lying in a bounded region and OPT $\geq 1/\text{poly}(n)$, a per-round polynomial time online algorithm would imply an FPRAS.

Theorem 2. *Suppose there is an online k-means clustering algorithm $\mathcal{A}$ that achieves regret $\widetilde{O}(T^{1-\alpha})$ after T time steps, and runs in time $f(t,k,d)$ at each time step $t \leq T$, where f is non-decreasing in t. Then, for any $\epsilon > 0$, there is a randomized offline algorithm that given an instance of k-means outputs a solution with cost at most* OPT $+\epsilon$ *with constant probability and runs in time polynomial in $n, k, d, \frac{1}{\epsilon}, f(n,k,d)$.*

The proof follows along standard lines and appears in Appendix A.2. Together with the existence of instance families for k-means with OPT $\geq 1/\text{poly}(n)$ which are known to be APX-hard (Awasthi et al., 2015) we can conclude that a per round polynomial-time algorithm for the online k-means problem does not exist unless RP = NP.

[2] There appears to be some information-theoretic cost in that we are only able to prove bounds on expected regret.

4 Follow The Leader : Lower Bounds

The *Follow the Leader* (FTL) algorithm is an online algorithm that simply picks the "best in hindsight" solution at time t, and gives optimal regret bounds in some cases, e.g. when optimizing strongly convex functions. In particular, it follows that the $k = 1$ version of the online k-means problem can be solved optimally using FTL. If we have oracle access to the k-means problem, one might wonder whether FTL achieves good regret bounds for $k > 1$. The previous lower bound is not applicable due to the oracle access provided to the online algorithm. We show that any such optimism is misguided, at least in the worst-case, and establish a linear regret lower bound for FTL in Theorem 3, a proof of which appears in Appendix B.

Theorem 3. FTL *incurs $\Omega(T)$ regret in the worst case, for any fixed $k \geq 2$ and any dimension $d \geq 1$.*

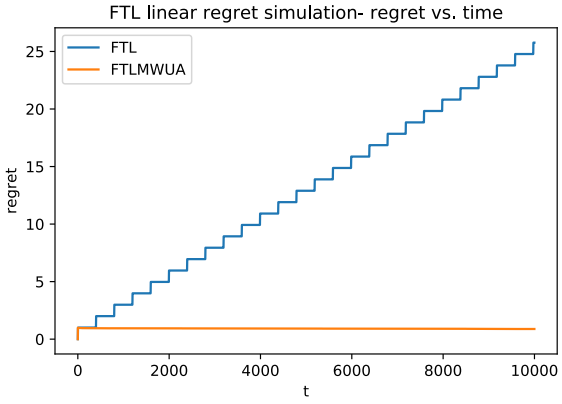

Figure 1: Regret– FTL Linear, MWUA-FTL Sublinear.

We performed some simulations, in part to demonstrate Theorem 3, but mainly to understand the behavior of FTL on synthetic and real-world data that actually exhibits cluster structure. For the counter-example constructed in Theorem 3 it is possible to exactly compute the k-means solution. For other data, we use the k-means++ implementation in `sklearn` as a proxy for a k-means oracle, set to run the k-means++ algorithm 300 times and to output the solution with minimum k-means cost among them.

Figure 1 contains the results of running two algorithms on the data stream generated using the sequence used

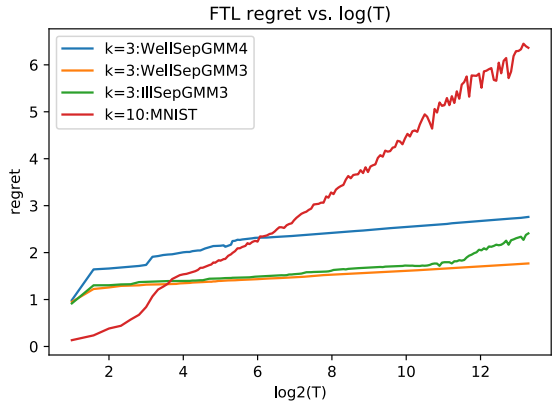

Figure 2: FTL Regret–MNIST and Gaussian Mixtures.

to prove Theorem 3. The *regret* at time t is computed using the "best in hindsight" solution on the data stream $X_{1:t}$. The first algorithm is an implementation of FTL; the second is an implementation of MWUA using the expert sets which C_t^* produced by FTL at time t, this is referred to as MWUA-FTL. Both these algorithms can be easily implemented exactly for this simple one-dimensional problem. The staircase-like line for FTL is caused by the fact that the specific data used makes FTL suffer a constant excess loss (w.r.t. the optimal solution) every few rounds, and negligible excess loss in the remaining rounds. This demonstrates that the sequence suggested by Theorem 3, though artificial, is numerically stable and the predicted behavior manifests itself in simulations. The MWUA-FTL presents low asymptotic regret in this case, which is clearly sub-linear and possibly logarithmic in T.

