Online Clustering with Experts

Anna Choromanska
Columbia University

Claire Monteleoni
George Washington University

Abstract

Approximating the k-means clustering objective with an online learning algorithm is an open problem. We introduce a family of online clustering algorithms by extending algorithms for online supervised learning, with access to expert predictors, to the unsupervised learning setting. Instead of computing prediction errors in order to re-weight the experts, the algorithms compute an approximation to the current value of the k-means objective obtained by each expert.

When the experts are batch clustering algorithms with b-approximation guarantees with respect to the k-means objective (for example, the k-means++ or k-means# algorithms), applied to a sliding window of the data stream, our algorithms obtain approximation guarantees with respect to the k-means objective. The form of these online clustering approximation guarantees is novel, and extends an evaluation framework proposed by Dasgupta as an analog to regret. Notably, our approximation bounds are with respect to the optimal k-means cost on the *entire* data stream seen so far, even though the algorithm is online. Our algorithm's empirical performance tracks that of the best clustering algorithm in its expert set.

1 Introduction

Practical algorithms for *clustering data streams* would be useful in a wide range of applications, such as topic detection in streaming media, and clustering weather patterns to detect extreme events such as cyclones and heat waves.

The *online learning* setting is applicable to a variety of data stream problems including forecasting, real-time decision making, and resource-constrained learning. Data streams can take many forms, such as stock prices, weather measurements, and internet transactions, or any data set so large compared to computational resources, that algorithms must access it sequentially. In the online learning model, an algorithm makes only one pass of an endless data stream.

Most data sources produce raw data (*e.g.* speech signal, or images on the web), that is not yet labeled for any classification task, which motivates the study of *unsupervised learning. Clustering* refers to a broad class of unsupervised learning tasks aimed at partitioning the data into clusters that are appropriate to the specific application. Clustering techniques are widely used in practice, in order to summarize large quantities of data (*e.g.* aggregating similar online news stories), however their outputs can be hard to evaluate. For any particular application, a domain expert may be useful in judging the quality of a resulting clustering, however having a human in the loop may be undesirable. Probabilistic assumptions have often been employed to analyze clustering algorithms, for example i.i.d. data, or further, that the data is generated by a well-separated mixture of Gaussians.

Without any distributional assumptions on the data, one way to analyze clustering algorithms is to formulate some objective function, and then to prove that the clustering algorithm either optimizes it, or is an approximation algorithm. Approximation guarantees, with respect to some reasonable objective, are therefore desirable. The k-means objective is a simple, intuitive, and widely-cited clustering objective, however few algorithms provably approximate it, even in the batch setting. In this work, inspired by an open problem posed by Dasgupta [15], our goal is to approximate the k-means objective in the online setting.

1.1 The k-means clustering objective

One of the most widely-cited clustering objectives for data in Euclidean space is the k-means objective. For a finite set, S, of n points in $\mathbb{R}^d$, and a fixed positive

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 22 of JMLR: W&CP 22. Copyright 2012 by the authors.

integer, k, the k-means objective is to choose a set of k cluster centers, C in $\mathbb{R}^d$, to minimize:

$$\Phi_X(C) = \sum_{x \in S} \min_{c \in C} \|x - c\|^2$$

which we refer to as the "k-means cost" of C on X. This objective formalizes an intuitive measure of goodness for a clustering of points in Euclidean space. Optimizing the k-means objective is known to be NP-hard, even for $k = 2$ [4]. Therefore the goal is to design *approximation algorithms.*

Definition 1. *A* **b-approximate** *k-means clustering algorithm, for a fixed constant b, on any input data set X, returns a clustering C such that: $\Phi_X(C) \leq b \cdot OPT_X$, where OPT_X is the optimum of the k-means objective on data set X.*

Definition 2. *An* **(a, b)-approximate** *k-means clustering algorithm, is a b-approximation algorithm that returns at most $a \cdot k$ centers.*

Surprisingly few algorithms have approximation guarantees with respect to k-means, even in the batch setting. Even the algorithm known as "k-means" does not have an approximation guarantee.

Our contribution is a family of online clustering algorithms, with regret bounds, and approximation guarantees with respect to the k-means objective, of a novel form for the online clustering setting. Notably, our approximation bounds are with respect to the optimal k-means cost on the *entire* data stream seen so far, even though the algorithm is online.

We extend algorithms from [19] and [25] to the unsupervised learning setting and introduce a flexible framework in which our algorithms take a set of candidate clustering algorithms, as experts, and track the performance of the "best" expert, or best sequence of experts, for the data. Instead of computing prediction errors in order to re-weight the experts, the algorithms compute an approximation to the current value of the k-means objective obtained by each expert. Our approach lends itself to settings in which the user is unsure of which clustering algorithm to use for a given data stream, and exploits the performance advantages of any batch clustering algorithms used as experts. Our algorithms vary in their models of the time-varying nature of the data; we demonstrate encouraging performance on a variety of data sets.

1.2 Related work

The widely used "k-means algorithm," is a batch clustering algorithm that can be viewed as a hard-assignment variant of Expectation-Maximization (EM). To avoid confusion with the k-means objective, we refer to it as Lloyd's algorithm [23]. While it typically (but not always [28]) converges quickly, its solution has not been shown to approximate the k-means objective. Much clustering research involves assumptions on the data to be clustered, such as i.i.d. data, or data that admits a clustering with well separated means *e.g.* [14, 13].[1] Another analysis model assumes a "target" clustering for the specific application and data set, that is close to any constant approximation of the clustering objective [7]. In contrast, our analyses require no distributional assumptions.

