Online k-means Clustering

Vincent Cohen-Addad
CNRS
vcohenad@gmail.com

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

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

Guy Rom
University of Oxford
guy.rom@cs.ox.ac.uk

August 25, 2019

Abstract

We study the problem of online clustering where a clustering algorithm has to assign a new point that arrives to one of k clusters. 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 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 ($\mathcal{C}$) in hindsight. We show that provided the data lies in a bounded region, an implementation of the Multiplicative Weights Update Algorithm (MWUA) using a discretized grid achieves a regret bound of $\tilde{O}(\sqrt{T})$ 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 centres at each round) implies an offline efficient algorithm for the k-means problem. In light of this hardness, we consider the slightly weaker requirement of comparing regret with respect to $(1+\epsilon)\mathcal{C}$ and present a no-regret algorithm with runtime $O\left(T(\text{poly}(\log(T), k, d, 1/\epsilon)^{k(d+O(1))}\right)$. Our algorithm is based on maintaining an incremental coreset and an adaptive variant of the MWUA. We show that naïve online algorithms, such as *Follow The Leader*, fail to produce sublinear regret in the worst case. We also report preliminary experiments with synthetic and real-world data.

1 Introduction

Clustering algorithms are one of the main tools of unsupervised learning and often form a key part of a data analysis pipeline. Unlabeled data is ubiquitous in the real world and discovering structure in such data is essential in many online applications. The focus of this work is on the *online* setting where data elements 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. While several objective functions for clustering exist, in our work we will focus on the k-means objective. Most of our results can be easily generalized to most centre-based objectives.

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

by a function that grows sublinearly with the number of data elements. We consider *regret analysis* in this paper.

More precisely, in the case of online k-means clustering, the online algorithm at time t maintains a set of k candidate cluster centres, $C_t = \{c_{t,1}, \ldots, c_{t,k}\}$ 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.

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

1.1 Our Contributions

We consider the setting where the data x_t all lie in the unit box in $[0,1]^d \subset \mathbb{R}^d$. We summarise our contributions below.

- A multiplicative weight-update algorithm (MWUA) over sets of size k of candidate centres drawn from a uniform grid over $[0,1]^d$ achieves expected regret $\tilde{O}(\sqrt{T})$; here the $\tilde{O}$ notation hides factors that are poly-logarithmic in T and polynomial in k and d. The algorithm and its analysis is along standard lines and the algorithm is computationally inefficient. Nevertheless, this algorithm establishes that information-theoretically achieving $\tilde{O}(\sqrt{T})$ regret is possible.
- We provide 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, $1/\epsilon$, $f(t,k,d)$. In particular, for an offline instance of k-means with n point in a bounded region with $\mathcal{C} \geq 1/\text{poly}(n)$, an online algorithm with polynomial run time would yield a fully poly-time approximation scheme. We note that there exist hard instances for k-means for which $\mathcal{C} \geq 1/\text{poly}(n)$ and that it is known that k-means is APX-hard [1]. Furthermore, all known (approximation) FPT algorithms for k-means are exponential in at least one of the two parameters k and d. This suggests that we need to relax performance requirements for efficient algorithms.
- We consider a weaker notion of regret called $(1+\epsilon)$-regret. Let $\mathcal{C}$ denote the loss of the best solution in hindsight and L_T the cumulative loss of the algorithm; in the definition of regret, instead of $L_T - \mathcal{C}$, we consider $L_T - (1+\epsilon)\mathcal{C}$. With this notion of regret, we provide an algorithm that achieves $(1+\epsilon)$-regret $\tilde{O}(\sqrt{T})$ and runs in time $O\left(T(d^{1/2}k^2\epsilon^{-2}\log(T))^{k(d+O(1))}\right)$.
- Finally, we consider online algorithms which have oracle access to k-means solver. For instance, this allows the us to implement *follow the leader*. We show that there exists a sequence of examples for which follow the leader has *linear* regret. We show that this construction indeed results in linear regret in simulations. We observe that FTL (using k-means++ as a proxy for oracle) works rather well on real-world data.

1.2 Related Work

Clustering has been studied from various perspectives, e.g. combinatorial optimization, probabilistic modelling, and there are several widely used algorithms such as Lloyd's local search algorithm with k-means++ seeding, spectral methods, the EM algorithm, etc. We will restrict discussion mainly to the clustering as combinatorial optimization viewpoint. The k-means objective is one of the family of centre-based objectives which uses the squared distance to the centre as a measure of variance. Framed as an optimization problem, the problem is NP-complete. As a result, theoretical work has focused on approximation and FPT algorithms.

A related model to the online framework is the streaming model. As in the online model, the data is received one at a time. The focus in the streaming framework 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.

Online learning with experts and related problems have been widely studied, see e.g. Cesa-Bianchi and Lugosi [2] and references therein. The *Multiplicative Weight Update Algorithm (MWUA)* is a widely studied algorithm that may be used for regret minimization in the prediction with expert advice setting. It maintains a distribution over experts that changes as new data points arrive, which is used to sample an advice of some expert to be used as the next prediction, resulting in low regret. For a thorough survey see Arora et al. [3]. FTL is a simple online algorithm that always predicts the best solution for the data witnessed so far. It is known to admit low regret for some problems, namely, strongly convex objectives [4]. Variants of FTL that optimize a regularized objective have been successfully utilized for a wider range of settings.

The closest related work to ours is the work of Dasgupta [5]. He defines the evaluation framework we use here, and presents a naïve greedy algorithm to adress it, with no analysis. In addition, he combines algorithms by Charikar et al. [6] and Beygelzimer et al. [7] that together maintain a set of constant approximations of the k-centre objective at any time, for a range of values for k.

Choromanska and Monteleoni [8] study the online clustering problem in the presence of experts. The experts are batch clustering algorithms that output the centre closest to the next point at each step (hence provide implicit information on the next point in the stream). Using experts that have approximation guarantees for the batches they obtain an approximate regret guarantee of $\log(T)$ for the stream. Our setting differs in that we must commit to the next cluster centres strictly before the next point in the stream is observed, or any implicit information about it.

Li et al. [9] provide a generalized Bayesian adaptive online clustering algorithm (built upon the PAC-Bayes theory). They describe a Gibbs Sampling procedure of $O(k)$ centres and prove it has a minimax sublinear regret. They present a reversible jump MCMC process to sample these centres with no theoretical mixing time analysis.

Moshkovitz [10] studies a similar problem that considers only data points as candidate cluster centres, and the offline solution is defined similarly (an ℓ_2 analog of the k-medoids problem). Furthermore, the algorithm starts with an empty set of cluster centres and must output an incremental solution– cluster centers are only added to the set, and this is done in a streaming fashion. The loss is measured in hindsight, using the final set of cluster centres. They provide tight bounds for both adversarial and randomly ordered streams, and show that knowing the length of the stream reduces the amount of centres required to obtain a constant approximation by a logarithmic factor.

Liberty et al. [11] handle a different definition of *online*– the data points are labeled in an online fashion but the loss is calculated according to the centroids of the final clusters. They allow $O(k \cdot f(n, \gamma))$ clusters where γ is the aspect ratio of the data (the ratio between the diameter of the set and the closest distance between two points), and guarantee a constant competitive ratio when compared to the best k clusters in hindsight.

