Online Clustering

Guy Rom

Green Templeton College

University of Oxford

Dissertation submitted for the degree of

MSc in Computer Science

Trinity Term 2019

Acknowledgements

Personal

First and foremost, I would like to thank Varun Kanade for supervising this project. I once attended a lecture about the obscurity in which academic research is conducted, where one runs in loops inside a cloud trying to get from A to B, ending up in C instead. I've seen none of that. Varun has had a clear vision and strategy, with good instincts, where the great majority of initial suggestions have become the final solutions. I thank him for giving me a wrong impression of how monotonously improving academic research can be. Furthermore, he has endured my stress and doubts throughout the process, and for that I'm grateful.

Secondly, I'd like to thank Vincent Viallat Cohen-Addad, for being the champion of computational geometry in the project, having an immediate answer for every question, and always in good spirit. We have had countless Skype and phone calls, and I thank him for trying to understand the impossible explanations I gave.

I thank Phillip and Patricia Frost, whose generosity allowed me this wonderful opportunity. I thank my family for all their support, and specifically my grandmother for planting the seed that lead to my interest in science, by introducing a fascination for ladybugs and flower buds even before I knew how to count. Lastly, I thank Dina, my wife, for making my project hers, and being willing to listen, and listen, and listen...

Abstract

We study the online clustering problem using the k-means objective, as presented by Dasgupta [1], where an algorithm must allocate k cluster centers in a bounded box in $\mathbb{R}^d$ at each step, incurring a loss according to the squared distance between the next point and the closest cluster center. We claim that *regret analysis* is the correct framework in which to analyze this problem and show that no algorithm can obtain any finite competitive ratio. Hence we aim to minimize the regret with respect to the best k-means solution in hindsight.

We present an online-to-offline reduction which implies that although the online algorithm may change cluster centers at each time step, an efficient algorithm with sublinear regret for the online clustering problem admits an efficient approximation algorithm for the offline k-means problem, which is known to be NP-Hard to approximate arbitrarily well, even in the Euclidean setting [2].

We analyze the textbook algorithm termed *Follow The Leader* (FTL), which is computationally inefficient in the worst case hence is not limited by the lower bound described above. We show that it has linear regret in the worst case, analytically and experimentally. We perform preliminary experiments on two FTL variants, namely, we measure their regret on different permutations of MNIST and Gaussian Mixture Models, and show examples where they obtain logarithmic regret for these real world cases. We provide an initial stability analysis that may provide the framework to explain these results.

We analyze the *Multiplicative Weight Update Algorithm* (MWUA) applied to a grid discretization of space, which achieves $\tilde{O}(\sqrt{T})$ regret. This shows that, in our setting, computational complexity is the main challenge in devising low-regret algorithms, as opposed to information theoretic constraints.

We then turn to approximate regret minimization, where we present an incremental coreset construction (due to Cohen-Addad) and an adaptive variant of MWUA such that along with an adaptive region decomposition scheme we describe, obtains $\tilde{O}(\sqrt{T})$ approximate regret, with a computational complexity of $O\left(T(k^2\varepsilon^{-2}d^{1/2}\log(T))^{k(d+O(1))}\right)$.

We generalize the previous MWUA variant and present another method of using MWUA in adaptive algorithms.

Contents

1 Introduction **1**

1.1 Structure of Dissertation . . . 2
1.1.1 Account of Work . . . 2
1.2 Online Algorithms . . . 3
1.2.1 Stochastic Data . . . 3
1.2.2 Adversarial Data . . . 3
1.2.3 Performance Measures . . . 4
1.2.4 Textbook Problems . . . 4
1.2.5 Follow The Leader . . . 5
1.2.6 The Doubling Trick . . . 6
1.2.7 Streaming Algorithms . . . 6
1.3 Multiplicative Weight Update Algorithm . . . 8
1.3.1 Background . . . 8
1.3.2 The Algorithm . . . 8
1.3.3 Expected Regret of MWUA . . . 9
1.3.4 Alternative Perspectives for MWUA . . . 9
1.4 Clustering . . . 11
1.4.1 Background . . . 11
1.4.2 Centroid-Based Clustering and Voronoi Diagrams . . . 11
1.4.3 Exact k-means Hardness . . . 12
1.4.4 Approximate k-means Hardness . . . 12
1.4.5 k-means Algorithms . . . 12
1.5 Coresets . . . 14
1.5.1 Point Set Coreset . . . 14
1.5.2 k-means Coresets . . . 14

2 Related Work **15**

2.1 Related Problems . . . 15
2.2 Online k-means Clustering Variants . . . 16

3 Preliminaries And Notation **19**

4 Regret Minimization Using Grid Discretization **20**

5 Lower Bounds for Online Clustering **22**
5.1 Unbounded Competitive Ratio . . . 22
5.2 Regret Minimization is NP-Hard . . . 22

6 Follow The Leader **25**
6.1 Linear Regret Worst Case . . . 25
6.2 Sublinear Regret on MNIST and Simulated Data . . . 27
6.3 Stability for Adversarial Streams . . . 28

7 Approximate Regret Minimization Using Adaptive Methods **31**
7.1 Incremental Coreset . . . 32
7.2 Hierarchical Region Decomposition . . . 33
7.2.1 Adaptive Grid Hierarchical Region Decomposition . . . 33
7.3 MTMW– MWUA for Tree Structured Experts . . . 38
7.4 Approximate Regret Bound . . . 41

8 Adaptive MWUA **45**
8.1 Incremental Expert Sets . . . 45
8.1.1 Notation . . . 46
8.1.2 Regret . . . 47
8.2 MTMW for Refining Equivalence Classes . . . 49

9 Conclusion **50**
9.1 Summary . . . 50
9.2 Comparison with Previous Work . . . 52
9.3 Extensions . . . 54
9.3.1 Approximate Regret Lower Bound . . . 54
9.3.2 Simpler Approximate Regret Algorithm . . . 54
9.3.3 Applying the Johnson Lindenstrauss Transfomation . . . 54
9.3.4 Logarithmic Regret via Stochastic Stability . . . 55

Appendices

A Incremental Coreset **58**
A.1 Incremental Coreset . . . 58
A.1.1 Maintaining an $O(1)$-Approximation for k-means . . . 59
A.1.2 Maintaining a Coreset for k-means . . . 59

Bibliography **62**

1 Introduction

Contents

1.1 Structure of Dissertation 2
1.1.1 Account of Work 2
1.2 Online Algorithms 3
1.2.1 Stochastic Data 3
1.2.2 Adversarial Data 3
1.2.3 Performance Measures 4
1.2.4 Textbook Problems 4
1.2.5 Follow The Leader 5
1.2.6 The Doubling Trick 6
1.2.7 Streaming Algorithms 6
1.3 Multiplicative Weight Update Algorithm 8
1.3.1 Background 8
1.3.2 The Algorithm 8
1.3.3 Expected Regret of MWUA 9
1.3.4 Alternative Perspectives for MWUA 9
1.4 Clustering 11
1.4.1 Background 11
1.4.2 Centroid-Based Clustering and Voronoi Diagrams 11
1.4.3 Exact k-means Hardness 12
1.4.4 Approximate k-means Hardness 12
1.4.5 k-means Algorithms 12
1.5 Coresets 14
1.5.1 Point Set Coreset 14
1.5.2 k-means Coresets 14

1.1 Structure of Dissertation

In this chapter I will present *The Online Clustering Problem*, which is the main subject of this research, and will provide a broad background including an introduction to Online Algorithms, Multiplicative Weights Update Algorithm, Clustering and Coresets. Chapter 2 reviews the recent academic work that is most related to this research topic and will provide context for the main results. Chapter 3 will define the problem exactly and provide notations that will be used throughout the dissertation. Chapters 5, 4, 6, 7 will describe the main work and results for the online clustering problem and 8 will describe a generalization of some of the work done in 7, which applies to a broader range of adaptive online algorithms. Chapter 9 will discuss the conclusions of the work and possible extensions. A Bibliography is provided at the end of this document.

1.1.1 Account of Work

The work was supervised by Varun Kanade of Oxford University and Vincent Viallat Cohen-Addad from CNRS at Sorbonne Université. Benjamin Guedj of UCL also participated in discussion.

Chapter 7 contains a reference to the Incremental Coreset construction which is due to Cohen-Addad solely, hence it was moved to the appendix, and only the main theorem was left, for completeness. Others' results that are used are referenced and the relevant sources cited.

1.2 Online Algorithms

Online Algorithms are those who obtain the data incrementally, and perform decisions or predictions throughout the duration of the algorithm. There are two main settings that define the type of data stream the algorithm must handle– stochastic stream data and adversarial stream data.

1.2.1 Stochastic Data

Stochastic streams typically aim to model some complex real-world data by some approximate distribution. Popular models for stochastic data typically consist of IID samples from some fixed distribution, or a distribution that changes over time through some process, such as a drifting mean. A family of underlying distributions worth mentioning is the Gaussian Mixture Models (GMM) which corresponds to data points that came from several normally distributed populations, with some proportion between the different populations.

1.2.2 Adversarial Data

Adversarial Data is typically classified as one of three types, presented here in ascending difficulty:

1. *Oblivious* – the adversary commits to the entire stream in advance, knowing only the algorithm

2. *Adaptive Online* – at each time step the adversary commits to the next point in the stream, hence it knows the algorithm and the choices made so far. This is sometimes expressed as an adversary that knows the algorithm and the random bits it used up to step t.