There exist some batch clustering algorithms with approximation guarantees with respect to the k-means objective; part of our analysis exploits this. Some examples include k-means++, an algorithm that approximates the k-means objective, by a factor of $O(\log k)$ [5]. Constant approximations have been shown for a local search technique [20], the k-means# algorithm [3], which outputs $O(k \log k)$ centers, and the work of [2], which outputs $O(k)$ centers. Several works have studied clustering *finite* data streams [17, 3, 1]. Work by [3] does so with respect to the k-means objective, and extends a streaming clustering algorithm ([17]) for the k-medoid objective (also known as k-median), which is to minimize the sum of distances, in a general metric space, of the points to their closest centers, using a subset of the input points. Facility location, a problem related to k-medoid, has also been studied in the online setting [24].

A number of techniques for online clustering have enjoyed success in practice, such as [21], and variants of EM (e.g. [22]), and some have been analyzed under stochastic assumptions on the data, *e.g.* [11]. Several online clustering algorithms have approximation guarantees with respect to clustering objectives other than k-means. A doubling algorithm due to [12], and the cover tree algorithm of [9], both provide a constant approximation to the k-center objective, which minimizes the maximum distance from an input point to its closest cluster center, in a general metric space.[2] There has also been some work on ensemble methods for clustering, in a batch setting, *e.g.* [16, 27].

We are not aware of previous regret analyses for clustering, other than an analysis of randomized PCA [30], which could be viewed as clustering for the case $k = 1$.

2 Online k-means approximation

Specifying an evaluation framework for online clustering is a challenge. One cannot use a similar analysis

[1] [8] provides an algorithm for an arbitrarily small separation, when the clusters are spherical Gaussians.

[2] Intuitively, this objective is less robust to outliers than k-means, for data in Euclidean space.

setting to [5, 3] in the finite data stream case. With no assumptions on the data, one can always design a sequence of observations that will fool a seeding algorithm (that picks a set of centers and does not update them) into choosing seeds that are arbitrarily bad (with respect to the k-means objective) for some future observations, or else into simply abstaining from choosing seeds.

Our analysis is inpired, in part, by an evaluation framework proposed by Dasgupta as an analog to regret [15]. The *regret* framework, for the analysis of supervised online learning algorithms, evaluates algorithms with respect to their additional prediction loss relative to a hindsight-optimal comparator method. With the goal of analyzing online clustering algorithms, Dasgupta proposed bounding the difference between the cumulative clustering loss since the first observation:

$$L_T(\texttt{alg}) = \sum_{t \leq T} \min_{c \in C_t} \|x_t - c\|^2 \tag{1}$$

where the algorithm outputs a clustering, C_t, before observing the current point, x_t, and the optimal k-means cost on the points seen so far.

Our approach, also inspired by regret analyses, is to instead evaluate the performance of an online clustering algorithm relative to a finite set of batch clustering algorithms, with respect to Equation 1. In our model, the online clustering algorithm has access to a set of (batch) clustering algorithms as "experts." We provide clustering variants of predictors with expert advice from [19] and [25], and analyze them by first bounding this quantity in terms of regret with respect to the cumulative clustering loss of the best expert, or best sequence of experts, computed in hindsight. Then, adding assumptions that the batch clustering algorithms are b-approximate with respect to the k-means objective, we extend our regret bounds to obtain bounds on this quantity with respect to the optimal k-means cost on the points seen so far.

3 Online clustering with experts

Here we extend algorithms from [19] and [25] to the unsupervised learning setting. These supervised online learning algorithms form their predictions on the basis of a set of n "expert" predictors, i, subject to a probability distribution over experts, $p_t(i)$, which is updated dynamically with time, t. In [25], algorithms in [19] were derived as Bayesian updates of an appropriately defined generalized Hidden Markov Model (HMM). The identity of the best performing expert at a given time is the hidden variable. Different update rules for $p_t(i)$ correspond to different transition dynamics, modeling different levels of non-stationarity.

In our setting, the experts output clusterings, and instead of computing prediction errors in order to re-weight the experts, we compute an approximation to the k-means objective obtained by each expert, inspired by Equation 1. We define "loss" and "clustering" functions, unsupervised analogs to "prediction loss," and "prediction." In the clustering setting, the algorithm is not "predicting" the current observation x_t; x_t is in fact used in the clusterings of each of the experts, that inform the current clustering. This is a real-time analog to the standard clustering task, in which the action of the algorithm is not to predict, but to assign x_t to a (dynamic) cluster center. We show that our choice of loss and clustering functions satisfies (c, η)-*realizability*, a condition that allows us to extend regret analyses from [19, 25], for a family of algorithms.

3.1 Preliminaries

Here we define notation. Let k be the desired number of clusters. We index time by t, and let x_t be the most recent observation in the stream. There are n experts, which we index by i. Let C_t^i denote the set of centers output by the i^{th} expert at time t. We denote the center in C_t^i closest to the data point x_t as $c_t^i = \arg\min_{c \in C_t^i} \|x_t - c\|^2$. We use the notation $D(\cdot\|\cdot)$ for the Kullback-Leibler divergence, and $H(\cdot)$ for the entropy.