Meyerson [12] studies *online facility location*, where one maintains a set of facilities at each time step, and suffers a loss which is the distance of the new point to the closest facility. Once a facility is located, it cannot be moved, and placing a new one incurs a loss. Meyerson presents an algorithm that has a constant competitive ratio on randomly ordered streams. For adversarial order he presents an algorithm with $\log(T)$ competitive ratio and provides a lower bound showing no algorithm can have a constant competitive ratio.

2 Basic Results

2.1 Preliminaries and Notation

To precisely define the online clustering problem we will consider points in the unit box $[0,1]^d \subseteq \mathbb{R}^d$. The point arriving at time t will be 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 of k candidate centres using only $X_{1:t-1}$. We refer to the set of all the candidate centres an algorithm is considering as *sites*. 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 denoted by $L_t = \sum_{\tau=1}^{t} \ell(C_\tau, x_\tau)$. The loss of the best k-means solution in *hindsight* after T steps is denoted by $\mathcal{C}$. The regret is defined as $L_T - \mathcal{C}$. Several of the algorithms we consider will pick cluster centres from a constrained set; we sometimes refer to any set of k sites from such a constrained set as an *expert*. We define the $(1+\epsilon)$-approximate regret as $L_T - (1+\epsilon)\mathcal{C}$.

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

We denote the best k-means solution, i.e. best k cluster centres, for $X_{1:t-1}$ by C_t^*, hence C_{T+1}^* is the best k-means solution in hindsight. In our setting, the *Follow-The-Leader* (FTL) algorithm simply picks C_t^* at time t. We use $\tilde{O}$ to suppress factors that are poly-logarithmic in T and polynomial in k and d.

2.2 MWUA with Grid Discretization

While the multiplicative weight update algorithm (MWUA) is very widely applicable, there are a couple of difficulties when it comes to applying it to our problem. In order to obtain a finite set of experts, we consider *sites* obtained by a δ-grid of $[0,1]^d$. In order to obtain regret bounds that are $\tilde{O}(T^{1-\alpha})$ for $0 < \alpha \leq 1/2$ (typically $\alpha = 1/2$), we need to choose $\delta = T^{-\alpha/2}$; this means that the number of experts is exponentially large in k and d and polynomially large in T. However, since the regret of MWUA only has logarithmic dependence on the number of experts, this mainly incurs a computational, rather than statistical cost.[1] In Section 3, we develop a more *data-adaptive* version of the MWUA. This 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 for $(1+\epsilon)$-regret. However, the results in Section 2.3 show that this is (under computational conjectures) unavoidable.

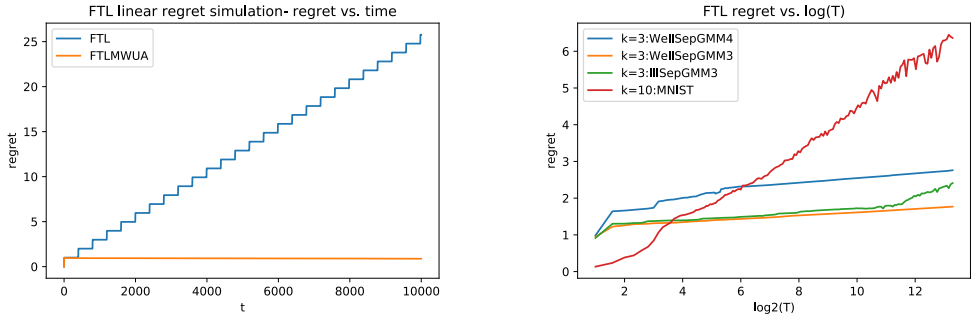
Theorem 2.1. *Let $S = \{i\delta \mid 0 \leq i \leq \delta^{-1}\}^d \subset \mathbb{R}^d$ be the set of* sites *and let $\mathcal{E} = \{C \subset S \mid |C| = k\}$ be the set of experts (k-centres chosen out of the sites). Then, for any $0 < \alpha \leq 1/2$, with $\delta = T^{-\alpha/2}$, the MWUA with the expert set $\mathcal{E}$ achieves regret $\tilde{O}(T^{1-\alpha})$; the per round running time is $O(T^{\alpha k d/2})$.*

2.3 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 under complexity-theoretic assumptions. In particular, a consequence of Theorem 2.2 is that for instances of k-means with data lying in a bounded region and $\mathcal{C} \geq 1/\text{poly}(n)$, a per-round polynomial time online algorithm would imply a fully polynomial-time approximation scheme. Recall that k-means is APX-hard and current best known algorithms are exponential in at least one of the two parameters k and d.

Theorem 2.2. *Suppose there is an online k-means clustering algorithm $\mathcal{A}$ that achieves regret $\tilde{O}(T^{1-\alpha})$ and runs at time t in time $f(t,k,d)$. 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 $\mathcal{C} + \epsilon$ with constant probability and runs in time polynomial in $n, k, d, \frac{1}{\epsilon}, f(n,k,d)$.*

[1] There appears to be some statistical cost in that we are only able to prove bounds on expected regret.



(a) FTL Linear Regret, MWUA-FTL Sublinear Regret (b) FTL Regret–MNIST and Gaussian Mixure Models

2.4 Follow The Leader– Linear Regret Worst Case

The lower bound presented here does not imply anything on the regret guarantees of *Follow-The-Leader (FTL)* that has orcale access to the best offline solution C_t^* at each step. This section will show that FTL incurs linear regret in the worst case,

Theorem 2.3. *FTL obtains $\Omega(T)$ regret in the worst case, for any fixed $k \geq 2$ and any dimension.*

Figure 1a shows the regret of FTL at any point in time if the stream halted at that point, which we refer to as the *regret halted at t*, for a stream that was generated according to the scheme describe above with $\delta = 0.1$, along with a MWUA over the set of leaders $\{C_t^*\}_{t=1}^T$, where the MWUA weight for any leader is calculated according to their historical loss, regardless of the time t when it was introduced to the expert set, i.e. $\forall t' > t : w_{C_t^*}^{(t')} = \prod_{\tau=1}^{t'-1}(1 - \eta\ell(C_t^*, x_\tau))$. 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 several iterations, and negligent excess loss in the rest of the iterations. This demonstrates that the counter example provided in A is viable and numerically stable without special care. The MWUA-FTL presents low asymptotic regret in this case, which is clearly sub-linear and possibly logarithmic in T. The best intermediary k-means solution at each step was calculated analytically.

2.5 Follow The Leader– Sublinear Regret on Natural Datasets

The previous section showed that there are worst case instances that make FTL perform badly. This section will present experimental results, on synthetic and real data sets, that suggest that FTL performs very well on natural data sets.

Figure 1b shows the regret halted at t of FTL vs. $\log(t)$ for four different data sets, all of size 10000. The first is a random sample from MNIST, labelled MNIST, treated as a $d = 784$ dimensional vector, 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. The GMM4 case was run with $k = 3$, where the 4 Gaussians are well separated (means distance is larger than 3 times the standard deviation) hence labelled WellSepGMM4. The two GMM3 data sets are labelled WellSepGMM3 for the well separated case, and IllSepGMM3 for a case where the Gaussians are ill separated (means distance is 0.7 fraction of the standard deviation). In all cases the best intermediary k-means solution was calculated using k-means++ with 300 iterations of local search, hence it is an approximation. The figure demonstrates a linear dependency between the regret and $\log(t)$ in these cases. All the standard deviations are set to 0.1.