3. *Adaptive Offline* – the adversary commits to the entire stream in advance, knowing the algorithm and all random bits the algorithm may use.

1.2.3 Performance Measures

Typically one defines the performance measure of an online algorithm as one of two standard notions which are the result of comparing some *loss* an online algorithm incurs to the loss that an offline algorithm incurs, i.e. an algorithm that gets access to the entire data at once. The first is called *competitive ratio*, which corresponds to a multiplicative approximation of the offline loss. The second one is called *regret* and corresponds to an additive error with respect to the offline loss. A combination of the two notions that is worth mentioning is the $\epsilon-$Approximate Regret, which corresponds to an ϵ approximation ratio plus an additive error term.

We will present a few notations in order to define the competitive ratio and regret. Let $X_{1:T} = \{x_t\}_{t=1}^T$ denote the stream up to time step t. An algorithm makes a prediction or a choice C_t for some (possibly infinite) set of choices before receiving point x_t; x_t is then revealed and the algorithm incurs a loss $\ell(x_t, C_t)$. Let C^* be the best solution in hindsight, i.e. $C^* = \arg\min_C \sum_{t=1}^T \ell(x_t, C)$. If $\{C_t\}_{t=1}^T$ then we can define the competitive ratio as $\frac{\sum_{t=1}^T \ell(C_t, x_t)}{\sum_{t=1}^T \ell(C^*, x_t)}$, and the regret as $\sum_{t=1}^T \ell(C_t, x_t) - \sum_{t=1}^T \ell(C^*, x_t)$. An algorithm is said to have some competitive ratio or regret, if it can guarantee this performance on any stream of the corresponding data setting. An asymptotic performance guarantee of $O(f(T))$ for some function f means that there exists two positive numbers N, a such that for any *sequence of streams* of increasing length T, $\{S_T\}_{T=1}^\infty$, running the algorithm on S_T for $T > N$ guarantees a performance better than $a \cdot f(T)$.

1.2.4 Textbook Problems

The Ski Rental Problem

The Ski Rental Problem is a standard example used to demonstrate the competitive ratio concept. It consists of a skier, that can either rent skis for 1\$ per day, or buy them for 10\$. The skier doesn't know how many days he will ski, hence the best strategy is not known a-priori. The break-even algorithm by Karlin et al. [3] suggests that the skier rent skis for the first 9 days and buys them on the 10^{th}

day, promising to pay the *best in hindsight* price on the first 9 days, and on the 10^{th} on pay at most 19\$ which is a factor of 1.9 worse than the *best in hindsight* price, guaranteeing a competitive ratio of 1.9.

Prediction With Expert Advice

Prediction With Expert Advice is a widely applicable online setting in which there exist a set of experts $\mathcal{E}$, and the algorithm is presented with advice $f_i^{(t)}$, which corresponds to the choice expert i makes at step t. The regret in this setting is defined with respect to the best expert, i.e.

$$\sum_{t=1}^{T} \ell(C_t, x_t) - \min_{i \in \mathcal{E}} \sum_{t=1}^{T} \ell(f_i^{(t)}, x_t)$$

An example of a classic problem defined in this setting is the *Portfolio Selection Problem* where one selects an investment portfolio at each time step and compares to the best single stock performance in hindsight. The next section will present MWUA, which is a meta-algorithm that provides a regret minimization approach, which minimizes regret for this setting [4].

1.2.5 Follow The Leader

Follow The Leader (FTL) Is a natural online algorithm that chooses the solution that minimized past loss, i.e. the optimal solution for the stream up to the current time step. The performance of FTL depends on the objective function, and one such family of objective functions, namely, $\sigma-$strongly convex function, which satisfy $f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\sigma}{2}\|y-x\|_2$for all x, y, one can obtain a regret bound of $\frac{L^2}{2\sigma}(\log(T)+1)$. For a derivation one may refer to Shalev-Shwartz and Kakade [5].

Randomization, FTPL and FTRL

One can show that for some objective no deterministic algorithm can obtain sub-linear regret (e.g. Dekel [6]) hence a few randomized variants of FTL evolved. Follow The Perturbed Leader (FTPL), first presented in Kalai and Vempala [7], adds random noise to the loss of an expert before choosing the leader, which obtains sub linear

regret for appropriate noise and objective. A second variation chooses a leader in the space defined by probability distributions over the experts, a simplex, in a deterministic manner, and then samples a concrete expert from this probability distribution. Adding a regularization term to the original problem objective for this simplex FTL leads to an algorithm called *Follow the Regularized Leader* (FTRL), which is a very versatile algorithm, which is very easy to describe, and many online algorithms can be viewed as FTRL with a specific regularizer term, such as online gradient descent, and even the MWUA algorithm presented in the next section. The reader is referred to Shalev-Shwartz et al. [8] and McMahan [9] for further reading.

1.2.6 The Doubling Trick

The parameters of online algorithms may depend on the time horizon which is not known in advance, e.g. η, a parameter of MWUA, depends on T (see 1.3) to balance the terms and obtain the final $\sqrt{T \cdot \log(N)}$ regret. A textbook solution for the issue is referred to as the *Doubling Trick* and consists of repeatedly running the algorithm with $T = 2^m$ steps for $m = 0, 1, \ldots \lceil \log(T) \rceil$ and obtain a modified regret bound, which is the sum of regret in the different segments. For a regret that is linear in $\sqrt{T}$, this regret is still linear in $\sqrt{T}$. For a derivation one may refer to Shalev-Shwartz et al. [8].

1.2.7 Streaming Algorithms

Streaming Algorithms are closely related to Online Algorithms, where both obtain the data incrementally, but they differ in the way their objective is described: While streaming algorithms will aim to obtain some solution at the end of the algorithm which will have some performance guarantee on the entire data set, online algorithms will generate **intermediary** solutions with some performance guarantee. The emphasis of streaming algorithms is typically on memory restrictions, and the entire data set may already be available at the start of the algorithm, while in

online algorithms one may allow saving the entire stream of data available so far and revisiting the entire history to generate the next solution.

1.3 Multiplicative Weight Update Algorithm

1.3.1 Background

MWUA is a Meta-Algorithm, whose variants were discovered several times in different fields– from solving matrix games in Game Theory, Through Winnow and ADA-Boost in Machine Learning to LP solving in Convex Optimization. All of these algorithms start from an arbitrary choice and update it to account for historical performance, be it in a probabilistic manner or not. Fictitious play is an online perspective of playing matrix games– a player starts with an arbitrary initial strategy and chooses the deterministic *best response* action to the historical frequencies of its opponent. Grigoriadis and Khachiyan [10] showed that a multiplicative weights version of fictitious play finds a strategy that approximates the game value. Winnow2 performs binary linear classification and multiplies the weight of each dimension according to historic performance and obtains low mistake bounds [11]. ADA-Boost, introduced by Freund and Schapire [12], is a classifier and training algorithm that uses weak learners, efficient classifiers that approximate a classification problem to a $1/2 + \gamma$ extent, by weighting their predictions using an exponential term that relates to their loss. For LP, using the Prediction with Expert Advice setting where each constraint is an expert and expressing the LP solution as a convex combination of the constraints, one can average the intermediary distributions of MWUA to obtain a solution that approximates the objective, as presented in Garg and Könemann [13] or Plotkin et al. [14]. For a thorough survey one may refer to Arora et al. [4].

1.3.2 The Algorithm

MWUA is best expressed in the finite *Prediction with Expert Advice* setting described earlier– each one of N experts makes some prediction (advice) from some set or space, and when points arrive an associated loss function assigns the loss each expert incurs. MWUA maintains a weight for expert i at any time step t which we denote $w_i^{(t)}$. At step t it samples an expert with a probability which is proportional to the

expert's weight at that time step, and outputs its advice, incurring the expert's loss at that time step. The weights are initialized to 1 at the beginning and every time a new point x_t is presented the weights are updated to account for the new loss information $w_i^{(t+1)} = u(\eta \cdot \ell(f_i^{(t)}, x_t))w_i^{(t)}$, where for $u(y) = (1-y)$ we get the variation that is typically referred to as the multiplicative weight algorithm, and for $u(y) = \exp(-y)$ we get the *Hedge Algorithm* also referred to as the exponential weight algorithm. η is chosen to balance the different terms in the regret guarantee and typically is a function of the time horizon T and the number of experts.

1.3.3 Expected Regret of MWUA

The Expected Regret proofs for MWUA take several forms but typically follow by bounding the total sum of weights at each step, which we denote $\Phi^{(t)} = \sum_i w_i^{(t)}$. A recursion relation is derived between $\Phi^{(t+1)}$ and $\Phi^{(t)}$ using the losses and weight distribution at step t [4]. By applying this recursion relation one can attain an upper bound on $\Phi^{(T)}$, which is lower bounded by the weight of the best expert, resulting with a bound on the regret. Two versions of this proof achieve a bound in terms of the cumulative loss that any fixed distribution over experts incurs, or in terms of the squared loss the algorithm incurred. In addition, both have a $\ln(N)/\eta$ regret term. Typically one uses a bound on the loss, and chooses η to balance the terms, leading to a regret of $O\left(\sqrt{T\ln(N)}\right)$. A different analysis yields a different bound instead of $\ln(N)/\eta$ which is $\mathrm{RE}(\mathbf{p}^{(1)}||\mathbf{p})/\eta$, where RE is the relative entropy and $\mathbf{p}^{(1)}$ and $\mathbf{p}$ are the initial weight distribution and the optimal fixed expert distribution, which gives further insight to the evolution of the initial distribution. One can show that MWUA is following the regularized leader, with the relative entropy as regularizer [8]. The next subsection will extend on this concept.