Definition 3. *The* **loss** *of a center, c_t, output by a clustering algorithm, at time t, is $L(x_t, c_t) = \left\|\frac{x_t - c_t}{2R}\right\|^2$. The normalization factor takes some $R \geq \|x_t\|$ for all t.*

The bound, $\|x_t\| \leq R$, is justified by the fact that any algorithm that can store points, x_t, in memory, has a physical constraint on the size of a word in memory. We assume that R also upper bounds $\|c_t^i\|$ for all i and t. Given the bound on $\|x\|$, this would certainly hold for any reasonable centers. L_t with a single argument computes loss of any clustering algorithm or set of centers, and evaluates to our loss function computed on the closest center to x_t. We refer to cumulative loss over time (from the first observation) of any clustering algorithm as: $L_T = \sum_{t=1}^T L_t$. For the loss on a sequence indexed from time s to t, we use the notation: $L_{<s,t>} = \sum_{t'=s}^t L_{t'}$, and for the loss on a sequence indexed from time $s+1$ to t, we use $L_{(s,t>}$.

The family of algorithms that we introduce differ in their update rules for the distribution over experts, $p_t(i)$. With respect to that distribution, we define the output of our algorithms as follows.

Definition 4. *The* **clustering** *of our algorithm at* *time t, with respect to its current distribution over experts, $p_t(i)$, is the weighted sum of the closest centers to x_t, per expert:* $\texttt{clust}(t) = \sum_{i=1}^{n} p_t(i)c_t^i$.

Note that this is a single center. For any usage in which k (or ak) centers must be output at every time-step, it is equivalent, with respect to all our analyses, for the algorithm to output the center above, in addition to the $k-1$ (or $ak-1$) centers, $C_t^{i^*} - \{c_t^{i^*}\}$, where $i^* = \arg\max_{i\in\{1,\dots,n\}} p_t(i)$, *i.e.* the remaining centers output by the clustering algorithm with the current highest weight. In the experiments, we use this version to evaluate the k-means cost of the algorithm on the entire stream seen so far, however since this does not affect our analysis, we will simply refer to the clustering as defined above.

3.2 Algorithms

Algorithm 1 summarizes our Online Clustering with Experts (OCE) algorithms. We present 3 variants of OCE in Algorithm 1: the Static-Expert algorithm (from [19], with a prior history in the literature), and Fixed-share, introduced by Herbster and Warmuth [19] as a simple way to model time-varying data in the experts setting. We also provide an OCE version of Learn-α, an algorithm introduced by Monteleoni and Jaakkola [25], in the supervised setting, to address the question of how to run Fixed-Share algorithms without the knowledge of the hindsight optimal value of the parameter α. Learn-α learns the parameter online using Static-Expert updates over a set of Fixed-share algorithms, each with a different value of the α parameter; we use the discretization procedure from [25].[3]

We state the algorithms in their most general form, in which the experts are arbitrary black boxes that output a set of cluster centers at each iteration, and filter them to just return the center closest to x_t, the current observation. The OCE algorithm views only this vector of n centers, c, before producing its output. Then, to compute loss and perform the weight updates, it views x_t, the current observation in the stream. Our regret analyses hold in this setting. When the experts are instantiated as batch clustering algorithms, we denote by W_t a sliding window, of size W, of the stream through and including x_t, on which the experts compute the current clustering. This aligns with approximation guarantees for clustering algorithms in the literature, *i.e.* algorithms cluster an input data set and are then evaluated with respect to the optimum of some objective function, computed on the same data set. Our approximation guarantees hold in this setting, and we show that no b-approximate clustering algorithm can trivially optimize our loss function by outputting x_t. The algorithm also permits the use of (a, b)-approximation algorithms as experts.

Running time analysis. Using a straight-forward analysis of our 3 variants of the OCE algorithm listed in Algorithm 1 yields running-times, per observation, of: $O(dkn)$, $O(dkn + n^2)$, and $O(dkn + m(n + n^2))$, respectively, in addition to the times to run each clustering expert, where n is the number of experts, d is the dimension, k is the desired number of clusters (k is replaced with ak when run with experts that output ak centers), and m is the size of the discretization of α in the Learn-α algorithm. The discretization can be set using the procedure from [25], in which $m = O(\sqrt{T})$ for desired sequence length T. For a strictly online algorithm, a uniform discretization of constant size m can be used instead; we have observed that the choice of discretization does not affect performance by much.

4 Performance guarantees

We will give two types of performance guarantees for our family of online clustering algorithms. First we will provide similar bounds to those in [19, 25] for the setting of supervised online learning with experts. Then we will instantiate the experts as batch clustering algoirthms, with approximation guarantees, to yield online variants of approximation guarantees with respect to the k-means objective. Omitted proofs appear in the appendix. Bounds are not optimized with respect to constants.

4.1 Regret bounds

In order to make use of analysis tools from the literature, we first need to show that for our clustering and loss functions, a certain property holds. Here we reformulate the definition of (c, η)-**realizability** presented in [19], and due to [18, 29], and demonstrate that it holds in our setting.