Algorithm 1: ε-Regret Minimization

Input: $\varepsilon, x_1, \ldots, x_T$
$t \leftarrow 1$ $\varepsilon_{\mathsf{c}}, \varepsilon_{\mathsf{hrd}} \leftarrow \varepsilon^*(\varepsilon, k, d, T)$;
Initialize $\mathcal{H}_0$ and MTMW;
for $t = 1$ **to** 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

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
end

3 Approximate Regret Minimization

We present an algorithm that aims to minimize approximate regret for the online clustering problem. The algorithm uses three main components

1. The *Incremental $\varepsilon_{\mathsf{c}}-$Coreset* algorithm of 3.1 that maintains a monotone sequence of sets $\{\mathcal{Q}_t\}_{t=0}^T$ such that $\mathcal{Q}_t$ contains a weighted $\varepsilon_{\mathsf{c}}-$coreset for $X_{1:t}$.
2. A *Hierarchical Region Decomposition* of 3.2 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.
3. MWUA for tree structured expert sets, 3.4, referred to as *Mass Tree MWUA* (MTMW), that is given $\mathcal{H}_t$ at every step, and outputs a choice of k centers.

We present our main theorem and provide proof later on.

Theorem 3.1. *Algorithm 1 has a regret of*

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

and runtime of

$$T \cdot O(\sqrt{d}k^2 \log(T)\varepsilon^{-2})^{k(d+O(1))}$$

We now continue to describe the different components, and then combine them.

3.1 Incremental Coreset

The *Incremental Coreset* algorithm presented in this subsection receives an unweighted stream of points $X_{1:T}$ one point in each time step and maintains a monotone sequence of sets $\{\mathcal{Q}_t\}_{t=0}^T$ such that $\mathcal{Q}_t$ contains a weighted $\varepsilon_{\mathsf{c}}-$coreset for $X_{1:t}$, for some given parameter $\varepsilon_{\mathsf{c}} > 0$. Formally, we have the following lemma, whose proof is provided in B.

Lemma 3.2. *For any time step t, the algorithm described in 3.1.2 outputs a set of points $\mathcal{Q}_t$ such that it contains a $(1+\varepsilon_{\mathsf{c}})-$coreset for $X_{1:t}$, which we denote $\chi(X_{1:t})$, and has size at most $O(k^2\varepsilon_{\mathsf{c}}^{-4}\log^4 T)$. Moreover, we have that $\mathcal{Q}_t \subseteq \mathcal{Q}_{t+1}$.*

3.1.1 Maintaining an $O(1)$-Approximation for k-means

The first part of our algorithm is to maintain the sets of points $S_1^{(t)}, \dots S_s^{(t)}$, for any time step t, representing possible sets of cluster centers where $s = O(\log T)$. We require that at any time, at least one set, denoted $\mathcal{S}^{(t)}$, induces a bicriteria $(O(k \log^2 T), O(1))$-approximation to the k-means problem, namely, $\mathcal{S}^{(t)}$ contains at most $O(k \log^2 T)$ centers and its loss is at most some constant times the loss of the best k-means solution using at most k centers. Moreover, for each $i \in [s]$, the sequence $\{S_i^{(t)}\}_{t=1}^T$ is an *incremental clustering*: First, it must be a monotone sequence, i.e. for any time step t, $S_i^{(t-1)} \subseteq S_i^{(t)}$. Furthermore, if a data point x of the data stream is assigned to a center point $c \in S_i^{(t)}$ at time t, it remains assigned to c in any $S_i^{(\tau)}$ for $t \leq \tau \leq T$, i.e. until the end of the algorithm.

Each set which contains more than $O(k \log^2 T)$ is said to be *inactive* and the algorithm stops adding centers to it. The remaining sets are said to be active.

To achieve this, we will use the algorithm of Charikar, O'callaghan and Panigrahy [13], which we call $\mathcal{A}_{cop}$, and whose performance guarantees are summarized by the following proposition, that follows immediately from Charikar et al. [13].

Theorem 3.3 (Combination of Lemma 1 and Corollary 1 in [13])**.** *With probability at least 1/2, at any time t, one set maintained by $\mathcal{A}_{cop}$ is an $O(1)$-approximation to the k-means problem which uses at most $O(k \log T)$ centers.*

3.1.2 Maintaining a Coreset for k-means

Our algorithm maintains a coreset Q_i based on each solution $S_i^{(t)}$ maintained by $\mathcal{A}_{cop}$. It makes use of coresets through the coreset construction introduced by Chen [14] whose properties are summarized in the following theorem.

Theorem 3.4 (Thm. 5.5/3.6 in Chen [14])**.** *Given a set P of T points in a metric space and parameters $1 > \varepsilon_{\mathsf{c}} > 0$ and $\lambda > 0$, one can compute a weighted set Q such that $|Q| = O(k\varepsilon_{\mathsf{c}}^{-2} \log T(k \log T + \log(1/\lambda))$ and Q is a $(1+\varepsilon_{\mathsf{c}})$-coreset of P for k-means clustering, with probability $1 - \lambda$.*

We now review the coreset construction of Chen. Given a bicriteria (α, β)-approximation $S_0^{(t)}$ to the k-means problem, Chen's algorithm works as follows. For each center $c \in S_0^{(t)}$, consider the points in P whose closest center in $S_0^{(t)}$ is c and proceed as follows. For each i, we define the i^{th} ring of c to be the set of points of cluster c that are at distance $[2^i, 2^{i+1})$ to c. The coreset construction simply samples $\zeta \geq \beta\varepsilon_{\mathsf{c}}^{-4} k \log T$ points among the points whose distance to c is in the range $[2^i, 2^{i+1})$ (if the number of such points is below ζ simply take the whole set). This ensures that the total number of points in the coreset is $\alpha k \zeta \log \Delta$, where Δ is the maximum to minimum distance ratio (which can be assume to be polynomial in n without loss of generality).

Our algorithm stores at each time t a set of points $\mathcal{Q}_t$ of small size that contains a $(1 + \varepsilon_{\mathsf{c}})$-coreset. Moreover, we have that the sets $\mathcal{Q}_t$ are incremental: $\mathcal{Q}_{t-1} \subseteq \mathcal{Q}_t$.

To do so, our algorithm uses the bicriteria approximation algorithm $\mathcal{A}_{cop}$ of Section 3.1.1 as follows. For each solution stored by $\mathcal{A}_{cop}$, the algorithm uses it to compute a coreset via a refinement of Chen's construction. Consider first applying Chen's construction to each solution $S_i^{(t)}$ maintained by $\mathcal{A}_{cop}$. Since $\mathcal{A}_{cop}$ is incremental, whatever decisions we have made until time t, center open and point assignment, will remain unchanged until the end. Thus, applying Chen's construction seems possible. The only problem is that for a given set $S_i^{(t)}$, a given center $c \in S_i^{(t)}$ and a given ring of c, we don't know in advance how many points are going to end up in the ring and so, what should be sampling rate so as to sample $\Theta(\zeta)$ elements uniformly.