We also experimented with the behavior of "FTL" on other synthetic and natural data (using k-means++ as a proxy for the optimal solution as mentioned before). The experiments suggest that when the data exhibits meaningful cluster structure and arrives in "random" order, FTL actually performs quite well. Figure 2 shows the time t of FTL vs. $\log(t)$ for four different data sets, all of size 10000. The first is a random sample from MNIST,normalized to a unit diameter box, using $k = 10$. The others three are Gaussian Mixture Models (GMM) with 3 Gaussians (GMM3) or 4 Gaussians (GMM4) in two dimensions, with standard deviation for each Gaussian set to 0.1. The GMM4 case was run with $k = 3$, where the 4 Gaussians are well separated (distance between the means is at least 3 times the standard deviation) hence labeled WellSepGMM4. The two GMM3 data sets are labeled WellSepGMM3 for the well separated case, and IllSepGMM3 for a case where the Gaussians are ill separated (distance between the means is at most 0.7 times the standard deviation). In all cases the "best" k-means solution was calculated using k-means++ in `sklearn` with 300 iterations of local search.

We conjecture that this behavior arises from the fact that once the k clusters are visible in the data, the FTL algorithm is essentially solving k parallel online clustering problems with $k = 1$. In this case, FTL is performing well since the problem for $k = 1$ is convex. Of course, this behavior relies on the stability of the data stream and most likely the order in which data arrives. Theorem 3 clearly shows that this behavior is not always exhibited. It would be interesting to identify stability conditions under which FTL could be shown to be optimal or near optimal for this problem.

5 Approximate Regret Minimization

In this section, we present our main result, an algorithm that minimizes the $(1 + \varepsilon)$-regret for the online clustering problem. The design of the algorithm and its analysis are intricate. We begin by giving a high-level overview with some intuition. The proofs of the lemmas are deferred to the Appendix.

The algorithm uses three main components which we describe below.

1. An algorithm described in Section 5.1 that maintains a monotone sequence of sets $\{\mathcal{Q}_t\}_{t=0}^T$, referred to as an *Incremental* ε_{c}*-Coreset*, as $\mathcal{Q}_t$ is a weighted $\varepsilon_{\mathsf{c}}-$coreset for $X_{1:t}$. Roughly speaking, an ϵ_c-coreset C for a set of points X is a set of points with $|C| \ll |X|$ together with a weight function ω, such that for any set of k points in $\mathbb{R}^d$, the weighted k-means cost of this solution on C with within a $(1 \pm \epsilon_c)$ factor of the k-means cost of this solution on X. The *incremental* nature is important for designing and efficient online algorithm as this will allows us to efficiently refine the space for candidate location for centers.

   Roughly speaking because a coreset C is a relatively small set, rather than keeping a very fine grid of the unit cube $[0, 1]^d$, we can keep a relatively small set of *candidate* centers such that any solution to the k-means problem is *close enough* (in terms of cost) to a solution that only uses these candidate centers. The incremental coreset allows us to gradually and *adaptively* refine this candidate set of centers. This is achieved using a *hierarchical region decomposition* which is described next.
2. An *Adaptive Hierarchical Region Decomposition* of Section 5.3 corresponding to $\{\mathcal{Q}_t\}_{t=0}^T$ and provides a tree structure $\mathcal{H}_t$ related to $\varepsilon_{\mathsf{hrd}}$-approximations of k-means. We restrict the set of candidate centers to be *centroids* of sub-cubes of the unit cube. Initially, we start with the entire unit cube with only *one*

candidate center. Depending on the current coreset, if the candidate centers are not sufficient to yield a $(1 \pm \varepsilon_{\mathsf{hrd}})$ approximation to some k-means solution (not necessarily optimal) for the coreset, then we subdivide some cube into 2^d subcubes, and recursively keep doing so until such an approximation is achieved. The size of the *coreset* ensures that not too many such subdivisions have to be performed in order to obtain a good solution.