Theorem 1. *The loss function L defined in Definition 3 and the clustering function defined in Definition 4 are $(2, \frac{1}{2})$-**realizable**, i.e.: $L(x_t, \texttt{clust}(t)) \leq -2\log\sum_{i=1}^{n} p(i)e^{-\frac{1}{2}L(x_t, c_t^i)}$ for all $n \in \mathbb{N}$ and all x_t and c_t^i such that $\|x_t\| \leq R$, $\|c_t^i\| \leq R$, and all stochastic vectors $p \in [0\,1]^n$.*

Using this, we can now provide regret bounds for a family of online clustering algorithms. For our OCE algorithm's Static-Expert variant, shown in Algorithm 1, we have the following bound.

Theorem 2. *Given any sequence of length T, let a_i* be the best expert in hindsight (with respect to cumulative loss on the sequence). Then the cumulative loss of*

[3] We also have analyses for a generalization of Fixed-Share, with arbitrary transition dynamics, studied in [25].

Algorithm 1 `OCE: Online Clustering with Experts` (3 variants)

```
INPUTS: Clustering algorithms $\{a_1, a_2, \ldots, a_n\}$, R: large constant, Fixed-Share only: $\alpha \in [0,1]$,
Learn-$\alpha$ only: $\{\alpha_1, \alpha_2, \ldots, \alpha_m\} \in [0,1]^m$
INITIALIZATION: $t = 1$, $p_1(i) = 1/n, \forall i \in \{1, \ldots, n\}$, Learn-$\alpha$ only: $p_1(j) = 1/m, \forall j \in \{1, \ldots, m\}$,
$p_{1,j}(i) = 1/n, \forall j \in \{1, \ldots, m\}, \forall i \in \{1, \ldots, n\}$
AT $t$-TH ITERATION:
Receive vector $c$, where $c^i$ is the output of algorithm $a_i$ at time $t$.
clust$(t) = \sum_{i=1}^{n} p_t(i) c^i$          // Learn-$\alpha$:  clust$(t) = \sum_{j=1}^{m} p_t(j) \sum_{i=1}^{n} p_{t,j}(i) c^i$
OUTPUT: clust$(t)$                      // Optional: additionally output $C^{i^*} - \{c^{i^*}\}$, where $i^* = \arg\max_{i \in \{1,\ldots,n\}} p_t(i)$
View $x_t$ in the stream.
For each $i \in \{1, \ldots, n\}$:
    $L(i,t) = \| \frac{x_t - c^i}{2R} \|^2$
---------------------------------------------------------------------------
    $p_{t+1}(i) = p_t(i) e^{-\frac{1}{2} L(i,t)}$                                1. Static-Expert
---------------------------------------------------------------------------
    For each $i \in \{1, \ldots, n\}$:                                          2. Fixed-Share
    $p_{t+1}(i) = \sum_{h=1}^{n} p_t(h) e^{-\frac{1}{2} L(h,t)} P(i \mid h; \alpha)$    // $P(i \mid h; \alpha) = (1-\alpha)$ if $i = h$, $\frac{\alpha}{n-1}$ o.w.
---------------------------------------------------------------------------
    For each $j \in \{1, \ldots, m\}$:                                          3. Learn-$\alpha$
    lossPerAlpha[j] $= -\log \sum_{i=1}^{n} p_{t,j}(i) e^{-\frac{1}{2} L(i,t)}$
    $p_{t+1}(j) = p_t(j) e^{-\text{lossPerAlpha[j]}}$
    For each $i \in \{1, \ldots, n\}$:
        $p_{t+1,j}(i) = \sum_{h=1}^{n} p_{t,j}(h) e^{-\frac{1}{2} L(h,t)} P(i \mid h; \alpha_j)$
        Normalize $p_{t+1,j}$.
---------------------------------------------------------------------------
    Normalize $p_{t+1}$.
    t=t+1
```

the algorithm obeys the following bound, with respect to that of the best expert:

$$L_T(\boldsymbol{alg}) \leq L_T(a_i*) + 2 \log n$$

The proof follows by an almost direct application of an analysis technique of [25]. We now provide regret bounds for our OCE Fixed-Share variant. First, we follow the analysis framework of [19] to bound the regret of the Fixed-Share algorithm (parameterized by α) with respect to the best s-partition of the sequence.[4]

Theorem 3. *For any sequence of length T, and for any $s < T$, consider the best partition, computed in hindsight, of the sequence into s segments, mapping each segment to its best expert. Then, letting $\alpha' = s/(T-1)$: $L_T(\boldsymbol{alg}) \leq L_T(\boldsymbol{best\ s\text{-}partition})$ $+\ 2[\log n + s \log(n-1) + (T-1)(H(\alpha') + D(\alpha' \| \alpha))]$*

We also provide bounds on the regret with respect to the Fixed-Share algorithm running with the hindsight optimal value of α. We extend analyses in [25, 26] to the clustering setting, which includes providing a more general bound in which the weight update over experts is parameterized by an arbitrary transition dynamics; Fixed-Share follows as a special case. Following [26, 25], we will express regret bounds for these algorithms in terms of the following log-loss: $L_t^{log} = -\log \sum_{i=1}^{n} p_t(i) e^{-\frac{1}{2} L(x_t, c_t^i)}$. This log-loss relates our clustering loss (Definition 3) as follows.