1.3.4 Alternative Perspectives for MWUA

This subsection is slightly more tricky and is aimed for readers who are acquainted with the world of online optimization. The reader may skip this subsection without

loss of continuity.

As Convex Optimization On The Simplex

Shalev-Shwartz et al. [8] show that although MWUA outputs a sampled prediction at each step, it actually moves from one distributions over experts to another at each step, hence it can be though of as performing a *convex* optimization process in the standard N-simplex $\Delta_N \subset [0,1]^N$ that corresponds to the space of all distributions over the experts even when the loss is not convex in the advice space, i.e. for $\mathbf{p} \in \Delta_N$ and $\mathbf{m} \in [-1,1]^N$ a vector of the loss each expert will incur at step t, the expected loss MWUA incurs is $\ell(\mathbf{p}, \mathbf{m}) = \mathbb{E}_{i \sim \mathbf{p}}[\mathbf{m}_i] = \mathbf{p} \cdot \mathbf{m}$ which is convex in $\mathbf{p}$. The next subsection gives an even more general perspective for MWUA as convex optimization process.

The Optimized Potential Functions

Cesa-Bianchi and Lugosi [15] present an interesting perspective for the MWUA algorithm. Let $\hat{p}_t$ be the prediction (or distribution over predictions) an algorithm made at step t, $f_i^{(t)}$ be the advice given by expert i at that step t, $r_{i,t} = \ell(\hat{p}_t, x_t) - \ell(f_i^{(t)}, x_t)$ be the instantaneous regret caused by not using that expert's advice at that step and $\mathbf{r}_t = (r_{1,t}, r_{n,t})$ be the instantaneous regret vector at step t, and $\mathbf{R}_{\tilde{T}} = \sum_{t=1}^{\tilde{T}} \mathbf{r}_t$ the regret vector. We define a potential function $\Phi : \mathbb{R}^N \to \mathbb{R}$ of the form

$$\Phi(\mathbf{u}) = \psi(\sum_{i=1}^{N} \phi(u_i))$$

where ϕ is non-negative, increasing and twice differentiable and ψ is non-negative, strictly increasing concave and twice differentiable. This potential is said to derive the MWUA weights if $w_i^{(t)} = (\nabla \Phi(R_{t-1}))_i$. One can then show that MWUA is moving against the gradient of Φ irregardless of the value of x_t at each stage, pushing R_t towards the minimum of Φ. The generalization of the previous proofs yields some generic regret bound in ϕ and ψ derivative terms. Choosing different functions yield MWUA variations with corresponding regret bounds: $\Phi_p(\mathbf{u}) = (\sum_{i=1}^{N} (u_i)_+^p)^{2/p}$ corresponds to a polynomial weighted updates and $\Phi_p(\mathbf{u}) = \frac{1}{\eta} \ln(\sum_{i=1}^{N} e^{\eta u_i})$ to the exponentially weighted version.

1.4 Clustering

1.4.1 Background

Clustering

Clustering is the process of partitioning a set of points either to subsets or to a tree structure (typically referred to as *Hierarchical Clustering*). Non-Hierarchical clustering consists of several types, including Centroid-based clustering (*k-means*), Distribution-based clustering (Gausssian Mixture Models) and Density-based clustering (DBSCAN, Spectral Clustering). Distribution-Based clustering differs from the rest in that it performs a soft partition, and rather that allocating a specific cluster identifier to each point (hard clustering), it gives the likelihood a point belongs to some clusters (soft clustering). Density-Based clustering algorithms such as DBSCAN may also admit *outliers* – points that are considered *noise* and not associated to any cluster.

1.4.2 Centroid-Based Clustering and Voronoi Diagrams

Centroid-based clustering is usually defined as an optimization problem. Let $X \subset \mathcal{R}$ be a set of points in some metric space $(\mathcal{R}, \rho)$, k a positive integer and ℓ some loss function, then find k prototypes $\boldsymbol{\mu} = \mu_1, \ldots, \mu_k$ from some set $S \subset \mathcal{R}$ such that the sum of losses for the optimal allocation to prototypes is minimized. i.e.

$$\min_{\boldsymbol{\mu} \in S^k} \left(\sum_{x \in X} \min_{\mu \in \boldsymbol{\mu}} \ell(\mu, x) \right)$$

The *Voronoi Diagram* of these prototypes partitions $\mathcal{R}$ (hence X) to subsets, that correspond to all the points in $\mathcal{R}$ (resp. X) which are closest to a specific prototype of the k prototypes. The *k-means Objective* is defined by choosing $S = \mathcal{R}$ and $\ell = \rho^2$ the squared distance. The *k-median Objective* is the result of choosing $S = \mathcal{R}$ and $\ell = \rho$ as the distance. For $S = X$ and $\ell = \rho$ we get the *k-Medoid Objective*. When the sum in the objective is replace by a *max* operation, $S = \mathcal{R}$ and ℓ is the distance we get the *k-Centers Objective*. The loss of the weighted k-means problem is defined similarly, given a weight function $\omega : X \to \mathbb{R}_+$, as $\ell(\boldsymbol{\mu}, X) = \sum_{x \in X} \omega(x) \min_{\mu \in \boldsymbol{\mu}} \|x - \mu\|_2^2$.

1.4.3 Exact k-means Hardness

The k-means optimization problem has been shown to be NP-Hard to solve exactly by Dasgupta [16]. An Exact algorithm for the Euclidean k-means problem was presented by Inaba et al. [17], where they show that there are at most $O(n^{kd})$ different Voronoi Partitions of n points due to k centers, hence a simple enumeration yields an optimal result.

1.4.4 Approximate k-means Hardness

The Euclidean k-means Problem, that naïvely may be an easier problem than the general one due to the additional Euclidean properties of the space, was shown to be hard to approximate by Awasthi et al. [2]. They prove that there exists an $\epsilon > 0$ such that approximating the k-means objective to more than a $(1 + \epsilon)$ factor is NP-Hard. PTAS (with FPT complexity) were given by Kumar et al. [18] with complexity $f(k, \epsilon) \cdot dn$ and were improved by Feldman et al. [19]. Cohen-Addad et al. [20] found a $(1 + 8/e + \epsilon)$ approximation with better complexity $f(k, \epsilon)n^{O(1)}$ and have shown that this is tight under the Gap-ETH assumption. Low-dimension PTAS (or in complexity terms, fixed k) were introduced by Friggstad et al. [21] and Cohen-Addad et al. [22].

1.4.5 k-means Algorithms

Lloyd's Algorithm

Lloyd's Algorithm [23], also known as *the k-means algorithm*, is a simple local search algorithm that starts from some initial cluster assignments of n points, and iteratively performs two stages:

1. Assignment: Assign each point the label of the closest cluster mean
2. Update: Replace the current means with the centroids of the current clusters.

This process is very similar to the *Expectation Maximization* algorithm, and it converges to a local minimum when the cluster assignments do not change after an iteration, hence the algorithm halts. As this only uses the pairwise distance between

points it is valid for non Euclidean distances as well but it may not converge in these cases. Several initialization/seeding schemes are used– Forgy chooses k points from the data points as initial means, whereas Random Partition assigns random labels to each of the data points [24]. Two interesting papers by David Arthur et. al show that Lloyd's algorithm may require $2^{\Theta(\sqrt{n})}$ iterations before converging [25], and that the algorithm has polynomial smoothed complexity, i.e. adding noise to the data points before running the algorithm, and taking the expected runtime [26].

k-means++

k-means++ is a popular seeding algorithm (it is the default seeding scheme in python's SKLearn) which was presented by Arthur and Vassilvitskii [27]. It guarantees a $\log(n)$ approximation ratio for the optimal clustering. The standard Lloyd's algorithm that follows can only improve the objective obtained, as it is a local search algorithm. The k-means++ algorithm follows these steps

1. Uniformly choose a data point as the first cluster center
2. Per data point compute $w_x = D(x)^2$, the squared distance to the closest existing cluster center
3. Add a new cluster center by choosing a data point x with a probability proportional to w_x
4. Repeat the last two steps until k centers are chosen

A variation of the k-means++ algorithm was presented in Aggarwal et al. [28] by adaptive sampling $O(k)$ centers and choosing the best subset of size k using LP, which produces a constant approximation with constant probability. A more scalable version of k-means++, nicknamed k-means||, was presented by Bahmani et al. [29] and requires less passes over the data.

1.5 Coresets

1.5.1 Point Set Coreset

The term Coresets refers to the paradigm used for approximating measures of a point set P by computing efficiently a small subset Q of P which is referred to as a *Coreset*, that approximates the original set P in the following sense– solving some computational problem on Q can be translated to a solution which is a good approximation of the same problem on the original point set P. Sometimes, if the coreset is sufficiently small, using relatively inefficient algorithms to solve the problem on Q may yield a more efficient algorithm. The coreset is typically referred to as a $(1+\epsilon)-$coreset or $\epsilon-$coreset if the measure of P is preserved for Q to within a $(1+\epsilon)$ ratio. *Random Sampling*, *Feature Extraction* and $\epsilon - Samples$ are all techniques used in the context of approximation algorithms and are generalized by the coreset concept. For a comprehensive survey one may refer to Agarwal et al. [30].