There remains a final challenge that the set of candidate centers increases with time, and as such *experts* or *good solutions* that are available later in the online learning process may not be available earlier in the process. In order to deal with this, rather than deal with a fixed set of experts, we allow the expert set to grow. We do this using a version of the MWUA for tree-structured experts, that is experts that can only branch into new experts, but never merge, described next.

3. We introduce a version of the MWUA for tree structured expert sets in Section 5.4, referred to as *Mass Tree* MWUA (MTMW), that is given $\mathcal{H}_t$ at every step, and outputs a distribution over choices of k cluster centers. Intuitively at time $t = 0$, there is only one expert, with all k centers at the centroid of the unit cube, and that expert has all the weight. When a cube is sub-divided and new centers created, *new experts* may be created which can be uniquely mapped to an expert in the previous round before the subdivision occurred. Intuitively, going forward we "hypothetically pretend" that the experts were always present, but were required to use the centroid that would have been available at that time given the particular coreset. The weight of the original expert is divided evenly between the new experts it creates. This ensures that the simulation of the MWUA algorithm is faithful notwithstanding the addition of new experts. Thus, the weight of an expert may decrease exponentially in terms of the number of divisions it took to reach that expert. The key part of the analysis is showing that the number of divisions is bounded because the size of the corset only increases logarithmically in T. We remark that the regret bound depends on the logarithm of the weight of *best expert* in hindsight and this allows us to prove the main result.

We now describe the algorithm formally and provide a few more technical details. The full proofs of all of these results appear in the Appendix.

Defining $\varepsilon^* := 2^i$ that satisfies $\frac{\varepsilon^2}{2ak^2 \log(T^3\sqrt{d})} \leq \varepsilon^* \leq \frac{\varepsilon^2}{ak^2 \log(T^3\sqrt{d})}$, where $a \geq 34^2$ is some arbitrary constant, we can describe the algorithm. We present our main theorem below and provide proofs in the appendices.

Algorithm 1 ε-Regret Minimization

input: ε, x_t (sequentially)
$t \leftarrow 1$, $\varepsilon_{\mathsf{c}}, \varepsilon_{\mathsf{hrd}} \leftarrow \varepsilon^*(\varepsilon, k, d, T)$
Initialize $\mathcal{H}_0$ and MTMW
for $t \leftarrow 1, T$ **do**
  Obtain C_t from MTMW
  Receive x_t and incur loss $\ell(C_t, x_t)$
  Update $\mathcal{Q}_t$ to represent x_t
  Update $\mathcal{H}_t$ to represent $\mathcal{Q}_t$
  Provide MTMW with $\mathcal{H}_t$ and x_t
end for

Algorithm 2 HRD update step

input: $t, \mathcal{R}_{t-1}, x_t, \varepsilon_{\mathsf{hrd}}$
Let $q(\cdot)$ be the refinement criteria for x_t at t
$\mathcal{R}_t \leftarrow \emptyset, \quad \mathcal{U}_t \leftarrow \mathcal{R}_{t-1}$
while $\mathcal{U}_t \neq \emptyset$ **do**
  Pick and remove a region R from $\mathcal{U}_t$
  if $q(R)$ **then**
    $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup R$
  else
    Halve R in all dimension, resulting with H
    $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup H$
  end if
end while

Theorem 4. *Algorithm 1 has an expected* $(1+\varepsilon)$*-regret of*

$$O\left(k\sqrt{d^3T}\log\left(\frac{kT^3\sqrt{d}}{\varepsilon^2}\right)\right),$$

and runtime of

$$T \cdot O(\sqrt{d}k^2\log(T)\varepsilon^{-2})^{k(d+10)}.$$

We now give a high-level description of the three main components, and then combine them.

5.1 Incremental Coreset

We present an algorithm for maintaining an *explicit incremental coreset*, whose details are provided in Appendix C. An explicit incremental coreset is a sequence of sets $C_1, \ldots, C_T$ together with a sequence of mappings of $\phi_1, \ldots, \phi_T$ such that

1. $C_i \subseteq C_{i+1}$ for all i, and C_i is a coreset for $X_{1:i}$.
2. ϕ_i is a function mapping the first i elements of the stream, $X_{1:i}$ to the elements of C_i
3. ϕ_i is consistent with ϕ_{i+1}, namely for each element x of $X_{1:i}$, $\phi_i(x) = \phi_{i+1}(x)$.

We show the following lemma, whose proof are provided in Appendix C.