To circumvent this issue, our algorithm proceeds as follows. For each set $S_i^{(t)}$, for each center $c \in S_i^{(t)}$, for each j, the algorithm maintains $\log T$ samples: one for each 2^i which

represents a "guess" on the number of points in the j^{th} ring that will eventually arrive. More precisely, for a given time t, let p_t be the newly inserted point. The algorithm then considers each solution $S_i^{(t)}$, and the center $c \in S_i^{(t)}$ that is the closest to p_t. If p_t belongs to the j^{th} ring, then p_t is added to the set $A(S_i^{(t)}, c, j, u)$ with probability $\zeta/2^u$ for each $u \in [\log T]$ if $|A(S_i^{(t)}, c, j, u)| \leq 2\zeta$. Let $A(S_i^{(t)})$ denote the union over all center $c \in S_i^{(t)}$, integers $j, u \in [\log T]$ of $A(S_i^{(t)}, c, j, u)$. Let $\mathcal{Q}_t = \bigcup_i A(S_i^{(t)})$. For more details, the reader is referred to the proof for Lemma (3.2) in B.

3.2 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. There is a node in $T_{\mathcal{H}}$ for each region of each $\mathcal{R}_\tau$. There is an edge from the node representing region R to the node representing region R' if $R \subseteq R'$ and there exists a τ such that $R \in \mathcal{R}_\tau$ and $R' \in \mathcal{R}_{\tau+1}$. We slightly abuse notation and refer to the node corresponding to region R by R. The bottom-level region decomposition is the region decomposition 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 $\{\tilde{S}_\tau\}_{\tau=1}^t$ the consists of the centroids of the representative regions of S in $\mathcal{H}_t$.

3.3 Adaptive Grid Hierarchical Region Decomposition

Given a sequence of points in $\mathbb{R}^d$, we describe an algorithm that maintains a hierarchical region decomposition with $d-$dimensional hypercube regions as follows. Let $\varepsilon_{\mathsf{hrd}} > 0$ be a parameter s.t. $\frac{\varepsilon_{\mathsf{hrd}}}{2T^3\sqrt{d}}$ is a power of 2. We require this in order to define an implicit grid with side length $\frac{\varepsilon_{\mathsf{hrd}}}{2T^3\sqrt{d}}$, i.e. diameter $\delta_T = \frac{\varepsilon_{\mathsf{hrd}}}{2T^3}$, such that it can be constructed from a single region containing the entire space by repeated halving in all dimensions. Denote $\delta_t = \frac{\varepsilon_{\mathsf{hrd}}}{2t^3}$. We refer to this implicit grid, along with the region tree structure that corresponds to the this halving process as the *Full Grid* and the *Full Grid Tree*. Consider a step t, $R \in \mathcal{R}_t$ and $x \in [0,1]^d$. Denote the diameter of R as ΔR, and $r = \min_{p \in R} \|p - x\|_2$ the distance between R and x. Notice that if $x \in R$ then $r = 0$.

We define the *refinement criteria induced by x at time t* as $q(R)$, which takes the value true if and only if the diameter of R is smaller or equal $\max(\varepsilon_{\mathsf{hrd}} \cdot r/2, \delta_t)$. At a given time t, a new point x_t is received and the hierarchical region decomposition obtained at the end of time $t-1$, $\mathcal{H}_{t-1}$, is refined using the following procedure, which guarantees that all the new regions satisfy the refinement criteria induced by all the points $X_{1:t}$ at the corresponding insertion times. The pseudocode is given in 2.

We now turn to proving *Structural Properties* of the hierarchical region decomposition $\mathcal{H}_t$ that the algorithm maintains. The proof of the following lemma follows immediately from the definition.

Lemma 3.5. *Consider the hierarchical region decomposition $\{\mathcal{R}_1, \ldots, \mathcal{R}_t\}$ produced by the algorithm at any time t. Consider a region $R \in \mathcal{R}_{t-1}$ and let ΔR be the diameter of region R, then the following holds. Either region R belongs to $\mathcal{R}_t$ or each child region of R in $\mathcal{R}_t$ has diameter at most $\frac{1}{2}\Delta R$.*

Corollary 3.6. *Consider $\{R_t \in \mathcal{R}_t\}_{t=1}^T$ such that $\forall t: \quad R_{t+1} \subseteq R_t$, a sequence of nested regions of length T. We say that such a sequence cannot be refined more than Λ times, i.e. $|\{t|R_{t+1} \neq R_t\}| \leq \Lambda$. Lemma (3.5) along with the fact that the algorithm does not refine regions with diameter smaller than δ_T give us that*

$$\Lambda \leq -\log\left(\delta_T/\sqrt{d}\right) = -\log\left(\frac{\varepsilon_{\mathsf{hrd}}}{2T^3\sqrt{d}}\right)$$

The proof for the following Lemma is provided in B.

Lemma 3.7. *For any stream of length N, using the above algorithm, we have that the total number of regions that are added at step t is at most $(9\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d \log(T^3)$ hence the total amount of regions in $\mathcal{R}_N$ is $N(9\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d \log(T^3)$.*

Corollary 3.8. *Let $R \in \mathcal{R}_t$ be any region at step t and $S = \{R' \in \mathcal{R}_{t+1} | R' \subseteq R\}$ be the set of regions that refine R in the next time step, then we have that the log max branch β is*

$$\beta = \log \max_{t,R} |S| \leq \log\left(\left(\frac{9\sqrt{d}}{\varepsilon_{\mathsf{hrd}}}\right)^d \log(T^3)\right)$$

Due to Lemma (3.7). Furthermore for sufficiently large T we have that $\beta \leq d \cdot \Lambda$

We will now present a few properties relating to the approximation of the k-means problem.

Lemma 3.9. *Let $\varepsilon_{\mathsf{hrd}} > 0$. Consider an online instance of the weighted k-means problem where a new point and its weight are inserted at each time step. Let x_t be the weighted point that arrives at time t, such that its weight is bounded by t. Consider the hierarchical region decomposition with parameter $\varepsilon_{\mathsf{hrd}}$ produced by the algorithm $\mathcal{H}_t = \{\mathcal{R}_1, \ldots, \mathcal{R}_t\}$, for some time step t.*

Consider two multisets of k centers $S = \{c_1, \ldots, c_k\}, S' = \{c'_1, \ldots, c'_k\}$ such that for all $i \in \{1 \ldots k\}$, c_i and c'_i are contained in the same region of the Region Decomposition of step t. Then, the following holds.

$$\forall 1 \leq \tau < t, \quad \ell(S', x_\tau) \leq (1 + \varepsilon_{\mathsf{hrd}})\ell(S, x_\tau) + \varepsilon_{\mathsf{hrd}}/\tau^5$$

We now extend this lemma to solutions for the k-means problem. Given a set of k centers $S = \{c_1, \ldots, c_k\}$ and a hierarchical region decomposition $\mathcal{H} = \{\mathcal{R}_1, \ldots, \mathcal{R}_t\}$, we associate a sequence $\{\tilde{S}_1, \ldots, \tilde{S}_t\}$ of *approximate centers for S induced by $\mathcal{H}$* by picking for each c_i the approximation of c_i induced by $\mathcal{H}$ at step t– the centroid of the region in $\mathcal{R}_t$ that contains c_i. Note that this is a multiset. The next lemma follows directly from applying Lemma (3.9) to these approximate centers, and summing over t.

