Journal of Machine Learning Research vol (2011) 1–18 Submitted September 1, 2011; Published publication date

Online Clustering with Experts

Anna Choromanska AEC2163@COLUMBIA.EDU
Department of Electrical Engineering, Columbia University
Claire Monteleoni* CMONTEL@GWU.EDU
Department of Computer Science, George Washington University

Editor: Editor's name

Abstract

We propose an online clustering algorithm that manages the exploration/exploitation trade-off using an adaptive weighting over batch clustering algorithms. We extend 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 algorithm computes 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 algorithm achieves an approximation guarantee with respect to the k-means objective. The form of this online clustering approximation guarantee is novel, and extends an evaluation framework proposed by Dasgupta as an analog to regret. Our algorithm tracks the best clustering algorithm on real and simulated data sets.

Keywords: Clustering, online learning, online clustering, clustering data streams, k-means clustering objective, learning with expert advice, approximation algorithms

1. Introduction

As data sources continue to grow at an unprecedented rate, it is increasingly important that algorithms to analyze this data operate in the *online learning* setting. This 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 that is so large compared to computational resources, that algorithms must access it in a sequential manner. In the online learning model, only one pass is allowed, and the data stream is infinite.

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

* This work was done while C.M. was at the Center for Computational Learning Systems, Columbia University.

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. Moreover, our goal is the design of generic clustering algorithms, as opposed to application-specific ones. 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 mixture of Gaussians where the means are separated by some function of the covariance matrices (Dasgupta, 1999).

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 useful. 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 (2008), our goal is to approximate the k-means objective in the online setting. Our approach manages the online exploration/exploitation tradeoff via adaptive weighting of batch clustering algorithms.

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, which is simple and intuitive. For a finite set, S, of n points in $\mathbb{R}^d$, and a fixed positive 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 $k = 2$ (Aloise et al., 2009).[1] Therefore approximation algorithms are desirable. We will refer to the following types of approximation guarantee.

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

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

Although many clustering algorithms have been designed with the k-means objective in mind, very few have approximation guarantees with respect to this objective, even in the batch setting. Even the algorithm commonly known as "k-means" does not have an approximation guarantee.

1. Although this result has been reported by earlier authors, according to Aloise et al. (2009) the first reported *correct* proofs were shown independently in 2008 in unpublished works by S. Dasgupta, and by A. Deshpande and P. Popat.

Our contribution is an online learning algorithm, with a regret bound, and an online variant of an approximation guarantee with respect to the k-means objective. We extend algorithms from Herbster and Warmuth (1998) to the unsupervised learning setting, and introduce a flexible framework in which our algorithm takes a set of candidate clustering algorithms, as experts, and tracks the performance of the "best" expert for the data. 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.[2]

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 (Lloyd, 1957). While it typically (but not always (Vattani, 2009)) 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.* Dasgupta (1999); Chaudhuri and Rao (2008).[3] 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 (Balcan et al., 2009). 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 $O(\log k)$ (Arthur and Vassilvitskii, 2007). Constant approximations have been shown for a local search technique (Kanungo et al., 2004), and recent work (Aggarwal et al., 2009), which outputs $O(k)$ centers. The k-means# algorithm (Ailon et al., 2009) provides a constant approximation, and outputs $O(k \log k)$ centers. Several works have studied clustering *finite* data streams, *e.g.* Guha et al. (2003); Ailon et al. (2009); Ackermann et al. (2010). Work by Ailon et al. (2009) does so with respect to the k-means objective, and extends a streaming clustering algorithm (Guha et al., 2003) 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.

A number of techniques for online clustering have enjoyed success in practice, such as Kranen et al. (2010), and variants of EM (e.g. Liang and Klein (2009)), and some have been analyzed under stochastic assumptions on the data, *e.g.* Cappé and Moulines (2009). Several online clustering algorithms have approximation guarantees with respect to clustering objectives other than k-means. A doubling algorithm due to Charikar et al. (2004), and the cover tree algorithm of Beygelzimer et al. (2006), both provide a constant

2. In subsequent work (Choromanska and Monteleoni, 2012) we have extended this approach to provide a family of online clustering algorithms which differ in their models of the time-varying nature of the data. These algorithms track a shifting sequence of "best" experts, as opposed to the best fixed expert. The conference version also fixes some technical issues from this workshop version. Therefore, for full, updated proofs, and algorithm pseudocode, please refer to the conference version.
3. Belkin and Sinha (2010) provides an algorithm for an arbitrarily small separation, when the clusters are spherical Gaussians.