Lemma 1. $L_t \leq 2 L_t^{log}$

Proof. By Theorem 1: $L_t = L(x_t, \texttt{clust}(t)) \leq -2 \log \sum_{i=1}^{n} e^{-\frac{1}{2} L(x_t, c_t^i)} = 2 L_t^{log}$ □

Theorem 4. *The cumulative log-loss of a generalized OCE algorithm that performs Bayesian updates with arbitrary transition dynamics, Θ, a matrix where rows, Θ_i, are stochastic vectors specifying an expert's distribution over transitioning to the other experts at the next time step, obeys*[5]*:*

$$L_T^{\log}(\Theta) \leq L_T^{\log}(\Theta^*) + (T-1) \sum_{i=1}^{n} \rho_i^* D(\Theta_i^* \| \Theta_i)$$

where Θ^* is the hindsight optimal (minimizing log-loss)

[4] To evaluate this bound, one must specify s.

[5] As in [26, 25], we express cumulative loss with respect to the algorithms' transition dynamics parameters.

setting of the transition matrix, for the observed sequence, and ρ_i^* is the marginal probability of being in state i, of the hindsight optimal algorithm.

Corollary 1. *For our OCE algorithm's Fixed-share(α) variant:*

$$L_T^{\log}(\alpha) \leq L_T^{\log}(\alpha^*) + (T-1)D(\alpha^* \| \alpha)$$

where α^* is the hindsight optimal setting of the parameter α for the observed sequence.

In the supervised setting, the analogous regret for Fixed-share has a sequence-dependent lower bound which can be linear in T [25]. Now we address the setting in which α^* is not known beforehand, and provide a regret bound for the OCE Learn-α variant.

Theorem 5. *For our OCE algorithm's Learn-α variant, using a discretization of the α parameter, $\{\alpha_j\}$ of size m, where α^* is the hindsight optimal α:*

$$L_T^{\log}(\mathtt{alg}) \leq L_T^{\log}(\alpha^*) + (T-1)\min_{\{\alpha_j\}} D(\alpha^* \| \alpha_j) + \log m$$

The proof uses Corollary 1 and a variant of regret bound for Static-Expert, which is used by Learn-α to update the weights over Fixed-share algorithms. By choice of discretization, $\{\alpha_j\}$, we can control the second term. For example, allowing the discretization to depend on T, [25] optimized their regret bound.

4.2 Approximation guarantees

When the experts are k-means approximation algorithms, we can extend our regret bounds to provide online variants of approximation guarantees for OCE. Now each expert i is a b_i-approximate k-means clustering algorithm, a_i. First we show that no b-approximate clustering algorithm will trivially minimize our clustering loss by always outputting x_t as the current center, to attain zero loss. That is, given b, we provide a sequence of examples such that outputting x_t violates the b-approximation guarantee.

Lemma 2. *If an algorithm is b-approximate with respect to the k-means objective, there exist sequences ending in x_t for which it cannot output x_t as a center.*

We continue by providing the following lemmas, which may also be of general interest.

Lemma 3. *Let OPT_{W_1} be the optimum value of the k-means objective for the data set seen in window W_1, OPT_{W_2} be the optimum value of the k-means objective for the data set seen in window W_2, and $OPT_{W_1 \cup W_2}$ be the optimum value of the k-means objective for the data set seen in window $W_1 \cup W_2$. Then: $OPT_{W_1} + OPT_{W_2} \leq OPT_{W_1 \cup W_2}$.*

Proof. Let C_1 be the clustering minimizing the k-means objective on the window W_1 and C_2 be the clustering minimizing the k-means objective on the window W_2 and let the C_3 be the clustering minimizing the k-means objective on the window $W_1 \cup W_2$. Then: $OPT_{W_1 \cup W_2} = \sum_{x \in W_1 \cup W_2} \min_{z \in C_3} \|x - z\|^2 = \sum_{x \in W_1} \min_{z \in C_3} \|x-z\|^2 + \sum_{x \in W_2} \min_{z \in C_3} \|x-z\|^2 \geq \sum_{x \in W_1} \min_{z \in C_1} \|x-z\|^2 + \sum_{x \in W_2} \min_{z \in C_2} \|x-z\|^2 = OPT_{W_1} + OPT_{W_2}$. □

Lemma 4. *Given any b-approximate k-means clustering algorithm, the sum over a sequence of length T of its k-means costs when run on sliding windows W_t of size $\leq W$, obeys: $\sum_{t=1}^T \Phi_{W_t} \leq b \cdot W \cdot OPT_{<1,T>}$.*

Lemma 5. *Given any b-approximate k-means clustering algorithm, a, its cumulative loss when run on a sliding window of size $\leq W$ on a stream of length T, obeys: $\sum_{t=1}^T L_t(a) \leq \frac{b \cdot W}{4R^2} OPT_{<1,T>}$.*

In all results below, we denote by OPT_T the optimum value of the k-means objective for the entire sequence of length T. Now we state the bound for the OCE Static-Expert algorithm.