Lemma 1. *For any $\varepsilon > 0$, the algorithm described in Section C.1.2 maintains an explicit incremental ε-coreset of size at most $O(k^2\varepsilon_{\mathsf{c}}^{-4}\log^4 T)$.*

In the remaining discussion, we denote by $\chi(X_{1:t})$ the coreset at time t maintained by the algorithm described in Section C.1.2,

5.2 Hierarchical Region Decomposition

A *region decomposition* is a partition $\mathcal{R} = \{R_1, \ldots, R_r\}$ of $[0,1]^d$, each part R_i is referred to as *region*. A *hierarchical region decomposition* (HRD) is a sequence of region decompositions $\{\mathcal{R}_1, \ldots, \mathcal{R}_t\}$ such that $\mathcal{R}_\tau$ is a refinement of $\mathcal{R}_{\tau-1}$, for all $1 < \tau \leq t$. In other words, for all $1 < \tau \leq t$, for all region $R \in \mathcal{R}_\tau$ there exists a region $R' \in \mathcal{R}_{\tau-1}$ such that $R \subseteq R'$.

As the hierarchical region decomposition $\mathcal{H} = \{\mathcal{R}_1, \ldots, \mathcal{R}_t\}$ only partitions existing regions, it allows us to naturally define a tree structure $T_{\mathcal{H}}$, rather than a DAG. The nodes of each level τ of the tree correspond to the regions of $\mathcal{R}_\tau$, and the edges connect each region to the single region that contains it on the previous level. The bottom-level region decomposition is the one induced by the leaves of the tree. Moreover, given a hierarchical decomposition $\mathcal{H}_t = \{\mathcal{R}_1, \ldots, \mathcal{R}_t\}$ and a set of points S of size k, we define the *representative regions of S in $\mathcal{H}_t$* as a sequence of multisets $\{\tilde{R}_\tau\}_{\tau=1}^t$ where $\tilde{R}_\tau = \{R \in \mathcal{R}_\tau | \exists s \in S . s \in R\}$ with the correct multiplicity w.r.t. S. Note that these correspond to a path in $T_{\mathcal{H}}$. We define the *Approximate Centers of S induced by $\mathcal{H}_t$* as the sequence of multisets $\{\tilde{S}_\tau\}_{\tau=1}^t$ that consists of the centers of the representative regions of S in $\mathcal{H}_t$.

5.3 Adaptive Grid Hierarchical Region Decomposition

We provide Algorithm 2 that computes an *Adaptive Grid Hierarchical Region Decomposition* that refines regions only if they are close enough to the stream point seen so far, according to some criteria which we denote $q(\cdot)$, which relates to k-means approximations of the stream data. The details are presented in Appendix C. This decomposition admits a tree with low degree and bounded leaf count, as well as represents k-means approximations, as explained in Appendix C.

Corollary 1. *For the optimal set of candidate centers in hindsight S^* and $\tilde{S}_t$ the approximate centers induced by the Hierarchical Region Distribution at time step t, for an **unweighted** stream $X_{1:t}$*

$$\sum_{t'=1}^{T} \ell(\tilde{S}_{t'}, x_{t'}) \leq (1+\varepsilon_{\mathsf{hrd}})\mathsf{OPT} + kd\Lambda + 2\varepsilon_{\mathsf{hrd}}.$$

5.4 MTMW : MWUA for Tree Structured Experts

We present an algorithm which we name *Mass Tree* MWUA *(*MTMW*)* which obtains low regret in the setting of *Prediction from Expert Advice*, as described in Arora et al. (2012), for a set of experts that has the tree structure that will soon follow. The algorithm will be a modification of the *Multiplicative Weights Algorithm* and we will present a simple modification to the proof of Theorem 2.1 of Arora et al. (2012), to obtain a regret bound.

Let $\ell_1(\cdot,\cdot)$ denote a bounded loss function in $[-1,1]$. Consider $\mathcal{T}_T$, a tree whose leaves are all of depth T and the vertices correspond to expert predictions. The expert set is the set of all paths from the root to the leaves, denoted $\mathcal{P}(\mathcal{T})$. We say that for a path $p = (v_1, \ldots, v_T) \in \mathcal{P}(\mathcal{T})$, the prediction that is associated with p at step t is v_t such that we can write the loss of the path w.r.t. the stream of elements $X_{1:T}$ as $\ell_1(p, X_{1:T}) = \sum_{t=1}^T \ell_1(v_t, x_t)$.