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.[4] Finally, there has also been some work on ensemble methods for clustering, in a batch setting, *e.g.* Fern and Brodley (2004); Singh et al. (2010).

2. Online k-means approximation

First we address the challenge of specifying an evaluation framework for online clustering. One cannot use a similar analysis setting to Arthur and Vassilvitskii (2007); Ailon et al. (2009), for the finite data stream case. With no assumptions on the data stream, one can always design a sequence of observations that will fool a seeding algorithm (one 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 (Dasgupta, 2008). 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 (where the algorithm outputs a clustering, C_t, before observing the current point):

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

and the optimal k-means cost on the points seen so far. We analyze a similar quantity, but we bound the non-predictive clustering loss; the clustering algorithms observe the current point before trying to cluster it, and thus the clustering output at time t takes into account point x_t, in addition the rest of the points in the sliding window. We first obtain performance guarantees for our algorithm that compare the quantity in Equation 1 (where the point x_t is now observed before the clusterings are output) to that of the best (in hindsight) attainable by a finite set of batch clustering algorithms. Then, we encorporate the b-approximation guarantees of the batch clustering algorithms into the regret bounds to obtain bounds with respect to the optimum of the k-means objective on all the points seen.

2.1. 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. Let C_t^i denote the set of centers output by the i^{th} expert at time t. The n experts are indexed by i. We denote the center in C_t^i closest to the data point x_t as $c_t^i = \arg\min_{c \in C_t^i} \|x_t - c\|^2$.

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

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

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 i and all t. Given the bound on $\|x\|$, this would certainly hold for any reasonable centers. When we compute the loss of our combined algorithm, L will just take one argument. We refer to cumulative loss over time (from the first observation) of either an expert or the 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 $L_{(s,t>}$.

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

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, and the $k-1$ (or $ak-1$) centers, $C_t^{i^*} - \{c_t^{i^*}\}$, where $i^* = \arg\max p_t(i)$, *i.e.* the remaining centers output by the clustering algorithm with the current highest weight. However, since this does not affect the math, we will simply refer to the clustering as defined above.

3. Algorithm

Our "loss" and "clustering" functions are designed as unsupervised analogs to "prediction loss," and "prediction," from the supervised online learning literature. While our regret bounds hold in both predictive and non-predictive clustering settings, for our approximation guarantees 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.

Using these analogies, instantiating a loss function and a clustering function immediately suggests unsupervised variants of several algorithms proposed in Herbster and Warmuth (1998), that vary with the definition of the update rule for $p_t(i)$. Different update rules correspond to different assumptions on the level of non-stationarity in the data, and manage the exploration/exploitation tradeoff accordingly. In the present paper we give the analysis for a variant of their Static-Expert algorithm (which also has a long history in the literature), however in subsequent work (**?**) we have extended our analyses to variants of their Fixed-Share algorithm, and the Learn-α algorithm of Monteleoni and Jaakkola (2003), in order to address applications in which the observations are likely to vary with time.

Algorithm 1 specifies our Online Clustering with Experts (OCE) algorithm, when experts are instantiated by clustering algorithms run on sliding windows W_t of the data stream, which include the current observation x_t, and slide forward by one observation at a time. More generally, our regret analysis holds (Theorem 6) even if the experts are arbitrary black-boxes that output a set of centers at each iteration. Note that the set of centers output by any expert can be of any size, including greater than k, which allows for the use of (a, b)-approximation algorithms as experts. Additionally, our analysis holds even when different experts are trained on different window sizes.

The time for forming a clustering is $O(knd)$ when run with experts that are clustering algorithms that return exactly k centers, and $O(aknd)$ when run with (a_i, b_i)-approximate clustering algorithms as experts, where $a = \max_{i\in\{1,\ldots,n\}} a_i$.[5]

Algorithm 1: Online Clustering with Experts (OCE)