Theorem 6. *Given any sequence of length T, let a_i* be the best expert in hindsight (with respect to cumulative loss on the sequence). When a_i* is a b-approximate batch clustering algorithm run on sliding windows W_t of size $\leq W$: $L_T(\mathtt{alg}) \leq \frac{b \cdot W}{4R^2} OPT_T + 2\log n$.*

Proof. We will expand the result from Theorem 2, using our instantiation of the experts as b_i-approximate clustering algorithms, trained on sliding windows of the data. For ease of notation let us denote by b, the approximation factor for a_i^*, the best expert with respect to minimizing $L_T(a_i)$ on the observed sequence of length T. $L_T(\mathtt{alg}) \leq L_T(a_i^*) + 2\log n = \sum_{t=1}^T L_t(a_i^*) + 2\log n \leq \frac{b \cdot W}{4R^2} OPT_{<1,T>} + 2\log n$. where the last inequality follows from Lemma 5. □

Using similar arguments, along with Lemma 1, we can extend our regret bounds for the other algorithms to provide online variants of approximation guarantees. We provide two such bounds for our OCE variant of Fixed-Share, corresponding to Theorems 3 and 1; the appendix contains our bound for the general version.

Theorem 7. *For any sequence of length T, and for any $s < T$, consider the best partition, computed in hindsight, of the sequence into s segments, mapping each segment to its best expert. Let each of the n experts be b-approximate w.r.t. k-means, and run on sliding windows W_t of size $\leq W$. Then, letting $\alpha' = s/(T-1)$: $L_T(\mathtt{alg}) \leq \frac{bW}{4R^2} OPT_T$*
$+ 2[\log n + s\log(n-1) + (T-1)(H(\alpha') + D(\alpha' \| \alpha))]$

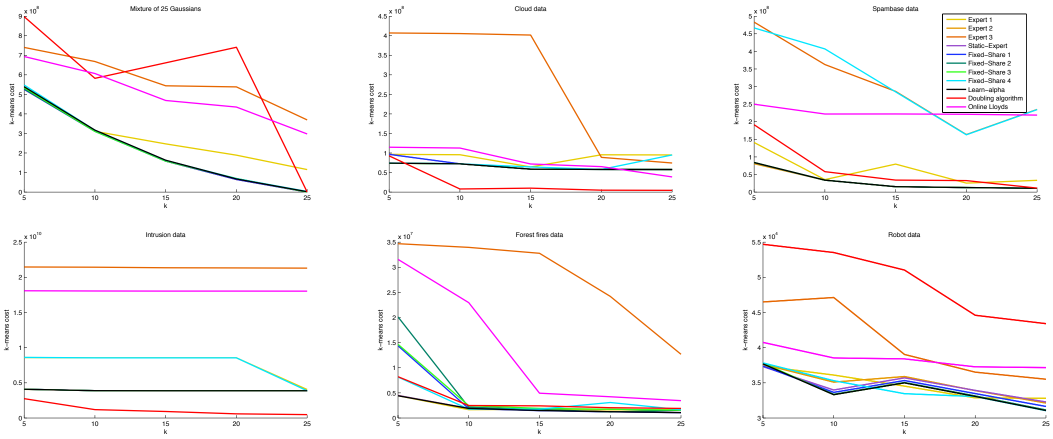

Figure 1: k-means cost on the entire sequence, versus k, per experiment. Legend in upper right.

Theorem 8. *For our OCE algorithm's Fixed-share(α) variant, where α^* is the hindsight-optimal α:*

$$L_T(\alpha) \leq \frac{bW}{4R^2} OPT_T + 2(T-1) D(\alpha^* \| \alpha)$$

In our bound for the OCE Learn-α variant, the choice of discretization, $\{\alpha_j\}$, governs the second term.

Corollary 2. *For our OCE algorithm's Learn-α variant, using a discretization of the α parameter, $\{\alpha_j\}$ of size m, where α^* is the hindsight optimal α:*

$$L_T(\mathtt{alg}) \leq \frac{bW}{4R^2} OPT_T + 2(T-1) \min_{\{\alpha_j\}} D(\alpha^* \| \alpha_j) + \log m$$

The proof combines Theorem 5 with the bound on $L_T^{\log}(\alpha^*)$ that was derived in proving Theorem 8.

5 Experiments

We ran experiments with OCE, using other clustering algorithms from the literature as experts. We used real and simulated data sets, some of which had been used in past works on batch clustering [5], and clustering (finite) data streams [3]. The simulated data consists of samples from a mixture of 25 well-separated Gaussians in $\mathbb{R}^{15}$; the "true" k is therefore 25. We also experimented on 5 UCI data sets, in which the "true" k is unknown: Cloud ($d = 10$), Spambase ($d = 58$), Intrusion (KDD cup 99; $d = 38$), Forest Fires ($d = 13$), and Wall-following robot navigation ($d = 24$) [6]. We used $N = 1000$ samples for all data sets except Cloud ($N = 1024$) and Forest Fires ($N = 517$).

We used 6 clustering algorithms as experts:[6] 1. Lloyd's algorithm.[7] 2. k-means++. 3. An heuristic called "Online k-means."[8] 4.-6. 3 variants of k-means# [3]. These are all batch clustering algorithms, except Online k-means, which we also restricted to operate only on a sliding window. Only the k-means++ and k-means# variants have approximation guarantees with respect to the k-means objective. Our Theorem 2 holds regardless, but Theorem 6 requires at least the best performing expert (in hind-sight) to be b-approximate with respect to the k-means objective.

The window size, W, and R (used in our analysis to upper bound the norm of any data point, and any cluster center output by an expert), were set arbitrarily. Tuning or informing these decisions by choice of data and expert clustering algorithms, could lead to improved peformance. While we set $W = 200$, we also experimented over window size, and observed, as expected, decreasing loss with increasing window size for very stationary data sets, yet no trend for non-stationary data sets, which is consistent with our analysis (where the upper bound on loss increases with window size).