1.5.2 k-means Coresets

Given a set of points P in $\mathbb{R}^d$, a triple $(Q, \omega, \mathcal{M})$, $Q \subseteq \mathbb{R}^d$, $\omega : Q \mapsto \mathbb{R}_+$, $\mathcal{M} : P \mapsto Q$ is a (strong) $(1+\varepsilon)-$*coreset* for k-means if for any $\alpha > 0$, If a $k-$partition $\{Q_1, \ldots, Q_k\}$ of Q is an $\alpha-$approximation to the (weighted) k-means problem on Q, the partition $\{P_1, \ldots, P_k\}$ where $P_i = \{p \mid \mathcal{M}(p) \in Q_i\}$, is an $\alpha(1+\varepsilon)-$approximation to the k-means problem on P. Relevant examples for efficiently computing k-means coresets can be found in Chen [31] and Har-Peled and Mazumdar [32], and are typically the result of a combination of space discretization techniques and random sampling.

2
Related Work

2.1 Related Problems

Online Facility Location

Meyerson [33] solves a similar problem termed *Online Facility Location*, where we are not restricted to k centers but rather we pay an additive cost per new facility (cluster center) we add, and the facility may not be moved. The k-means loss of the each point is referred to as the service cost. The online facility location objective is the sum of the facility cost and the k-means objective. Meyerson shows that no constant approximation algorithm for adversarial data exists and presents a $\log(n)$ approximation algorithm, and a constant approximation algorithm for data inputted in random order. In that sense, Our work resembles the definition of *online*– we focus on the *service cost* (using the already located facilities), but allow to use exactly k facilities, and move them from step to step.

The k-Server Problem

The k-Server Problem, presented by Manasse et al. [34], has some similarity to the online clustering problem and is a major problem in online optimization. An algorithm receives a stream of points (requests) in a metric space and must move one of k existing servers (also represented by points) to service the request. The objective is to minimize the sum of distances travelled by the servers compared to the best offline solution and is typically analyzed in the competitive ratio framework. In our setting,

we are not forced to move one of the k cluster centers but rather pay the squared distance to the closest one. Koutsoupias [35] provides a good survey on the problem.

2.2 Online k-means Clustering Variants

Online k-means Clustering

Dasgupta [1] presents the online clustering problem and the evaluation formalism we use here, and leaves it as an open problem, while presenting a naïve online algorithm that can be dubbed as *follow an arbitrary initial clustering* without analysis.

Online Cluster Labelling

Liberty et al. [36] present a different definition of *online*– the points are labelled 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 compare to the best k clusters in hindsight.

Bayesian Online Cluster Labelling On Sphere

Mansfield et al. [37] restrict their attention to unit spheres in high dimension and use the vector angle as distance measure. They define a bayesian model that generates the cluster centers and from these the data points using a fixed distribution around each of the cluster centers, which is exponentially decreasing. Lastly, they use the same online definition as Liberty et al. [36] and their loss is calculated for the centroids of the final clusters.

Consistent k-Clustering

Vassilvitskii and Lattanzi [38] present another similar problem which asks how many times one needs to change a fixed set of k-cluster centers in order to maintain some competitive ratio with respect to the offline clustering objective. They extends Meyerson [33] to the consistent k-median problem and analyze an algorithm from

Charikar et al. [39] and show that it performs well for the consistent k-center problem. They present a lower bound for the amount of re-clustering needed to maintain a constant approximation at any step.

MCMC Solution for online k-means clustering

Li et al. [40] tackle the online clustering problem with $O(k)$ clusters, and show a regret bound for a Gibbs sampling algorithm over the continuum that is not tractable, and present a *Reversible Jump MCMC* approach for sampling the varying amount of cluster centers, with no theoretical mixing time analysis.

Online k-means Clustering with Experts

Online Clustering with Experts presented in Choromanska and Monteleoni [41], use almost the same problem definition we use. They devise an algorithm that uses auxiliary black box batch clustering algorithms (*experts*) which they apply for sliding windows of size W. If the auxiliary algorithms give a b-approximation of the k-means objective on any batch, their algorithm has an approximate-regret of $O(\log n)$ with $\epsilon = bW/4R^2$ where R is a bound on data norm. Their work assumes that the auxiliary black box algorithms provide their algorithm with their closest cluster center to the next point in the stream. This is possible only due to the fact that the auxiliary black boxes are allowed to observe the next data point before the online algorithm commits to the current clustering. They provide a proof that shows that the auxiliary algorithms can't naïvely minimize the regret by always outputting the next point in the stream as it would break their approximation guarantee over the sliding window. This thesis deals with a setting in which the algorithm must commit to a clustering *strictly* before observing the next point, or any implicit information about the new point.

Online k-means clustering for $k = 1$

Shalev-Shwartz et al. [8] examine the case of online convex optimization for the squared distance objective with respect to a vector z_t that changes in each step,

which corresponds to the $k = 1$ case of online k-means clustering. They present an analysis that shows that follow the leader obtains a regret of $4L^2(\log(T) + 1)$ where $\forall t : \|z_t\|_2 \leq L$ is a bound on norm.

3

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 centers using only $X_{1:t-1}$. We refer to the set of all the candidate centers 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 centers 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 $L_T - (1+\epsilon)\mathcal{C}$.

We denote the best k-means solution, i.e. best k cluster centers, 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. Sometimes, we will be interested in the projection of a set of centers C to a 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 $\|C_i - S\|_\infty$. We use $\tilde{O}$ to suppress factors that are poly-logarithmic in T and polynomial in k and d.

4

Regret Minimization Using Grid Discretization

While MWUA is the go to solution for regret minimization, there are challenges in applying MWUA to the online clustering setting. One has to handle the fact that there is an infinite set of possible cluster centers. Broadly speaking, there are three main approaches one can modify MWUA in order to apply it to the online clustering setting. The first includes having a continuous expert set, which requires the construction of a sampler for the continuous distribution induced by the MWUA weights, and bounding the regret, as the original MWUA has regret bounds for finite sets. The second requires a very large discrete set of points that act as a sufficiently dense ϵ-net of the bounded space, which will be the main focus of this section. The third is handling an adaptive set of experts that depends on the stream data, by modifying MWUA to support an evolving expert set, and bounding the regret of such an algorithm. Devising such an adaptive expert set and modifying MWUA to fit these needs will be the task of section 7, but the bound we obtain is for approximate regret.

We turn now to define an algorithm that follows the second approach described above. Consider the *sites* corresponding to 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 $\delta = T^{-\alpha}$.

Before we prove the main theorem we provide a useful lemma with no proof, which appears in the folklore.

Lemma 4.1. *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 4.2. *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 4.3. *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 $0 < \alpha \leq 1/4$, with $\delta = T^{-\alpha}$, the MWUA with 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. [4]. 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 MWUA requires 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 $\|c^*_{T+1} - C_g\|_\infty \leq \frac{\sqrt{d}}{2}\delta$, (4.2) gives $\ell(C_g, X_{1:T}) - \ell(C^*_{T+1}, X_{1:T}) \leq \frac{d\delta^2}{4}T$. Hence the regret of the algorithm is at most $2\sqrt{kd\ln(\delta^{-1})T} + \frac{d\delta^2}{4}T$, so choosing $\delta = T^{-\alpha}$ for $\alpha \in (0, \frac{1}{4}]$ yields an algorithm with $\tilde{O}(T^{1-2\alpha})$ regret and $O(T^{\alpha kd+1})$ time complexity. □

5 Lower Bounds for Online Clustering

Contents

5.1 Unbounded Competitive Ratio 22
5.2 Regret Minimization is NP-Hard 22

5.1 Unbounded Competitive Ratio

Next we will present an instance of the online k-means clustering problem in one dimension with $k = 2$ that demonstrates that no algorithm can achieve any meaningful guarantee for the competitive ratio. This leads this thesis to focus on regret minimization for online k-means clustering. The stream consists of n points located in $x = 0$, and the last point is uniformly sampled in the unit interval. The expected loss incurred at the last step is positive for any algorithm, but in hindsight there is a clustering whose loss over the entire stream is zero, which is the one that puts one cluster center at $x = 0$ and one at the location of the last point in the stream, leading to an unbounded competitive ratio.

5.2 Regret Minimization is NP-Hard

This section will show that one cannot expect much more from regret minimization algorithms for the online clustering problem than the disappointing complexity

of the grid algorithm described in 4. We show that the complexity of obtaining sublinear regret is lower bounded by the complexity of approximating the offline k-means problem as shown in the following theorem. As Awasthi et al. [2] show, the offline k-means problem is NP-Hard to approximate, making regret minimization NP-Hard in our case.

Theorem 5.1. *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 $n, k, d, \frac{1}{\epsilon}, f(n,k,d)$.*

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

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

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

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

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

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

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 $\forall t : \epsilon_t \geq 0$.

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

Rearranging

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

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

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

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. [2] shows that this is NP-Hard, finishing the proof. □

6
Follow The Leader

Contents

6.1 Linear Regret Worst Case 25
6.2 Sublinear Regret on MNIST and Simulated Data 27
6.3 Stability for Adversarial Streams 28

The lower bound presented here does not imply anything on the regret guarantees of *Follow-The-Leader* (FTL) as it solves the offline problem internally, hence it is not an efficient algorithm in the worst case. The next section presents a case where FTL incurs linear regret for $k = 2$, which complements the analysis of Shalev-Shwartz et al. [8] for $k = 1$, along with experimental results. The section that follows examines FTL regret on various more natural data sets.