Inputs: Clustering algorithms $\{a_1, a_2, \ldots, a_n\}$,
k: desired number of clusters, R: large constant
W: window size
Initialization: $p_0(i) = 1/n$, for all $i \in \{1, \ldots, n\}$.
For t = 1,2,...
  Receive data point x_t in the stream
  For each $i \in \{1, \ldots, n\}$:
    Run a_i on data chunk W_t, outputting C^i
    Compute $c^i = \arg\min_{c\in C^i} ||x_t - c||^2$
    $p_t(i) = p_{t-1}(i)e^{-\frac{1}{2}L(x_t, C_t^i)}$.
  Normalize p_t.
  $\texttt{clust}(x_t) = \sum_{i=1}^n p_t(i)c^i$

4. Performance Guarantees

We will give two types of performance guarantees for our clustering variant of the Static-Expert algorithm. First we will show that a similar bound to the Herbster and Warmuth (1998) bound for the setting of supervised online learning with experts holds. Then we will instantiate the experts as batch clustering algorithms, with aproximation guarantees, to yield an online variant of an approximation guarantee with respect to the k-means objective. The bounds are not optimized with respect to constants.

In order to make use of analysis tools from the literature, we first need to show that for our prediction and loss function, a certain property holds. Here we reformulate the definition of (c, η)-**realizability** presented in Herbster and Warmuth (1998), and due to Haussler et al. (1998); Vovk (1998), and demonstrate that it holds in our setting.

Theorem 5 *The loss function L defined in Definition 3 and the prediction function defined in Definition 4 are $(2, \frac{1}{2})$-**realizable**, in that the following inequality is satisfied.*

$$L(\texttt{clust}(x_t)) \leq -2\log \sum_{i=1}^{n} p(i)e^{-\frac{1}{2}L(x_t, C_t^i)}$$

for all $n \in \mathbb{N}$ and all x_t and c_t^i such that $||x_t|| \leq R$, $||c_t^i|| \leq R$, where R is a normalization factor from Definition 3, and all stochastic vectors $p \in [0\,1]^n$.

Proof Using Definitions 3 and 4, we can express the loss of the algorithm on a point x_t as

$$L(\texttt{clust}(x_t)) = \frac{\| \sum_{i=1}^n p(i)(x_t - c_t^i)\|^2}{4R^2}$$

5. This is in addition to the total running time of the clustering sub-algorithms used as experts.

Then the following chain of inequalities are equivalent to eachother.

$$\begin{aligned} L(\texttt{clust}(x_t)) &\leq -2\log\sum_{i=1}^{n} p(i)e^{-\frac{1}{2}L(x_t,C_t^i)} \\ \frac{\|\sum_{i=1}^{n} p(i)(x_t - c_t^i)\|^2}{4R^2} &\leq -2\log\sum_{i=1}^{n} p(i)e^{-\frac{1}{2}\frac{\|x_t-c_t^i\|^2}{4R^2}} \\ e^{\frac{\|\sum_{i=1}^{n} p(i)(x_t-c_t^i)\|^2}{4R^2}} &\leq \left(\sum_{i=1}^{n} p(i)e^{-\frac{1}{2}\frac{\|x_t-c_t^i\|^2}{4R^2}}\right)^{-2} \\ \sum_{i=1}^{n} p(i)e^{-\frac{1}{2}\frac{\|x_t-c_t^i\|^2}{4R^2}} &\leq e^{-\frac{1}{2}\frac{\|\sum_{i=1}^{n} p(i)(x_t-c_t^i)\|^2}{4R^2}} \end{aligned} \tag{2}$$

Let $v_t^i = \frac{x_t - c_t^i}{2R}$. Since $\|x_t\| \leq R$ and $||c_t^i|| \leq R$ then $v_t^i \in [-1\ \ 1]^n$. Equation (2) is equivalent to

$$\sum_{i=1}^{n} p(i)e^{-\frac{1}{2}\|v_t^i\|^2} \leq e^{\frac{-1}{2}\|\sum_{i=1}^{n} p(i)v_t^i\|^2}$$

This inequality holds by Jensen's Theorem since the function $f(v_t^i) = e^{-\frac{1}{2}\|v_t^i\|^2}$ is concave when $v_t^i \in [-1\ \ 1]^n$. ■

4.1. Regret Bound

Theorem 6 *Let T be the number of training iterations and let a_i* be the best expert (with respect to cumulative loss, computed in hind-sight). Then the following inequality between the cumulative loss of the algorithm on T iterations and the cumulative loss of the best expert on T iterations holds:*

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

The proof follows by an almost direct application of an analysis technique of Monteleoni and Jaakkola (2003) which views the algorithms of Herbster and Warmuth (1998) as Bayesian updates of a generalized Hidden Markov Model, in which the hidden variable is the identity of the best expert. We defer the proof to the supplementary material.