Lemma 3.10. *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 a weighted stream $X_{1:t}$*

$$(1 - \varepsilon_{\mathsf{hrd}})\mathcal{C} - 2\varepsilon_{\mathsf{hrd}} \leq \sum_{t=1}^{T} \ell(\tilde{S}_{t+1}, x_t) \leq (1 + \varepsilon_{\mathsf{hrd}})\mathcal{C} + 2\varepsilon_{\mathsf{hrd}}$$

As Lemma (3.6) gives that that $\tilde{S}_{t+1} \neq \tilde{S}_t$ at most $k \cdot \Lambda$ times (each of the k regions may be refined Λ times), and the loss is bounded by d, then along with Lemma (3.10) we get the following corollary.

Corollary 3.11. *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 $X_{1:t}$*

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

3.4 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 [3], 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 simple modification to the proof of Theorem (2.1) of Arora et al. [3], 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 3.12 (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.

Theorem 3.13. *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 has a time complexity of $O(|\mathcal{T}_T|)$, the number of vertices of $\mathcal{T}_T$.

Corollary 3.14. *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*

3.5 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 (3.11). 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 (3.1).

Lemma 3.15. *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*

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

Consider the region tree structure $T_{\mathcal{H}_t}$ described in 3.2 that corresponds to the Hierarchical Region Decomposition $\mathcal{H}_t$ at step t defined in (3.15). 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. An important thing to note is that Lemma (3.15) 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 $\mathcal{C}$ 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 3.16. *Using the definitions from Lemmas (3.6), (3.8) we have that* $-\ln(\mathcal{M}(p^*)) \leq k^2 \Lambda \beta$

Defining $\frac{\varepsilon^2}{2ak^2 \log(T^3\sqrt{d})} \leq \varepsilon^* \leq \frac{\varepsilon^2}{ak^2 \log(T^3\sqrt{d})}$, s.t. $\varepsilon^* = 2^i$, where $a \geq 34^2$ is some constant, gives Theorem (3.1). The complete proof is provided in B.

4 Discussion

The online k-means clustering problem has been studied in a variety of settings. In the setting we use, we have shown that no efficient algorithm with sublinear regret exists, even in the Euclidean setting, under typical complexity theory conjunctures, due to k-means being APX-hard [1]. We have presented a no-regret algorithm with runtime that is exponential in k, d, showing that the main obstacle in devising these algorithms is computational rather than information-theoretic.

We have shown that FTL with orcale access to the best clustering so far fails to guarantee sublinear regret in the worst case, but performs very well on natural datasets. This opens a door for further study, specifically– what stability constraints on the data stream, such as well separation of clusters, or data points that are IID using well behaved distributions, allow FTL to obtain logarithmic regret?

We presented an algorithm that obtains $\tilde{O}\left(\sqrt{T}\right)$ approximate regret with a runtime of $O(T(k^2\varepsilon^{-2}d^{1/2}\log(T))^{k(d+O(1))})$ 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 FTL with a regularizer fit for purpose. 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.*

References

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

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

[3] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

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

[5] 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.

[6] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6): 1417–1440, 2004.

[7] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

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

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

A Proofs of Results from Section 2

A.1 MWUA

Let's look at the projection of a set of centres C to the set of *sites* S. This projection is a multiset defined as $\Pi(C,S) = \{\arg\min_{s\in S}(\|s-\mu\|_2) \mid \mu \in C\}$. Let $C' = \Pi(C,S)$ such that $C_i^{'}$ is the projected point corresponding to C_i. We define the projection infinity distance of C onto S as the maximum distance between these pairs, i.e. $\max_{i\in 1\ldots|C|} \|C_i^{'} - C_i\|_2$, and denote it as $\|C_i - S\|_\infty$.

The following lemma comes from the folklore.

Lemma A.1. *Let μ be the centroid (mean) of a set of points P, and $\hat{\mu}$ another point in space. Then $\ell(\hat{\mu}, P) = |P| \cdot \|\mu - \hat{\mu}\|_2^2 + \ell(\mu, P)$.*

Corollary A.2. *Let P be a set of points, $\boldsymbol{\mu}$ the set of k optimal centers, and $\hat{\boldsymbol{\mu}}$ k alternative centers such that $\|\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\|_\infty \leq \epsilon$ then $\ell(\hat{\boldsymbol{\mu}}, P) \leq |P| \cdot \epsilon^2 + \ell(\boldsymbol{\mu}, P)$*

We now turn to the main theorem

Theorem A.3. *Let $S = \{i\delta \mid 0 \le i \le \delta^{-1}\}^d \subset \mathbb{R}^d$ 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 $0 < \alpha \le 1/4$, with $\delta = T^{-\alpha}$, the MWUAwith the expert set $\mathcal{E}$ achieves regret $\tilde{O}(T^{1-2\alpha})$; the per round running time is $O(T^{\alpha kd})$.*

Proof. Following section (3.9) in Arora et al. [3]. The grid distance δ results with $n = \frac{1}{\delta^d}$ sites hence $N = \binom{n}{k}$ experts, which is $N = O(\delta^{-kd})$. Denote the regret with respect to the grid experts as the *grid-regret*. Running a single step in MWUArequires sampling and weight update, taking $O(kdN)$ time and the algorithm guarantees a grid-regret of at most $2\sqrt{ln(N)T} = 2\sqrt{kd\ln(\delta^{-1})T}$.

Let $C_g = \Pi(C^*_{T+1}, S)$ be the closest grid sites to C^*_{T+1}. Because $\|c^*_{T+1} - C_g\|_\infty \le \frac{\sqrt{d}}{2}\delta$, (A.2) gives $\ell(C_g, X_{1:T}) - \ell(C^*_{T+1}, X_{1:T}) \le \frac{d\delta^2}{4}T$. Hence the regret of the algorithm is at most $2\sqrt{kd\ln(\delta^{-1})T} + \frac{d\delta^2}{4}T$, so choosing $\delta = T^{-\alpha/2}$ for $\alpha \in (0, \frac{1}{2}]$ yields an algorithm with $\tilde{O}(T^{1-\alpha})$ regret and a per step time complexity of $O(T^{\alpha kd/2})$. □

A.2 Reduction

The following is a proof for Theorem (2.2)

Proof. We will reduce the offline problem with point set P to the online problem by generating a stream X of T uniformly sampled points from P and running $\mathcal{A}$ on X; $\mathcal{A}$ generates T intermediate cluster centers $\{C_t\}_{t=1}^T$, and we return the best one with respect to the entire set P.