We associate a *mass* to any vertex in $\mathcal{T}_T$ as follows. Define the mass of the root as 1, and the mass of any other node v, denoted $\mathcal{M}(v)$, as $\mathcal{M}(v) = \frac{\mathcal{M}(v')}{\deg(v')}$, where v' is its parent and $\deg(v')$ is the out degree of v'. We define the mass of a path as the mass of the leaf node at the end of the path. before we move on to prove the regret bound, we provide a useful lemma.

Lemma 2 (Preservation of Mass)**.** *Let v be a vertex in the tree, $\tilde{\mathcal{T}}$ a subtree of $\mathcal{T}$ with v as root, and $\tilde{V}$ the leaves of $\tilde{\mathcal{T}}$, then $\mathcal{M}(v) = \sum_{v' \in \tilde{V}} \mathcal{M}(v')$.*

We move on to bound the regret of the algorithm.

Proposition 1. *For a tree $\mathcal{T}_T$ running* MWUA *over the expert set that corresponds to the paths of the final tree, $\mathcal{P}(\mathcal{T}_T)$, is possible even if the tree is known up to depth t at any time step t, provided the initial weight for path p is modified to $\mathcal{M}(p)$. The algorithm has a regret with respect to any path $p \in \mathcal{P}(\mathcal{T}_T)$ of*

$$\sqrt{-T\ln(\mathcal{M}(p))}$$

and runs in time $O(|\mathcal{T}_T|)$, the number of vertices of $\mathcal{T}_T$.

Corollary 2. MTMW *for a loss function bounded by d obtains a regret of $d\sqrt{-T\ln(\mathcal{M}(p))}$ by using a normalized loss function.*

5.5 Approximate Regret Bound

We will now combine the three components described above to form the final algorithm. First, we will show the main property of a Hierarchical Region Decomposition that is constructed according to the points

that are added to form the sequence $\{\mathcal{Q}_t\}_{t=1}^T$, which is analogous to Lemma 1. Next we define the *k-tree* structure that corresponds to a Hierarchical Region Decomposition, and show that MTMW performs well on this k-tree. Lastly we show that an intelligent choice of parameters for ε_{c} and $\varepsilon_{\mathsf{hrd}}$ allows Algorithm 1 to obtain our main result, Theorem 4.

Lemma 3. *Let $\mathcal{H}$ be a Hierarchical Region Decomposition with parameter $\varepsilon_{\mathsf{hrd}}$ that was constructed according to $\{\mathcal{Q}_t\}_{t=1}^T$. For S^* the best k cluster centers in hindsight and $\tilde{S}_t$ the approximate centers of S^* induced by $\mathcal{H}$ we have that*

$$\sum_{t=1}^{T} \ell(\tilde{S}_t, x_t) \leq (1 + \varepsilon_{\mathsf{c}} + 8(\varepsilon_{\mathsf{hrd}} + \varepsilon_{\mathsf{c}})k\Lambda)\mathsf{OPT} + dk\Lambda.$$

Consider the region tree structure $T_{\mathcal{H}_t}$ described in Section 5.2 that corresponds to the Hierarchical Region Decomposition $\mathcal{H}_t$ at step t defined in (3). We define a *k-region tree induced by $\mathcal{H}_T$* as a level-wise k-tensor product of $T_{\mathcal{H}_T}$, namely, a tree whose vertices at depth t correspond to k-tuples of vertices of level t of $T_{\mathcal{H}_T}$. A directed edge from a vertex $(v_1 \otimes \ldots \otimes v_k)$ of level t to vertex $(u_1 \otimes \ldots \otimes u_k)$ of level $t+1$ exists iff all the edges (v_i, u_i) exists in $T_{\mathcal{H}_T}$ for every $i \in 1 \ldots k$. We define the *k-center tree induced by $\mathcal{H}_T$* or k-tree, as a tree with the same topology as the k-region tree, but the vertices correspond to multiset of centroids, rather than tensor products of regions, i.e. the k-region tree vertex $(v_1 \otimes \ldots \otimes v_k)$ corresponds to the k-tree vertex $v = [\mu(v_1), \ldots, \mu(v_k)]$ where $\mu(\cdot)$ is the centroid of the given region. The representative regions of any set S of k cluster centers correspond to a path in the k-region tree, and the approximate centers correspond to equivalent path in the k-tree. Importantly, note that Lemma 3 proves that there exists a path in the k-tree such that the loss of the sequence of approximate centers it contains is close to OPT as described therein.