6.1 Linear Regret Worst Case

The following theorem shows that FTL is not a sublinear algorithm in the worst case.

Theorem 6.1. *FTL obtains $\Omega(T)$ regret in the worst case, for any fixed $k \geq 2$ and any dimension.*

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.

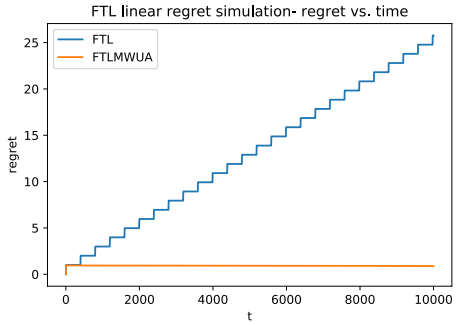
The stream will consist of points in 3 locations $-\delta, 0, (1-\delta)$ for $\delta < \frac{1}{4}$. We will call C_t a $(-\delta)$-clustering if it puts all the points at $(-\delta)$ in one cluster and the rest in the other cluster, and a $(1-\delta)$-clustering puts all the point at $(1-\delta)$ in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

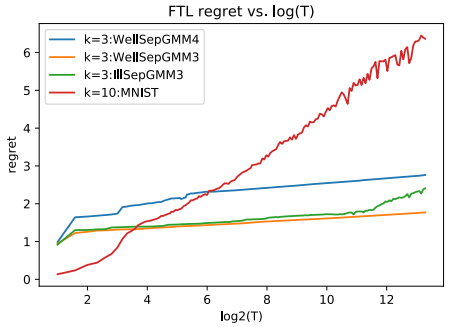
Our algorithm will start with a point in $(1-\delta)$ making C_t a $(-\delta)$-clustering. The next points will be added as follows– as long as C_t is a $(1-\delta)$-clustering add a point to $(-\delta)$ or 0, balancing them; if C_t has just become a $(-\delta)$-clustering, the next point will be $(1-\delta)$, 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, FTL moves 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. □

Figure 6.1a 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



(a) Linear Regret for FTL and Sublinear Regret for MWUA-FTL



(b) Regret for FTL on MNIST and Gaussian Mixure Models

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 due to negligent regret during the episodes in which the leader is in $(1-\delta)$-clustering and then a constant jump every time the leader transitions back from a $(-\delta)$-clustering. This demonstrates that the counter example is viable and is 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.

6.2 Sublinear Regret on MNIST and Simulated Data

Figure 6.1b 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.

6.3 Stability for Adversarial Streams

If the stream satisfies some stability constraints, strong regret bounds can be obtained for FTL, extending the results of Shalev-Shwartz et al. [8] for the $k = 1$ case, aiming to account for the experimental results described here. A simple approach to define such stability constraints will aim to partition the kmeans problem to k distinguishable $k = 1$ problems, where each point in the stream will immediately be assigned to some sub problem, which will result with $O(L^2 k \log(T/k))$ regret, where L is the diameter of the space.

To discuss such conditions, we will refer to the adversarial stream as coming from some ordering of the union of sets that corresponds to the optimal partition, so a point has an associated cluster label that corresponds to the optimal partition label, and the Voronoi Diagram that corresponds to the optimal cluster centers will partition the space to *cluster regions* that are the voronoi cells associated to these cluster centers. The convex hull of each of the subsets will be called the *tight cluster region*.

Stability conditions that enable partitioning the problem to k distinct $k = 1$ problems are *balanced streams* and *well separated* clusters. These are sufficient, but might not be necessary.

Formally, a simplest version of a well separation criteria is a γ-*hard separation* for $\gamma \geq 1$ that means that for any two points p_1, p_2 from the same cluster in the optimal partition and p_3 from another cluster, $\gamma\|p_1 - p_2\|_2 < \|p_1 - p_3\|_2$. The hardest balance condition is the *round robin* stream, which requires that in any consecutive substream of length k contains exactly one point from each of the optimal clusters.

Informally, these conditions aim to keep the optimal solution for any substream $X_{1:t}$ in the form of 1 center in each of the k tight cluster regions. In addition to that,

they force FTL to associate every new point to the same label it receives in the optimal partition, due to the fact that they force the diameter of each tight cluster region to be smaller than the distance to the closest point from another tight cluster region.

Theorem 6.2. *For a stream X which is both* round robin *and* 1-hard separated*, FTL guarantees a regret of at most $O(L^2 k \log(T/k))$ where L is the diameter of the space*

Proof. After k steps, FTL will chose these k points as the cluster centers, meaning that there will be one cluster center in each of the tight regions, hence every consequent point will be assigned a final cluster label (which is the same as in the best solution in hindsight), resulting with the problem being equivalent to k distinct $k = 1$ problems, hence we get a regret of $O(L^2 k \log(T/k))$. □

One can relax the *round robin* balance condition to obtain *uniform $\rho-$well balanced* for $1 \leq \rho \leq k$: For any time step $t > k\rho$ and any cluster label i, Let $q_i^{(t)}$ denote the amount of points in $X_{1:t}$ that are from optimal cluster i, then $1/\rho \leq \frac{kq_i^{(t)}}{t} \leq \rho$.

Theorem 6.3. *For a stream X which is both* uniform $\rho-$well balanced *and* γ-hard separated*, for a large enough $\gamma = f(\rho, k)$, FTL guarantees a regret of at most $O(L^2(k \log(\rho T/k) + k\rho))$ where L is the diameter of the space*

Proof. We'll prove the theorem for the case all tight clusters regions have the same diameter, Δ, but the proof should extend to the general case. For any $t > k\rho$, FTL will chose a point from each tight cluster region as the cluster centers for step t, hence every consequent point will be assigned the final label (which is the same as in the best solution in hindsight), resulting with the problem being equivalent to k distinct $k = 1$ problems after the first $k\rho$ steps, after which we get a regret of $O(k \log(\rho T/k))$.

Let's consider 2 Voronoi partitions of the stream points– P^* partitions according to tight cluster regions and $\tilde{P}$ that doesn't respect the tight cluster regions (hence one of the tight cluster regions doesn't contain a cluster center due to the γ-hard

separation). Finally P^k is the overlay of the two partitions, i.e. it partitions the stream to at most k^2 subsets, each is from the same tight cluster region and the same cluster in $\tilde{P}$. clearly, P^k has a loss which is smaller than $\tilde{P}$ and P^*. Using the closest center in P^* for cluster $C \in P^k$ will result with an addition to the loss of at most $|C| \cdot \Delta^2$ (due to (4.2)). Denoting $\ell(P^*)$ as the loss of the centroids of the partition P^*, and analogously for the other partitions, we have that $\ell(P^*) - t \cdot \Delta^2 \leq \ell(P^k)$.

On the other hand, let r be the cluster from the optimal partition that has the largest distance to the closest cluster center in $\tilde{P}$, and denote this distance λ. λ must be larger than $\Delta \cdot (\gamma - 1)$ in order for $\tilde{P}$ to not respect the tight cluster regions. Hence this tight cluster region incurs a loss in $\tilde{P}$ which is larger that the sum of the cost of the clusters that correspond to it in P^k by at least $|r| \cdot \lambda^2$, while all other clusters have a loss which is larger or equal the corresponding P^k cluster losses, hence $\ell(\tilde{P}) \geq \ell(P^k) + |r| \cdot \Delta^2(\gamma - 1)^2$. The uniform $\rho-$well balanced condition means that after $k\rho$ steps, $|r| \geq \rho^{-1}\frac{t}{k}$ hence $\ell(\tilde{P}) - \rho^{-1}\frac{t}{k} \cdot \Delta^2(\gamma - 1)^2 \geq \ell(P^k)$

Combining the two gives $\ell(P^*) \leq \ell(\tilde{P}) + t\Delta^2(1 - \frac{(\gamma-1)^2}{k\rho})$ hence when $1 + \sqrt{k\rho} < \gamma$ then P^* is the optimal clustering for the stream at any step t, making the labels final and optimal (after the first $k\rho$ steps), hence we get k distinguishable $k = 1$ problems. This gives the bound, using the $\rho-$balance condition again. □

7

Approximate Regret Minimization Using Adaptive Methods

Contents

7.1 Incremental Coreset 32
7.2 Hierarchical Region Decomposition 33
7.2.1 Adaptive Grid Hierarchical Region Decomposition 33
7.3 MTMW– MWUA for Tree Structured Experts 38
7.4 Approximate Regret Bound 41

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 7.1 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 7.2 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, 7.3, referred to as *Mass Tree MWUA* (MTMW), that is given $\mathcal{H}_t$ at every step, and outputs a choice of k centers.

Algorithm 1: $\varepsilon-$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

We present our main theorem and provide proof later on.

Theorem 7.1. *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.

7.1 Incremental Coreset

Section 7.1 is the result of the work of Cohen-Addad solely, and is attached here for completeness. For the complete chapter, see Appendix A. The *Incremental Coreset* algorithm 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 the A.

Lemma 7.2. *For any time step t, the algorithm described in A.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 $\mathcal{Q}_t \subseteq \mathcal{Q}_{t+1}$.

7.2 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 $\mathcal{H}_t$.

7.2.1 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.

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

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 7.3. *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 7.4. *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 (7.3) along with the fact that the*

algorithm does not refine regions with diameter smaller than δ_T give us that

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

Lemma 7.5. *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 $N(9\sqrt{d}/\varepsilon_{\mathsf{hrd}})^d \log(T^3)$.*

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. □

Corollary 7.6. *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 (7.5). 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 7.7. *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.