Table 1 reports mean and standard deviation, over the sequence, of the k-means cost on all points seen so far. The experts are denoted e_i; the OCE methods are: se=Static-Expert (equivalent to Fixed-Share with $\alpha = 0$), f_{1-3} are Fixed-Share algorithms using the smallest 3 values of α in the discretization, f_4 uses the middle value, and f_5 uses the highest value, and la=Learn-α. We ran experiments with 3 experts ($e1 - e3$) and 6 experts ($e1 - e6$); the table separates the experts from the algorithms compared, in each experiment. In the 3-expert experiment, we also compared with several online clustering algorithms from the literature, run on the whole sequence: ol=Online k-means (which, run on windows, was also used as ex-

[6] We used implementations of these clustering algorithms by Ragesh Jaiswal.

[7] Centers were initialized randomly, per sliding window.

[8] This algorithm has apparently been used in practice for a while; pseudocode is stated in [15].

	25 Gaussians	Cloud ×10^7	Spam ×10^8	Intrus. ×10^{10}	For. fire ×10^6	Robot ×10^4
e_1	0.6193±0.3195×10^8	1.3180±1.9395	0.2706±0.2793	0.1988±0.2104	0.7766±0.6413	1.8362±1.2172
e_2	**0.0036**±0.0290×10^7	**0.8837**±1.3834	**0.1042**±0.1463	**0.0743**±0.1041	**0.6616**±0.4832	**1.8199**±1.2102
e_3	2.0859±0.9204×10^8	4.6601±7.8013	1.6291±1.3292	0.7145±0.5376	7.1172±7.6576	2.3590±1.4070
da	0.0179±0.0723×10^8	**0.5285**±0.2959	0.1971±0.0826	**0.0050**±0.0529	1.4496±0.6484	2.5514±1.4239
ol	1.7714±0.6888×10^8	4.2322±2.4965	0.8222±0.7619	1.3518±0.3827	2.9617±1.3006	1.9806±1.0160
se	**0.0014**±0.0143×10^8	**0.8855**±1.3824	**0.1059**±0.1469	**0.0778**±0.1094	0.6620±0.4831	1.8139±1.2032
f_1	**0.0014**±0.0143×10^8	**0.8855**±1.3823	**0.1059**±0.1469	0.0779±0.1100	**0.6614**±0.4819	1.8137±1.2032
f_2	**0.0014**±0.0143×10^8	0.9114±1.4381	**0.1059**±0.1470	**0.0778**±0.1099	0.7008±0.5382	**1.8134**±1.2031
f_3	**0.0014**±0.0143×10^8	1.0715±1.6511	**0.1059**±0.1470	0.0779±0.1099	0.6996±0.5361	1.8145±1.2031
f_4	0.0124±0.0193×10^8	1.4806±2.6257	0.3723±0.7351	0.1803±0.2358	1.0489±1.4817	1.8334±1.2212
f_5	1.3811±1.0881×10^8	3.0837±6.3553	0.8212±1.1583	0.4126±0.5040	4.4481±6.2816	2.2576±1.3849
la	**0.0012**±0.0136×10^8	0.8862±1.3920	**0.1076**±0.1483	0.0785±0.1108	**0.6616**±0.4805	**1.8130**±1.2026
e_4	**7.3703**±4.2635×10^3	**0.6742**±1.2301	**0.0687**±0.1355	**0.0704**±0.1042	**0.2316**±0.2573	**1.3667**±1.0176
e_5	8.2289±4.4386×10^3	0.6833±1.2278	0.0692±0.1356	**0.0704**±0.1042	0.2625±0.2685	1.4385±1.0495
e_6	9.8080±4.7863×10^3	0.7079±1.2364	0.0710±0.1360	0.0705±0.1042	0.3256±0.2889	1.5713±1.1011
se	**0.1360**±1.4323×10^6	**0.6743**±1.2300	**0.0687**±0.1355	**0.0705**±0.1045	0.2322±0.2571	1.3642±1.0138
f_1	**0.1360**±1.4323×10^6	**0.6743**±1.2300	**0.0687**±0.1355	**0.0705**±0.1045	0.2322±0.2571	1.3640±1.0135
f_2	0.1361±1.4322×10^6	0.6746±1.2298	**0.0687**±0.1355	**0.0705**±0.1045	0.2322±0.2572	1.3636±1.0130
f_3	0.1364±1.4322×10^6	**0.6743**±1.2300	**0.0687**±0.1355	0.0711±0.1055	**0.2321** ±0.2570	**1.3634**±1.0127
f_4	0.0027±0.0144×10^8	0.7207±1.3025	**0.0707**±0.1357	0.0773±0.1203	0.2776±0.4917	1.3963±1.0339
f_5	1.4039±1.0790×10^8	3.0786±6.4109	0.7155±1.0650	0.4227±0.5179	4.6103±6.3019	2.3142±1.4127
la	**0.0012**±0.0134×10^8	**0.6742**±1.2300	**0.0687**±0.1355	**0.0708**±0.1046	**0.2318**±0.2573	**1.3632**±1.0128

Table 1: Mean and standard deviation, over the sequence, of k-means cost on points seen so far. $k = 25$ for Gaussians, $k = 15$ otherwise. The best expert and the best 2 scores of the algorithms, per experiment, are bold. Below the triple lines, 3 more experts are added to the ensemble.

pert 3.), and the "Doubling algorithm" (da).[9] Neither of these have known approximation guarantees with respect to the k-means objective, however the Doubling algorithm approximates the k-center objective [12]. This evaluation is a k-means cost variant of progressive validation analysis, a standard evaluation of online learning algorithms in the supervised setting, analyzed in [10] with respect to k-fold cross-validation error and standard batch holdout error.