We will now analyze the run of MTMW on the aforementioned k-tree. Let p^* denote the k-tree path that corresponds to the best cluster centers in hindsight.

Lemma 4. *Using the definitions from Lemmas 4, 5 we have that* $-\ln(\mathcal{M}(p^*)) \leq k^2\Lambda\beta$.

Finally, using our choice of ε^* we obtain Theorem 4. The complete proof is provided in Appendix C.

6 Discussion

The online k-means clustering problem has been studied in a variety of settings; in the setting considered in this work, we have shown that no efficient algorithm with sublinear regret exists, even in the Euclidean setting, assuming $\mathsf{P} \neq \mathsf{NP}$, due to k-means being APX-hard (Cohen-Addad and Karthik, 2019; Awasthi et al., 2015). That a no-regret algorithm with runtime that is exponential in k, d, shows that the main obstacle in devising these algorithms is computational rather than information-theoretic.

We shown that FTL with a k-means oracle fails to guarantee sublinear regret in the worst case, but performs very well on natural datasets. This merits further study, specifically: what stability constraints on the data stream, e.g., well-enough separated clusters, i.i.d. data from a mixture model, allow FTL to obtain logarithmic regret? Can a modification of FTPL, like the one by Dudik et al. (2017), work? The difficulty of not knowing a (small enough) suitable expert set in advance makes their results inapplicable in this setting.

We presented an algorithm that obtains $\tilde{O}(\sqrt{T})$ approximate regret with a runtime of $O(T(k^2\varepsilon^{-2}d^{1/2}\log(T))^{k(d+10)})$ using an adaptive variant of MWUA and an incremental coreset construction, which provides a theoretical upper bound for the approximate regret minimization problem. The next steps in this line of research will involve studying lower bounds for this approximate regret minimization problem and providing simpler algorithms such as some regularized form of FTL. Another extension may reduce the dependency on the dimension by performing dimensionality reduction for the data that preserves k-means cost of clusters, such as the *Johnson-Lindenstrauss tranformation* (JL); applying JL to our algorithm is not straightforward, as JL preserves only the loss of cluster centroids, but not arbitrary cluster centres, which are used in our algorithm. The need for additive approximation in terms of regret bounds make applying coreset and JL which come with multiplicative approximation guarantees makes this significantly challenging.

7 Acknowledgements

Varun Kanade was supported in part by the Alan Turing Institute under the EPSRC grant EP/N510129/1. This work was done in part while Guy Rom was a student at the University of Oxford.

References

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 61–72, 2017. doi: 10.1109/FOCS.2017.15. URL `https://doi.org/10.1109/FOCS.2017.15`.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The mul-

tiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. *arXiv preprint arXiv:1502.03316*, 2015.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 30–39, 2003. doi: 10.1145/780542.780548. URL `http://doi.acm.org/10.1145/780542.780548`.

Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

Anna Choromanska and Claire Monteleoni. Online clustering with experts. In *Artificial Intelligence and Statistics*, pages 227–235, 2012.

Vincent Cohen-Addad and CS Karthik. Inapproximability of clustering in lp metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539. IEEE, 2019.

Vincent Cohen-Addad, Arnaud De Mesmay, Eva Rotenberg, and Alan Roytman. The bane of low-dimensionality clustering. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 441–456. SIAM, 2018.

Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 42:1–42:14, 2019. doi: 10.4230/LIPIcs.ICALP.2019.42. URL `https://doi.org/10.4230/LIPIcs.ICALP.2019.42`.

Sanjoy Dasgupta. Course notes, cse 291: Topics in unsupervised learning. lecture 6: Clustering in an online/streaming setting, 2008. URL `http://cseweb.ucsd.edu/~dasgupta/291-unsup/lec6.pdf`.

Miroslav Dudik, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 528–539. IEEE, 2017.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 653–662. IEEE, 1998.

Le Li, Benjamin Guedj, and Sébastien Loustau. A quasi-Bayesian perspective to online clustering. *Electronic Journal of Statistics*, 12(2):3071–3113, 2018. URL `https://projecteuclid.org/euclid.ejs/1537430425`.

Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 81–89. SIAM, 2016.

Adam Meyerson. Online facility location. In *Proceedings 2001 IEEE International Conference on Cluster Computing*, pages 426–431. IEEE, 2001.

Michal Moshkovitz. Unexpected effects of online k-means clustering. *arXiv preprint arXiv:1908.06818*, 2019.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.