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

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

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

□

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 (7.7) to these approximate centers, and summing over t.

Lemma 7.8. *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 (7.4) 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 (7.8) we get the following corollary.

Corollary 7.9. *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 $X_{1:t}$*

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

7.3 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 [4], 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. [4], 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 7.10 (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 $\mathcal{M}(v) = \sum_{v' \in \tilde{V}} \mathcal{M}(v')$*

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. □

We move on to bound the regret of the algorithm.

Theorem 7.11. *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*

$$\sqrt{-Tln(\mathcal{M}(p))}$$

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

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 (7.10). Following the proof of Theorem (2.1) of Arora et al. [4], 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 [4] 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. □

Corollary 7.12. *MTMW for a loss function bounded by d obtains a regret of*

$$d\sqrt{-Tln(\mathcal{M}(p))}$$

by using a normalized loss function

7.4 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 (7.9). 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 (7.1).

Lemma 7.13. *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*

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

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

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

As the loss is nonnegative

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

Using the coreset property

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

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 (7.8) and get

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

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 (7.4) we have that $T_0 \leq k \cdot \Lambda$, which finishes the proof. □

Consider the region tree structure $T_{\mathcal{H}_t}$ described in 7.2 that corresponds to the Hierarchical Region Decomposition $\mathcal{H}_t$ at step t defined in (7.13). 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 (7.13) 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 7.14. *Using the definitions from Lemmas (7.4), (7.6) we have that* - $\ln(\mathcal{M}(p^*)) \leq k^2 \cdot \Lambda \cdot \beta$

Proof. Using Lemma (7.4) 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. □

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, we can now prove Theorem (7.1)

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 $\varepsilon/k\Lambda$

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

As stated in Corollary (7.6), for sufficiently large T we have $\beta < d \cdot \Lambda$, hence, along with Lemma (7.14), 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 (7.13) 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 (7.5) 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

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

□

8
Adaptive MWUA

Contents

8.1 Incremental Expert Sets 45
8.1.1 Notation 46
8.1.2 Regret 47
8.2 MTMW for Refining Equivalence Classes 49

In the previous chapter we have presented a modification of MWUA that have enabled the expert set to evolve while keeping the regret low. Any adaptive algorithm that would like to facilitate MWUA for regret minimization encounters this difficulty, hence in this chapter we will present two methods of doing so in this level of abstraction, without referencing clustering or any geometric context. The algorithms differ in the method and the bounds on the adaptive expert set required to ensure low regret.

8.1 Incremental Expert Sets

The first approach assumes that one has a monotone sequence of expert sets, i.e. if $\{Q_t\}_{t=1}^T$ is the sequence of expert sets considered for step t, then for each time step t we have that $Q_t \subseteq Q_{t+1}$. For adaptive algorithms, this corresponds to an initial guess of the experts that should be considered, and as data is revealed and the guess is updated in an increment only fashion. We would like to obtain a regret

Algorithm 3: Incremental Expert Set

Input: $x_1, \ldots, x_T$
Input: $Q_1, \ldots, Q_T$
$t \leftarrow 1$;
while $t \leq T$ **do**
  for $i \in Q_t$ **do**
    $w_i^{(t)} \leftarrow \exp\left(-\eta L_i^{(t-1)}\right)$
  end
  Sample i from Q_t according to $w_i^{(t)}$;
  Output $f_i^{(t)}$ as prediction for step t;
  Receive x_t, Q_{t+1};
  Incur loss $\ell(f_i^{(t)}, x_t)$;
  $t \leftarrow t + 1$;
end

bound with respect to the final set Q_T.

At this point we encounter the question of assigning weights to the new experts. A simple approach would be to assign these experts the weight they would have had if they have been considered since the beginning of the algorithm. For simplicity, we will provide a proof for the algorithm presented in (3) that uses the *Exponential Weights Algorithm*, also referred to as the *Hedge* Algorithm.

8.1.1 Notation

An expert is characterized by an index, hence we define Q_t to be a set of natural numbers. The prediction, or advice, expert i gave at step t will be denoted as $f_i^{(t)}$ and assume it is contained in $[-1, 1]$. Let $\ell(\cdot, \cdot)$ a convex loss function such that $\ell(f, x)$ is the loss an expert incurred for predicting f and then witnessing x. Let $w_i^{(t)}$ be the weight of expert i at step t, just before receiving the t^{th} point. For convenience we denote

$$Q_{t_1 \setminus t_2} = Q_{t_1} \setminus Q_{t_2} \quad Q_{T+1} = Q_T \quad W(Q, t) = \sum_{i \in Q} w_i^{(t)} \quad p_i^{(t)} = w_i^{(t)} / W(Q_t, t)$$

Furthermore, we denote the distribution over experts in Q_t that corresponds to $p_i^{(t)}$ as $\mathbf{p}^{(t)}$ and the loss of such a distribution

$$\ell(\mathbf{p}^{(t)}, x_t) = \mathop{\mathbb{E}}_{i\sim\mathbf{p}^{(t)}}[\ell(f_i^{(t)}, x_t)]$$

Let $L_i^{(t)}$ denote the sum over the loss expert i incurred from the beginning of time up to step t, including t, irregardless of when it was introduced in to the expert set. Specifically $L_i^{(T)}$ is the total loss the expert incurs for the stream, and we will use $L_i^{(0)} = 0$. Lastly, let L_t denote the expected loss the algorithm incurred up to step t

$$L_t = \sum_{\tau=1}^{t} \ell(\mathbf{p}^{(\tau)}, x_\tau)$$

The regret can now be defined as $L_T - \min_{i\in Q_T} L_i^{(T)}$. We can now move on to prove a regret bound for this algorithm.

8.1.2 Regret

Theorem 8.1. *Algorithm (3) has an expected regret of at most*

$$\left(2 + \sum_{t=1}^{T} \frac{W(Q_{t+1\setminus t}, t+1)}{W(Q_t, t+1)}\right)\sqrt{\frac{T}{8}\ln\left(|Q_1|\right)}$$

Proof. We follow the proof presented by Rakhlin [42]. We first notice

$$\begin{aligned}\ln\left(W(Q_{T+1}, T+1)\right) &= \ln\sum_{i\in Q_T}\exp(-\eta L_i^{(T)}) \geq \\ &\geq \ln\max_{i\in Q_T}\exp(-\eta L_i^{(T)}) = -\eta\min_{i\in Q_T} L_i^{(T)}\end{aligned}$$

Subtracting $\ln\left(W(Q_1, 1)\right)$ from both side and using $Q_T = Q_{T+1}$ gives

$$\ln\left(\frac{W(Q_{T+1}, T+1)}{W(Q_1, 1)}\right) \geq -\eta\min_{i\in Q_T} L_i^{(T)} - \ln\left(|Q_1|\right)$$

Before moving to the other side of the inequality we notice the following identity

$$\begin{aligned}\ln\left(\frac{W(Q_{t+1}, t+1)}{W(Q_t, t)}\right) &= \ln\left(\frac{W(Q_{t+1}, t+1)}{W(Q_t, t+1)}\frac{W(Q_t, t+1)}{W(Q_t, t)}\right) \\ &= \ln\left(\frac{W(Q_{t+1}, t+1)}{W(Q_t, t+1)}\right) + \ln\left(\frac{W(Q_t, t+1)}{W(Q_t, t)}\right)\end{aligned}$$

Using Theorem (2.1) of [42] yields the following, up to different notations we use

$$\ln\left(\frac{W(Q_t,t+1)}{W(Q_t,t)}\right) \leq -\eta\cdot\ell(\mathbf{p}^{(t)},x_t)+\frac{\eta^2}{8}$$

Combining the last two equations yields

$$\ln\left(\frac{W(Q_{t+1},t+1)}{W(Q_t,t)}\right) \leq \ln\left(\frac{W(Q_{t+1},t+1)}{W(Q_t,t+1)}\right)-\eta\cdot\ell(\mathbf{p}^{(t)},x_t)+\frac{\eta^2}{8}$$

By summing over all the time steps yields

$$\ln\left(\frac{W(Q_{T+1},T+1)}{W(Q_1,1)}\right) \leq \sum_{t=1}^{T}\ln\left(\frac{W(Q_{t+1},t+1)}{W(Q_t,t+1)}\right)-\eta\cdot L_T+\frac{T\cdot\eta^2}{8}$$

Combining with the lower bound we get

$$-\eta\min_{i\in Q_T}L_i^{(T)}-\ln\left(|Q_1|\right) \leq \sum_{t=1}^{T}\ln\left(\frac{W(Q_{t+1},t+1)}{W(Q_t,t+1)}\right)-\eta\cdot L_T+\frac{T\cdot\eta^2}{8}$$

Which translates to a regret of

$$L_T-\min_{i\in Q_T}L_i^{(T)} \leq \frac{1}{\eta}\ln\left(|Q_1|\right)+\frac{T\cdot\eta}{8}+\frac{1}{\eta}\sum_{t=1}^{T}\ln\left(\frac{W(Q_{t+1},t+1)}{W(Q_t,t+1)}\right)$$

We can write the regret another way using

$$\ln\left(\frac{W(Q_{t+1},t+1)}{W(Q_t,t+1)}\right)=\ln\left(1+\frac{W(Q_{t+1\setminus t},t+1)}{W(Q_t,t+1)}\right)\leq\frac{W(Q_{t+1\setminus t},t+1)}{W(Q_t,t+1)}$$

Which gives

$$L_T-\min_{i\in Q_T}L_i^{(T)} \leq \frac{1}{\eta}\ln\left(|Q_1|\right)+\frac{T\cdot\eta}{8}+\frac{1}{\eta}\sum_{t=1}^{T}\frac{W(Q_{t+1\setminus t},t+1)}{W(Q_t,t+1)}$$

Choosing $\eta=\sqrt{\frac{8\ln(|Q_1|)}{T}}$ finishes the proof. □

We can interpret the new term in the regret bound, namely, the sum of weight ratios, as meaning that the more experts that have performed well historically one adds, the larger the regret will be. Scaling the loss by a constant will allow loss functions with arbitrary constant bounds, and as mentioned earlier, moving to the simplex that corresponds to distributions over experts admits a convex loss function even when the original loss in not convex in the space of predictions.