In two of the experiments, the Doubling algorithm achieves the lowest mean k-means cost over the sequence; in the other experiments, at least one of our OCE methods does. Both the 3 and 6-expert experiments demonstrate that, without prior knowledge of the sequence, Learn-α is the most applicable, as its performance tracks that of the best Fixed-Share(α).

In Figure 1 we vary k from 5 to 25 in multiples of 5 as in [3], and state the final k-means cost achieved by each method in the 3-experts experiments. The OCE methods often achieve lower final k-means cost than the other algorithms. In particular, Learn-α suffers lower k-means cost than Online k-means in all experiments except Cloud: $k = 25$. For most k, the Doubling algorithm achieves lower cost on 2 data sets, but higher cost on the remaining 4 data sets, although in the simulation where the true $k = 25$, the results are comparable. For Robot data, which is clearly non-stationary, the performance of Doubling is significantly worse than that of Learn-α for all k tested. Thus OCE has (at least) comparable performance over a variety of data sets to these existing methods that have not been analyzed with respect to the k-means objective; moreover, it exploits the performance advantages of the clustering algorithms used as experts.

6 Conclusions and future work

There are several interesting directions for future work. Extending from our present work, we are looking into aggregating all the cluster centers over all experts, so as to remove the assumption that experts filter their centers to return the one closest to the current point in the stream. We would also like to run thorough experiments on extremely non-stationary data sets to better explore the empirical differences among the various algorithms.

We believe this is the first work providing regret bounds for online clustering in the case $k > 1$. This opens up a range of possibilities; we can also explore an online clustering with experts setting in which the experts need not be clustering algorithms.

[9] These algorithms output k centers so running them with experts 4-6, which can output more than k centers, would not be a fair comparison, since OCE can output as many centers as its experts.

References

[1] Marcel R. Ackermann, Christian Lammersen, Marcus Märtens, Christoph Raupach, Christian Sohler, and Kamil Swierkot. Streamkm++: A clustering algorithms for data streams. In *ALENEX*, 2010.

[2] A. Aggarwal, A. Deshpande, and R. Kannan. Adaptive sampling for k-means clustering. In *APPROX*, 2009.

[3] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In *NIPS*, 2009.

[4] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75:245–248, May 2009.

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, 2007.

[6] A. Asuncion and D.J. Newman. UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences, 2007.

[7] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *SODA*, 2009.

[8] M. Belkin and K. Sinha. Toward learning gaussian mixtures with arbitrary separation. In *COLT*, 2010.

[9] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *ICML*, pages 97–104, 2006.

[10] A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *COLT*, 1999.

[11] Olivier Cappé and Eric Moulines. Online EM algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71:593–613, 2009.

[12] Moses Charikar, Chandra Chekuri, Tomas Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

[13] K. Chaudhuri and S. Rao. Learning mixtures of product distributions using correlations and independence. In *COLT*, 2008.

[14] Sanjoy Dasgupta. Learning mixtures of gaussians. In *FOCS*, 1999.

[15] Sanjoy Dasgupta. Course notes, CSE 291: Topics in unsupervised learning. Lecture 6: Clustering in an online/streaming setting. Section 6.2.3. In *http://www-cse.ucsd.edu/∼dasgupta/291/lec6.pdf, University of California, San Diego, Spring Quarter*, 2008.

[16] X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *ICML*, 2004.

[17] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[18] David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Trans. on Information Theory*, 44(5):1906–1925, 1998.

[19] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32:151–178, 1998.

[20] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry: Theory and Applications*, 28:89–112, 2004.

[21] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The clustree: Indexing micro-clusters for anytime stream mining. In *Knowledge and Information Systems Journal (KAIS)*, 2010.

[22] Percy Liang and Dan Klein. Online EM for unsupervised models. In *NAACL*, pages 611–619, 2009.

[23] S. P. Lloyd. Least square quantization in pcm. *Bell Telephone Laboratories Paper*, 1957.

[24] Adam Meyerson. Online facility location. In *FOCS*, 2001.

[25] Claire Monteleoni and Tommi Jaakkola. Online learning of non-stationary sequences. In *NIPS*, 2003.

[26] Claire E. Monteleoni. Online learning of non-stationary sequences. SM Thesis. In *MIT Artificial Intelligence Technical Report 2003-011*, 2003.

[27] Vikas Singh, Lopamudra Mukherjee, Jiming Peng, and Jinhui Xu. Ensemble clustering using semidefinite programming with applications. *Mach. Learn.*, 79:177–200, May 2010.

[28] Andrea Vattani. k-means requires exponentially many iterations even in the plane. In *25th Annual Symposium on Computational Geometry*, 2009.

[29] V. G. Vovk. A game of prediction with expert advice. *J. Comput. Syst. Sci.*, 56:153–173, April 1998.

[30] Manfred K. Warmuth and Dima Kuzmin. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems 19*, pages 1481–1488, 2007.