Denote the best offline k-means solution as C^*– the optimal clustering for the stream C^*_{T+1} may not coincide with the optimal offline clustering C^*, but it must perform at least as good as C^* on $X_{1:T}$. Denote r the internal randomness of $\mathcal{A}$. The regret guarantee gives

$$\mathbb{E}_r[\mathbb{E}_X[\sum_{t=1}^{T} \ell(C_t, x_t) - \ell(C^*_{T+1}, X)]] \le T^\alpha$$

Using the linearity of expectation and noticing C^*_{T+1} doesn't depend on r.

$$\begin{aligned}\sum_{t=1}^{T} \mathbb{E}_r[\mathbb{E}_X[\ell(C_t, x_t)]] &\le T^\alpha + \mathbb{E}_X[\ell(C^*_{T+1}, X)] \\ &\le T^\alpha + \mathbb{E}_X[\ell(C^*, X)]\end{aligned}$$

Using $\forall C : \mathbb{E}_{x_t}[\ell(C, x_t)] = \frac{\ell(C,P)}{|P|}$

$$\sum_{t=1}^{T} \mathbb{E}_r[\mathbb{E}_X[\frac{\ell(C, P)}{|P|}]] \le T^\alpha + \frac{T\mathcal{C}}{|P|}$$

Define ϵ_t s.t. $\mathbb{E}_r[\mathbb{E}_X[\ell(C, P)]] = (1 + \epsilon_t)\mathcal{C}$. Because C^* is optimal, we know $\forall t : \epsilon_t \ge 0$.

$$\sum_{t=1}^{T} \frac{(1 + \epsilon_t)\mathcal{C}}{|P|} \le T^\alpha + \frac{T\mathcal{C}}{|P|}$$

Rearranging

$$\sum_{t=1}^{T} \epsilon_t \le \frac{|P|T^\alpha}{\mathcal{C}}$$

Denote $\epsilon^* = \min_t(\epsilon_t)$

$$T \cdot \epsilon^* \leq \frac{|P|T^\alpha}{\mathcal{C}}$$
$$\epsilon^* \leq \frac{|P|T^{\alpha-1}}{\mathcal{C}}$$

So provided $\mathcal{C}^{-1} = \text{poly}(|P|)$ one can choose $T = \text{poly}(|P|)$ s.t. ϵ^* is arbitrarily small. ϵ^* is a non-negative random variable, hence this condition suffices to produce an approximation algorithm with arbitrary ϵ for k-means. Awasthi et al. [1] shows that this is NP-hard, finishing the proof. □

A.3 FTL

The following is a proof for Theorem (2.3)

Proof. We will present an algorithm that generates a stream of points on a line, for $k = 2$ and any value of T, such that the regret FTL obtains for the stream can be bounded from below by $c \cdot T$ where c is some constant. Extending the result to arbitrary k can be done by contracting the bounding box where the algorithm generates points by a factor of $2k$, and adding data points outside of it in $k-2$ equally spaced locations, to force the creation of $k-2$ clusters, one for each location.

The stream will consist of points in 3 locations $-\delta, 0, (1-\delta)$ for $\delta < \frac{1}{4}$. We will call C_t a $(-\delta)$-clustering if it puts all the points at $(-\delta)$ in one cluster and the rest in the other cluster, and a $(1-\delta)$-clustering puts all the point at $(1-\delta)$ in one cluster and the rest in the other cluster. We will define a stream that has a $(1-\delta)$ optimal C^*_{T+1} clustering.

We will keep the amount of points in 0 and $(-\delta)$ equal up to a difference of 1 at any step by alternative between the two any time we put points in one of them. There exists $n^* = f(\delta) = O(\delta^{-2})$ such that if there is one point at $(1-\delta)$ and n^* points in each of $0, -\delta$ then the optimal clustering is the $(1-\delta)$-clustering and for n^*+1 points in each of $0, -\delta$ the optimal clustering is the $(-\delta)$-clustering.

Our algorithm will start with a point in $(1-\delta)$ making C_t a $(-\delta)$-clustering. The next points will be added as follows– as long as C_t is a $(1-\delta)$-clustering add a point to $(-\delta)$ or 0, balancing them; if C_t has just become a $(-\delta)$-clustering, the next point will be $(1-\delta)$, inflicting an loss for FTL which depends only on δ hence it is $O(1)$, and making C_{t+1} a $(1-\delta)$-clustering again. Halt when we have T points.

When the algorithm halts a $(1-\delta)$-clustering is either the optimal clustering or is at most $O(1)$ below the optimal clustering, so we will say that C^*_{T+1} is a $(1-\delta)$-clustering without loss of generality. FTL will jump from $(-\delta)$-clustering to $(1-\delta)$-clustering $O(T/n^*)$ times, and suffer a loss of at least the loss C^*_{T+1} incurs at that step (because we balance $(-\delta)$ and 0, FTLmoves the lower cluster away from the middle hence suffer a slightly larger loss than C^*_{T+1} at the next stage). This means that the regret is larger than $O(T/n^*)$ which is bounded from below by a linear function in T for a fixed δ. □

B Proofs of Results from Section 3

B.1 Incremental Coreset

The following is a proof for Lemma (3.2)

Proof. Note that by the definition of the algorithm, each set $S_i^{(t)} \in \{S_1^{(t)}, \ldots, S_s^{(t)}\}$ contains at most $O(k \log^2 T)$ centers. It follows that, by the definition of the algorithm, any $S_i^{(t)} \in$

$\{S_1^{(t)}, \ldots, S_s^{(t)}\}$ is such that

$$|A(S_i^{(t)})| = \sum_{c \in S_i^{(t)}, j \in [\log T], u \in [\log T]} |A(\mathcal{S}^{(t)}, c, j, u)| \leq O(k\zeta \log^3 T)$$

The overall bound follows from the fact that $s = O(\log T)$. Moreover, the fact that at any time t, we have that $A(S_i^{(t)}) \subseteq A(S_i^{(t')})$ for $t' \geq t$ follows from Theorem (3.3) and the definition of the algorithm.

We then argue that $\mathcal{Q}_t$ contains a $(1 + \varepsilon_{\mathsf{c}})$-coreset. Recall that there exists a set $\mathcal{S}^{(t)} \in \{S_1^{(t)}, \ldots, S_s^{(t)}\}$ that induces a bicriteria $(O(\log^2 T), O(1))$-approximation to the k-meansproblem. Thus, for a given center $c \in \mathcal{S}^{(t)}$, for any j, the j^{th} ring of c is sampled appropriately (up to a factor 2), for the value u which is such that $2^{u-1} \leq \log T(c, j, t) < 2^u$ where $n(c, j, t)$ is the total number of points in the j^{th} ring of c (in solution $\mathcal{S}^{(t)}$) at time t. Thus, combining this observation with Theorem (3.4), we have that $A(\mathcal{S}^{(t)}) \subseteq \mathcal{Q}_t$ indeed contains a set of points that is a $(1 + \varepsilon_{\mathsf{c}})$-coreset. □

B.2 Hierarchical Region Decomposition

The following is a proof for Lemma (3.7)

Proof. At any time step t, each point x_t corresponds to a refinement criteria at time t over regions in the *Full Grid Tree*, namely, $q(\cdot)$, s.t. there exists a frontier (i.e. the leaves of a subtree of the Full Grid Tree) that separates the vertices from above, which have $q(R) =$ False, and the vertices below have $q(R) =$ True. This frontier can be thought of as the minimal refinement requirement along each path in the Full Grid Tree that corresponds to adding x_t. In order to satisfy the requirements of all the points in the stream and have that all the regions in $\mathcal{R}_t$ have $q(R) =$ True (for the corresponding time steps $t' \leq t$) the algorithm iterates the existing frontier in the Full Grid Tree that corresponds to the current region decomposition and extends it to match the requirements due to $X_{1:t-1}$ together with the new minimum requirements introduced by the new point. Hence the algorithm removes a subset of regions from the $\mathcal{R}_{t-1}$ and replaces them with a subset of the frontier that corresponds to x_t. We now turn to prove that the frontier in the Full Grid Tree of any point in space is of size $(9\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d \log(T^3)$, proving the lemma.