8.2 MTMW for Refining Equivalence Classes

We will now present an interesting perspective of the MTMW algorithm presented in chapter 7. The algorithm starts with a fixed, final set of experts Q of unknown size, possibly infinite, rather than the aforementioned diverging expert tree. At any time step t we have an equivalence relation $\simeq_t$ over the experts in Q

$$\forall q_1, q_2 \in Q, t \in \mathbb{N}: \quad q_1 \simeq_t q_2 \iff \left(\forall \tau \leq t \quad f_{q_1}^{(t)} = f_{q_2}^{(t)}\right)$$

These equivalence relations partition the experts in Q to classes according to their historical predictions. For ease of notation we define $\simeq_0$ as an equivalence relation that equates all the elements of Q. This sequence of equivalence classes is only being refined, due to the nature of $\simeq_t$. Formally, $\forall q \in Q, t > 0$ let $[q]_{\simeq_t}$ denote the equivalence class of q that corresponds to $\simeq_t$. Then we have that $[q]_{\simeq_t} \subseteq [q]_{\simeq_{t-1}}$. Using this property one can define a tree graph, where the vertices in depth t are all the equivalence classes of $\simeq_t$ and a directed edge connects a class $[q_1]_{\simeq_t}$ to $[q_2]_{\simeq_t}$ if and only if $[q_2]_{\simeq_t} \subseteq [q_1]_{\simeq_t}$. Every vertex of the tree in depth t can be associated with the prediction all the experts in its corresponding equivalence class gave at that step. We will refer this this prediction as *the prediction of the vertex*.

The bound the algorithm obtains is for the regret with respect to the best expert in Q denoted $q^* \in Q$, which corresponds to some leaf of the final tree, namely, $[q^*]_{\simeq_T}$. The predictions that the experts in $[q^*]_{\simeq_T}$ gave at each time step correspond to the prediction of the vertices along the path that connects the root to this leaf.

9 Conclusion

Contents

9.1 Summary 50
9.2 Comparison with Previous Work 52
9.3 Extensions 54
9.3.1 Approximate Regret Lower Bound 54
9.3.2 Simpler Approximate Regret Algorithm 54
9.3.3 Applying the Johnson Lindenstrauss Transfomation . . . 54
9.3.4 Logarithmic Regret via Stochastic Stability 55

The following chapter summarizes the main results, compares this work to the previous work described in detail on Chapter 2, and describes possible extensions.

9.1 Summary

We have examined the online clustering problem using the k-means objective where an algorithm allocates k cluster centers in a unit box in $\mathbb{R}^d$ at each step, incurring a loss according to the squared distance between the next point and the closest cluster center, as defined by Dasgupta [1]. Dasgupta has suggested analyzing algorithms in the *regret* framework, and we have shown why this is the correct choice– no algorithm can obtain any finite competitive ratio. In this setting, the algorithm should minimize the regret with respect to the best k-means solution in hindsight. Our analysis focused on expected regret, without trying to prove corresponding results *with high probability*.

We have presented an online-to-offline reduction which implies that there exists no efficient algorithm with sublinear regret even in the Euclidean setting, under typical complexity theory assumptions, using the hardness of approximation results of Awasthi et al. [2] for $k \geq 2$. Despite the obvious similarities between the problems, there are settings in which the online problem is tractable while the offline version is NP-Hard, e.g. Hazan et al. [43]. One should note that the online algorithm is allowed to change cluster centers at each time step– the optimal online solution predicts precisely the next point at each step, hence differs from that of the offline version.

In order to circumvent the hard lower bound, and on the other hand to examine simple textbook algorithms, we turned to analyze *Follow The Leader* (FTL), which uses a blackbox to obtain the optimal solution to the stream up to step t. We claimed that as the exact k-means problem is NP-Hard, FTL is not constrained by the lower bound. We have shown that nonetheless it has linear regret in the worst case, analytically and experimentally, suggesting it should not be the algorithm of choice. On the other hand we presented experimental results showing it has logarithmic regret on both synthetic and real datasets, namely, different permutations of MNIST and Gaussian Mixture Models, Suggesting FTL works well for some natural datasets. We presented an initial stability analysis that may provide the framework to explain these results, and suggests that for streams that have approximately equal sized clusters, in terms of ratio of points per cluster, and are well separated, FTL obtains logarithmic regret. We have shown this by reducing the online clustering with k clusters to k distinct $k = 1$ instances of the online clustering problem, which was shown to admit low regret in Shalev-Shwartz et al. [8], using FTL. Our analysis suggests that for non worst case data FTL should be the algorithm of choice for further investigation.

Aiming to measure the information theoretic constraints imposed on the online k-means problem if complexity constraints are removed we turned to analyze exponential time algorithms, providing upper bounds for regret minimization. We examined the *Multiplicative Weight Update Algorithm* (MWUA) applied to a grid discretization of space with a resolution which is polynomial in $1/T$. We have shown that one can

achieves $\tilde{O}\left(T^{1-2\alpha}\right)$ regret, with a runtime of $T^{\alpha kd}$ for $\alpha \in (0, 1/4)$, which demonstrates the trade-off between regret and runtime for this approachby using different grid resolution. Hence we conclude that computational complexity is the main challenge in devising low-regret algorithms, as opposed to information theoretic constraints.

Then we turned to the approximate regret minimization framework, where we presented an algorithm that obtains $\tilde{O}(\sqrt{T})$ approximate regret with a computational complexity of $O(T(k^2\varepsilon^{-2}d^{1/2}\log(T))^{k(d+O(1))})$. In order to achieve this we use a novel incremental coreset construction (due to Cohen-Addad), and an adaptive version of MWUA coined *Mass Tree Multiplicative Weights Update Algorithm* (MTMW) that allows for new experts to be introduced, as long as they conform to some tree structure. We provided an adaptive Hierarchical Region Decomposition of space and proved structural properties for it to satisfy the low regret requirements of MTMW. We combined the Region Decomposition and the coreset construction and proved k-means approximation properties of the decomposition, relating the regret guarantees w.r.t. the best expert of MTMW to the best offline solution. The combination of all these along with a clever choice of parameters obtained the aforementioned complexity and approximate regret. This is a complicated algorithm to implement, and is still exponential in k, d, so despite having a $log(T)$ base rather than T, it mainly provides a theoretical upper bound for the approximate regret minimization problem.

Lastly, we presented an additional method to facilitate MWUA for adaptive algorithms and provided a generalized perspective for MTMW , that allows relating the tree structured defined for MTMW to a broad range of settings.

9.2 Comparison with Previous Work

This is the first work that aimed to give a comprehensive theoretical understanding of the online k-means clustering problem. Other works mostly dealt with different problem definitions. A large corpus of work exists addressing the streaming k-means problem which reads the data as a stream and aims to output an approximate solution to the offline k-means problem at the end of the stream. The works that

relates the most to this dissertation is described in Chapter 2, hence we will review the most similar works and where we differ.

Liberty et al. [36] and Mansfield et al. [37] dealt with labelling points in an online manner, but incur the k-means loss w.r.t. the centroids of the final clusters, either using $O(k)$ clusters [36], and assuming strict bayesian assumptions on the data stream in [37]. Vassilvitskii and Lattanzi [38] count the amount of reclustering operations needed to maintain a constant approximation of the optimal solution at any point in time. The online facility location problem is related to our setting, which is dealt by Meyerson [33], where an algorithm must serve the next point in the stream using the closest facility. When the service cost is the k-means objective, the difference between the settings rises from the fact that allocating a facility incurs a cost, the facility cannot be moved, and the amount of facilities is not limited. one can say that the online clustering problem is a facility location problem where the facilities are mobile, bounded by k, and are cheap compared to the service cost, as in minimizing mortality for a fixed set of mobile health related facilities given as foreign aid, e.g. defibrillators.

As mentioned, Dasgupta [1] first presented the problem formulation we use, and performs a preliminary investigation, presenting a naive efficient algorithm, with no analysis. Our lower bound addresses the open problem raised by Dasgupta, and settles it for efficient algorithms– one cannot obtain sublinear regret efficiently.

Choromanska and Monteleoni [41] present an approximate regret minimization algorithm for an almost identical setting to the online clustering problem presented here, with weaker constraints. They use auxiliary batch clustering algorithms (*experts*) that report the cluster center closest to the next point in the stream at each step. When the experts have approximation guarantee w.r.t. the optimal clustering of the batch, they obtain regret guarantees relating to the batch size and approximation guarantee. Our results considered the problem where no implicit information on the next point is available to the algorithm at the time it commits to the next set of k cluster centers.

Li et al. [40] handle the online clustering problem where the algorithm may use $O(k)$ cluster centers at each step and the comparison is to the best k centers in hindsight. They provide a regret minimization algorithm, based on Gibbs Sampling, and present a Reversible Jump MCMC method without providing theoretical mixing time analysis.

9.3 Extensions

9.3.1 Approximate Regret Lower Bound

Although the research was concluded with a decent understanding of the theoretical limitations of the regret minimization problem for online clustering, approximate regret remains partially open– a lower bound for approximate regret will be a complementary result for any upper bound obtained. A straightforward approach will include another attempt of the online-to-offline reduction, this time for sublinear approximate regret, rather than regret.

9.3.2 Simpler Approximate Regret Algorithm

We have provided a proof that shows that FTL is bad in the worst case, but not for its randomized or regularized variants, FTPL and FTRL. A nice extension would be finding a regularizer that promises low regret for FTRL, or a perturbation that promises similar results for FTPL.

9.3.3 Applying the Johnson Lindenstrauss Transfomation

Our approximate regret algorithm has exponential dependency in both k and d. An approach we failed to implement is applying the Johnson-Lindenstrauss dimensionality reduction (JL), which preserved the distance of pair of points up to a $(1+\epsilon)$ factor for a reduced dimension that has logarithmic dependency on the amount of points [44]. Makarychev et al. [45] show that in order to preserve the k-means loss, one may reduce to a dimension that relates only to k, irregardless of the amount of points. The two flavours that may be considered are either projecting the entire stream to a dimension

of $O(\log(k/\epsilon)/\epsilon^2)$ only, or using the standard JL to project the coreset to a dimension which is some function of $loglog(T), k, \epsilon$. Both approaches eliminate the dependency of the exponent on the dimension. The major problem we had with this approach is how to translate the candidate cluster centers in the reduced space to the non reduced space, while preserving some approximation guarantee on their loss. JL fails to relate the loss of candidate cluster centers that are not centroids of some partition of the data points, while keeping the dimension low. A promising approach will be to modify the adaptive Hierarchical Region Decomposition to one whose induced approximate centers are always centroids of partitions of the data (or the coreset).

Our most promising lead considers all the centroids of valid k-means partitions of the data. This approach goes along well with the coreset method and the JL reduction. Furthermore, Inaba et al. [17] showed upper bounds on the amount of valid k-means partitions which is polynomial in the amount of points, rather than exponential, which indicates that this is a promising lead.

9.3.4 Logarithmic Regret via Stochastic Stability

The stability constraints described here can be relaxed to extend the logarithmic regret analysis applicability to diverse settings. An essential change that may involve only slight modifications is the framework of stochastic streams, with a probabilistic notion of *balanced* and *well separated* conditions described in this work. An ideal model will apply to the Gaussian Mixture Models (GMM) we used, whose support encompasses the entire space, which stands in contradiction to the well separation requirements currently used. A viable option for analysis is examining the FTL algorithm for $k = 1$ in the presence of noise, i.e. points that are not considered when calculating the optimal solution in hindsight, but are given to FTL nonetheless.

This may enable the following analysis for a stream that is generated by repeatedly sampling for some fixed distribution $\mathcal{D}$ (be it GMM or any other). Consider C^*

the optimal centers for the infinite stream [1], i.e. $C^* = \arg\min_c \lim_{T\to\infty} \frac{\ell(c,X_{1:T})}{T}$. After $c \cdot \log(T)$ time steps, for some distribution dependent constant c, the optimal clustering for the stream up to t should be very close to C^* with high probability. We can partition the space to a *fuzzy Voronoi Diagram* which will correspond to regions that after enough points should not be mislabelled with high probability, and *fuzzy* regions around the edges of the Voronoi Diagram of C^* that correspond to the regions of points that may be mislabelled. If one can consider the points in the fuzzy regions as noise, and show that it is small enough to not shift the k centers too much, disrupting the fuzzy voronoi diagram, then as time goes on the center will converge to C^*, as it is an estimator for the mean of $\mathcal{D}$ restricted to that Voronoi Cell using IID samples, and the fuzzy regions will shrink with them.

[1] These may not coincide with the underlying means of mixture models, but for mixture models that are well separated under some definition, they should be approximately the same

Appendices

A

Incremental Coreset

Contents

A.1 Incremental Coreset 58
A.1.1 Maintaining an $O(1)$-Approximation for k-means 59
A.1.2 Maintaining a Coreset for k-means 59

This chapter is the result of the work of Vincent Viallat Cohen-Addad solely, and is attached for mathematical completeness of the proofs in Chapter 7.

A.1 Incremental Coreset

The *Incremental Coreset* algorithm described 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 ε_{c}-coreset for $X_{1:t}$, for some given parameter $\varepsilon_{\mathsf{c}} > 0$. Formally, we have the following lemma, whose proof is provided at the end of this subsection.

Lemma A.1. *Consider a time t, and the set $\{S_1^{(t)}, \ldots, S_s^{(t)}\}$, each a set of candidate cluster centers, maintained by $\mathcal{A}_{cop}$. Then the algorithm described in A.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 $\mathcal{Q}_t \subseteq \mathcal{Q}_{t+1}$.*

A.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 [46], which we call $\mathcal{A}_{cop}$, and whose performance guarantees are summarized by the following proposition, that follows immediately from Charikar et al. [46].

Theorem A.2 (Combination of Lemma 1 and Corollary 1 in [46])**.** *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.*

A.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 [31] whose properties are summarized in the following theorem.

Theorem A.3 (Thm. 5.5/3.6 in Chen [31])**.** *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 A.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 $\mathcal{Q}_t = \bigcup_i A(S_i^{(t)})$.

We can now provide proof for Lemma (A.1)

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 (A.2) 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-means problem. 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 (A.3), 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.

□

Bibliography

[1] 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.

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

[3] Anna R Karlin, Mark S Manasse, Larry Rudolph, and Daniel D Sleator. Competitive snoopy caching. *Algorithmica*, 3(1-4):79–119, 1988.

[4] 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.

[5] Shai Shalev-Shwartz and Sham M Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *Advances in Neural Information Processing Systems*, pages 1457–1464, 2009.

[6] Ofer Dekel. Course notes, cse599s: Online learning, spring 2014 university of washington. lecture 6: Online learning with expert advice, 2014. URL https://courses.cs.washington.edu/courses/cse599s/14sp/scribes/lecture6/lecture6.pdf.

[7] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

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

[9] H Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. 2011.

[10] Michael D. Grigoriadis and Leonid G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53 – 58, 1995. ISSN 0167-6377. doi: https://doi.org/10.1016/0167-6377(95)00032-0. URL http://www.sciencedirect.com/science/article/pii/0167637795000320.

[11] N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, Santa Cruz, CA, USA, 1989. UMI Order No: GAX89-26506.

[12] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55 (1):119–139, 1997.

[13] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 300–309, 1998.

[14] Serge A Plotkin, David B Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20 (2):257–301, 1995.

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

[16] Sanjoy Dasgupta. *The hardness of k-means clustering*.

[17] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 332–339. ACM, 1994.

[18] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, February 2010. ISSN 0004-5411. doi: 10.1145/1667053.1667054. URL http://doi.acm.org/10.1145/1667053.1667054.

[19] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1434–1453. Society for Industrial and Applied Mathematics, 2013.

[20] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight fpt approximations for k-median and k-means. *arXiv preprint arXiv:1904.12334*, 2019.

[21] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a ptas for k-means in doubling metrics. *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct 2016. doi: 10.1109/focs.2016.47. URL http://dx.doi.org/10.1109/FOCS.2016.47.

[22] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct 2016. doi: 10.1109/focs.2016.46. URL http://dx.doi.org/10.1109/FOCS.2016.46.

[23] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[24] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 600–607. ACM, 2002.

[25] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? 2006.

[26] David Arthur, Bodo Manthey, and Heiko Röglin. k-means has polynomial smoothed complexity. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 405–414. IEEE, 2009.

[27] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[28] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 15–28. Springer, 2009.

[29] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7): 622–633, 2012.

[30] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

[31] 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.

[32] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004.

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

[34] Mark S Manasse, Lyle A McGeoch, and Daniel D Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[35] Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009.

[36] 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.

[37] Philip Andrew Mansfield, Quan Wang, Carlton Downey, Li Wan, and Ignacio Lopez Moreno. Links: A High-Dimensional Online Clustering Method. *arXiv e-prints*, art. arXiv:1801.10123, Jan 2018.

[38] Sergei Vassilvitskii and Silvio Lattanzi. Consistent k-clustering. 2017.

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

[40] Le Li, Benjamin Guedj, Sébastien Loustau, et al. A quasi-bayesian perspective to online clustering. *Electronic journal of statistics*, 12(2):3071–3113, 2018.

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

[42] Sasha Rakhlin. Course notes mit 9.520, online learning spring 2008. lecture 9: Prediction with expert advice, 2008. URL http://www.mit.edu/~9.520/spring08/Classes/online_learning_2008.pdf.

[43] Elad Hazan, Satyen Kale, and Shai Shalev-Shwartz. Near-optimal algorithms for online matrix prediction. In *Conference on Learning Theory*, pages 38–1, 2012.

[44] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[45] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 1027–1038, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6705-9. doi: 10.1145/3313276.3316350. URL http://doi.acm.org/10.1145/3313276.3316350.

[46] 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.