Let x denote an arbitrary point in space whose frontier size we are trying to bound. The frontier is made up of an area in space close to x that contains only regions of the minimum diameter δ_T, where we use T instead of t as it lower bounds for the diameter. When the distance from x is larger than $r_{hrd} = 2\delta_T/\varepsilon_{\mathsf{hrd}}$ the dominant term in the refinement criteria is $\varepsilon_{\mathsf{hrd}} \cdot r/2$, hence this is a hypersphere of radius r_{hrd} centered around x.

Outside of the r_{hrd} sphere we will have thick shells with inner radius $r_{hrd} \cdot 2^i$ and outer radius of $r_{hrd} \cdot 2^{i+1}$ where i is some nonnegative integer, where the refinement criteria requires that the maximum diameter will be $\delta_T \cdot 2^i$.

Consider a bounding box for the shell that corresponds to $i \geq 0$. The sides of such a hypercube are of length $2 \cdot r_{hrd} \cdot 2^{i+1}$. In order to partition this hypercube to regions of diameter $\delta_T \cdot 2^i$ we require $(8\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d$ regions. If we iterate the shells from the outermost shell inward, we can assume that a sphere twice as large as shell i was already partitioned to regions with half the resolution of shell i before iterating shell i.

Now we can account for the fact that the spheres, of radius r, are not aligned with the regions of the Full Grid Tree for the corresponding resolution/depth. Let us extend the bounding box of shell i such that it is aligned with the full grid in the resolution of the next shell $(i + 1)$. This means that the edge length of the box is enlarged by at most two units of edge length $\delta_T \cdot 2^i/\sqrt{d}$ from each side, resulting in $(4 + 8\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d \leq (9\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d$ regions, for sufficiently small $\varepsilon_{\mathsf{hrd}}$.

There are at most $\log(1/r_{hrd}) = \log(2T^3)$ shells, and noticing that the innermost shell accounts for the core, this gives the bound. □

The following is a proof for Lemma (3.9)

Proof. Fix τ and focus on c_i the cluster center in S closest to x_τ, and its corresponding c'_i.

consider the region R containing c_i. If R also contains x_τ then, since x_τ has been inserted to the stream and so R has diameter of at most δ_t. Because c'_i is also contained in this region the weighted loss is at most $\tau \cdot (\frac{\varepsilon_{\mathsf{hrd}}}{2\tau^3})^2$.

Thus, we turn to the case where R does not contain x_τ, hence $||x_\tau - c_i|| > 0$. Denote $r = ||x_\tau - c_i||$. By definition of the algorithm and again since x_τ has already been inserted to the stream, this region must be a hypercube with a diameter of at most $\Delta R \leq \max(\varepsilon_{\mathsf{hrd}} \cdot r/2, \frac{\varepsilon_{\mathsf{hrd}}}{2\tau^3})$ since otherwise, the region is refined again when x_τ is inserted. Cauchy Schwartz gives us

$$||x_\tau - c'_i||^2 \leq ||x_\tau - c_i||^2 + 2||x_\tau - c_i|| \cdot ||c_i - c'_i|| + ||c_i - c'_i||^2 \leq r^2 + r \cdot \Delta R + (\Delta R)^2$$

If $r \leq 1/\tau^3$ then $\Delta R \leq \varepsilon_{\mathsf{hrd}}/2\tau^3$, hence

$$||x_\tau - c'_i||^2 \leq ||x_\tau - c_i||^2 + \varepsilon_{\mathsf{hrd}}/\tau^6$$

Otherwise, $\varepsilon_{\mathsf{hrd}} \cdot r/2 > \frac{\varepsilon_{\mathsf{hrd}}}{2\tau^3}$ hence $\Delta R \leq \varepsilon_{\mathsf{hrd}} \cdot r/2$

$$||x_\tau - c'_i||^2 \leq ||x_\tau - c_i||^2(1 + \varepsilon_{\mathsf{hrd}})$$

Hence, accounting for the τ weight one gets

$$\ell(S', x_\tau) \leq \ell(S, x_\tau)(1 + \varepsilon_{\mathsf{hrd}}) + \varepsilon_{\mathsf{hrd}}/\tau^5$$

□

B.3 MTMW

The following is a proof for Lemma (3.12)

Proof. We will show a proof by induction on the subtree height h. For $h = 1$ we have $\tilde{V} = \{v\}$ hence the property holds. For $h + 1$ we can denote the children of v as U and use the induction hypothesis for each child's subtree and get that

$$\sum_{v' \in \tilde{V}} \mathcal{M}(v') = \sum_{v' \in U} \mathcal{M}(v') = \sum_{v' \in U} \mathcal{M}(v)/|U| = \mathcal{M}(v)$$

Where the last equality used the recursive mass definition. □

The following is a proof for Theorem (3.13)

Proof. For a rooted path $p = (v_1 \dots v_T)$ define the *uniform weight of p at time t* with respect to the stream $X_{1:t}$, as $u_p^{(t)} = \prod_{\tau=1}^{t-1}(1 - \eta\ell_1(v_\tau, x_\tau))$, which corresponds to the weight MWUA with uniform initial weights would associate that expert p before witnessing x_t. We extend this notation for any rooted path p, as long as it has length at least t. In our modified algorithm, the weight associated to expert p at step t is $w_p^{(t)} = \mathcal{M}(p)u_p^{(t)}$. Denote $p(v)$ the path from the root to vertex v. Using the modified initial weight, the probability MWUA will output the prediction that corresponds to any node v_t at depth t at step t, due to choosing some path that contains it, is given by (before normalizing)

$$w_{v_t}^{(t)} = \sum_{p \in \mathcal{P}(\mathcal{T}_T): v_t \in p} w_p^{(t)} = \sum_{p \in \mathcal{P}(\mathcal{T}_T): v_t \in p} u_p^{(t)} \cdot \mathcal{M}(p) = u_{p(v_t)}^{(t)} \cdot \mathcal{M}(v_t)$$

The weight and mass are functions of known quantities at step t, where the equality is from the definition of $u_p^{(t)}$ and due to Lemma (3.12). Following the proof of Theorem (2.1) of Arora et al. [3], we define

$$\Phi^{(t)} = \sum_{p \in \mathcal{P}(\mathcal{T}_T)} w_p^{(t)} \qquad (\mathbf{m}^{(t)})_p = \ell_1(p, x_t) \qquad (\mathbf{p}^{(t)})_p \propto w_p^{(t)}$$

The potential, the loss vector, and the normalized probability vector, respectively. For any path p, due to the fact $\Phi^{(1)} = 1$ we can modify Equation (2.2) in [3] to

$$\Phi^{(T+1)}/\mathcal{M}(p) \leq (1/\mathcal{M}(p)) \exp(-\eta \sum_{t=1}^{T} \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)})$$

Using the weight of p after the last step as lower bound for $\Phi^{(T+1)}$, we change Equation (2.4) to

$$\Phi^{(T+1)}/\mathcal{M}(p) \geq w_p^{(T+1)}/\mathcal{M}(p) = u_p^{(T+1)}$$

Hence the rest of the proof is left intact, where n (the amount of experts) is replaced with $(1/\mathcal{M}(p))$, so the regret changes to

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

The running time is composed of sampling a path which is done by iterating the vertices of depth t and calculating their weights, which finishes the proof. □

B.4 Combining the Components

The following is a proof for Lemma (3.15)

Proof. Denote the times where $\tilde{S}_t \neq \tilde{S}_{t-1}$ as $t_1, t_2 \dots t_{T_0}$, where we consider $t_1 = 1$ for this purpose, and T_0 is the amount of different time steps where this occurs. For ease of notation we denote $t_{T_0+1} = T + 1$. Furthermore, we use $X_{[a:b)}$ to denote $X_{a:b-1}$. As $\tilde{S}_t = \tilde{S}_{t-1}$ for any other time step we have that $\tilde{S}_{t_{i+1}-1} = \tilde{S}_{t_i}$ hence

$$\sum_{t=1}^{T} \ell(\tilde{S}_t, x_t) = \sum_{i=1}^{T_0} \ell(\tilde{S}_{t_i}, X_{[t_i:t_{i+1})}) \leq \sum_{i=1}^{T_0} \ell(\tilde{S}_{t_{i+1}-1}, X_{[t_i:t_{i+1})})$$

Excluding the T_0 points that resulted with a region refinement from the sum, and using the fact that the loss is bounded by d

$$\begin{aligned}\sum_{t=1}^{T} \ell(\tilde{S}_t, x_t) &\leq \sum_{i=1}^{T_0} \ell(\tilde{S}_{t_{i+1}-1}, X_{[t_i:t_{i+1}-1)}) + dT_0 \\ &\leq \sum_{i=1}^{T_0} \left(\ell(\tilde{S}_{t_{i+1}-1}, X_{[1,t_{i+1}-1)}) - \ell(\tilde{S}_{t_{i+1}-1}, X_{[1,t_i)})\right) + dT_0\end{aligned}$$

As the loss is nonnegative

$$\sum_{t=1}^{T} \ell(\tilde{S}_t, x_t) \leq \sum_{i=1}^{T_0} \left(\ell(\tilde{S}_{t_{i+1}-1}, X_{[1,t_{i+1}-1)}) - \ell(\tilde{S}_{t_{i+1}-1}, X_{[1,t_i-1)})\right) + dT_0$$

Using the coreset property

$$\sum_{t=1}^{T}\ell(\tilde{S}_t,x_t)\leq\sum_{i=1}^{T_0}(1+\varepsilon_{\mathsf{c}})\ell(\tilde{S}_{t_{i+1}-1},\chi(X_{[1,t_{i+1}-1)}))-\\ \sum_{i=1}^{T_0}(1-\varepsilon_{\mathsf{c}})\ell(\tilde{S}_{t_{i+1}-1},\chi(X_{[1,t_i-1)}))+dT_0$$

As the Hierarchical Region Decomposition $\mathcal{H}_{t_{i+1}-1}$ was constructed according to a superset of $\chi(X_{1:t_{i+1}-2})$ and $\chi(X_{1:t_i-2})$, one can use Corollary (3.10) and get

$$\sum_{t=1}^{T}\ell(\tilde{S}_t,x_t)\leq\sum_{i=1}^{T_0}(1+\varepsilon_{\mathsf{c}})(1+\varepsilon_{\mathsf{hrd}})\ell(S^*,\chi(X_{[1,t_{i+1}-1)}))-\\ \sum_{i=1}^{T_0}(1-\varepsilon_{\mathsf{c}})(1-\varepsilon_{\mathsf{hrd}})\ell(S^*,\chi(X_{[1:t_i-1)}))+dT_0+2\varepsilon_{\mathsf{hrd}}$$

Now the series telescope, and along with $\varepsilon_{\mathsf{c}},\varepsilon_{\mathsf{hrd}}<1$ rearranging gives

$$\sum_{t=1}^{T}\ell(\tilde{S}_t,x_t)\leq\sum_{i=1}^{T_0}((\varepsilon_{\mathsf{c}}+\varepsilon_{\mathsf{hrd}})\ell(S^*,\chi(X_{[1,t_{i+1}-1)})))+\ell(S^*,\chi(X_{[1,T)}))+2dT_0$$

As the loss is nonnegative we can extend the substreams until T, and using the coreset property

$$\sum_{t=1}^{T}\ell(\tilde{S}_t,x_t)\leq((1+\varepsilon_{\mathsf{c}})+8T_0(\varepsilon_{\mathsf{c}}+\varepsilon_{\mathsf{hrd}}))\ell(S^*,X_{1:T})+2dT_0$$

finally, with Lemma (3.6) we have that $T_0\leq k\cdot\Lambda$, which finishes the proof. □

The following is a proof for Lemma (3.16)

Proof. Using Lemma (3.6) we can see that $\log(\deg(v_t))$ can take a non zero value at most $k\Lambda$ times, each is bounded by $k\cdot\beta$ as each node can be replaced by $((9\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d\log(T^3))^k$ children nodes in the k-tree. □

The following is a proof for Theorem (3.1)

Proof. Using the bounds for $\varepsilon_{\mathsf{hrd}},\varepsilon_{\mathsf{c}}$ we have that

$$\Lambda=\log(\frac{2T^3\sqrt{d}}{\varepsilon_{\mathsf{hrd}}})\leq\log(\frac{4adT^6k^2}{\varepsilon^2})\leq 2\log(\frac{akT^3\sqrt{d}}{\varepsilon^2})$$

Next we will show that $\varepsilon_{\mathsf{hrd}}$ is of the same order as $\varepsilon/k\Lambda$

$$\varepsilon_{\mathsf{hrd}}\cdot k\Lambda\leq\frac{\varepsilon^2}{ak^2\log(T^3\sqrt{d})}\cdot 2k\log(\frac{ak\sqrt{d}T^3}{\varepsilon^2})\leq\\ \leq\frac{2\varepsilon^2}{ak\log(T^3\sqrt{d})}\log(T^3\sqrt{d})k\log(\frac{a}{\varepsilon^2})\leq 4(\frac{\varepsilon}{\sqrt{a}})^2\log(\frac{\sqrt{a}}{\varepsilon})\leq\frac{2\varepsilon}{\sqrt{a}}$$

As stated in Corollary (3.8), for sufficiently large T we have $\beta<d\cdot\Lambda$, hence, along with Lemma (3.16), we have that MWUA has a regret w.r.t. the best path of $k\Lambda\sqrt{dT}\leq$

$2k\log(\frac{akT^3\sqrt{d}}{\varepsilon^2})\sqrt{dT}$. For $a \geq 34^2$ we get $\frac{2\varepsilon}{\sqrt{a}} \leq \frac{\varepsilon}{17}$, which makes Lemma (3.15) bound the ε-Approximate Regret of the best path. hence the final regret of

$$O\left(k\log\left(\frac{akT^3\sqrt{d}}{\varepsilon^2}\right)\sqrt{d^3T}\right) + \varepsilon \cdot \mathcal{C}$$

Furthermore, as $|\mathcal{Q}_T| \leq k^2\log^4(T)\varepsilon_{\mathsf{c}}^{-4} = O(k^{10}\log^8(T)\log^4(d)\varepsilon^{-8})$, using Lemma (3.7) with $N = |\mathcal{Q}_T|$ we can bound the k-tree vertices by the amount of leaves times the depth T, and have that the runtime is

$$O(T(|\mathcal{Q}_T|(32\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d\log(T^3))^k) = T\cdot O(k^{10}\log^9(T)\log^4(d)\varepsilon^{-8})^k O(\sqrt{d}k^2\log(T)\varepsilon^{-2})^{dk}$$

□