4.2. Approximation Guarantees

In the previous subsection, we gave a regret guarantee for the cumulative loss of the algorithm with respect to the cumulative loss of the best expert. In the case in which the experts are approximation algorithms for k-means, we can continue to prove an online variant of an approximation guarantee for the algorithm. Here we assume that each expert i is a b_i-approximate clustering algorithm, a_i, with respect to the k-means objective. Using this assumption we can give an online approximation bound for our algorithm. First we provide the following lemmas, which may also be of general interest.

Lemma 7 *Let OPT_{W_1} be the optimum value of the k-means objective for the dataset seen in the window W_1 and let OPT_{W_2} be the optimum value of the k-means objective for the dataset seen in the window W_2 and let $OPT_{W_1 \cup W_2}$ be the optimum value of the k-means objective for the dataset seen in the window $W_1 \cup W_2$. Then the following inequality holds:*

$$OPT_{W_1} + OPT_{W_2} \leq OPT_{W_1 \cup W_2}$$

The proof is deferred to the appendix.

Lemma 8 *Given any clustering algorithm that is b-approximate with respect to the k-means objective, the sum over a sequence of length T of its k-means cost when run on sliding windows W_t of size $\leq W$, obeys:*

$$\sum_{t=1}^{T} \Phi_{W_t} \quad \leq \quad b \cdot W \cdot OPT_{<1,T>}$$

The proof makes use of Lemma 7, and is deferred to the supplementary material.

Lemma 9 *Given any clustering algorithm, a, that is b-approximate with respect to the k-means objective, its cumulative loss, as defined in 3, when run on a sliding window of size $\leq W$ on a stream of length T, will obey the following bound.*

$$\sum_{t=1}^{T} L_t(a) \leq \frac{b \cdot W}{4R^2} OPT_{<1,T>}$$

Proof The loss of expert i at time t is defined in 3 as the scaled component of the k-means cost of algorithm a's clustering at time t. That is,

$$\begin{aligned}
\Phi_t(C_t) &= \sum_{x'_t \in W_t} \min_{c \in C_t} \|x_{t'} - c\|^2 \\
&= \min_{c \in C_t} \|x_t - c\|^2 + \sum_{x'_t \in W_t \setminus x_t} \min_{c \in C_t} \|x_{t'} - c\|^2 \\
&= 4R^2 \cdot L_t(C_t) + \sum_{x'_t \in W_t \setminus x_t} \min_{c \in C_t} \|x_{t'} - c\|^2
\end{aligned}$$

Therefore, since all terms in the sum are positive,

$$\begin{aligned}
4R^2 \cdot L_t(C_t) &\leq \Phi_t(C_t) \\
L_t(a) &\leq \frac{\Phi_t}{4R^2}
\end{aligned}$$

where in the second inequality we substitute in our simplifying notation, where C_t are the set of clusters generated by algorithm a at time t, and Φ_t is algorithm a's k-means cost on W_t. Now, summing over T iterations, and applying Lemma 8, we obtain

$$\sum_{t=1}^{T} L_t(a) \leq \frac{b \cdot W}{4R^2} OPT_{<1,T>}$$

■

We are now ready to state the Theorem.

Theorem 10 *Let T be the number of training iterations and let a_i* be the best expert (with respect to cumulative loss, computed in hind-sight). When a_i* is a b-approximate batch clustering algorithm run on sliding windows W_t of size $\leq W$, the following inequality holds:*

$$L_T(alg) \leq \frac{b \cdot W}{4R^2} OPT_T + 2 \log n$$

Proof We will expand the result from Theorem 5, using our instantiation of the experts as b_i-approximate clustering algorithms, training 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.

$$\begin{aligned} L_T(\texttt{alg}) &\leq L_T(a_i^*) + 2 \log n \\ &= \sum_{t=1}^{T} L_t(a_i^*) + 2 \log n \\ &\leq \frac{b \cdot W}{4R^2} OPT_{<1,T>} + 2 \log n \end{aligned}$$

where the last inequality follows from Lemma 9. ■

5. Experiments

We ran experiments with our algorithm for online clustering with experts, using other clustering algorithms from the literature as experts. We chose real and simulated data sets that had been used in several previous related works on batch clustering (Arthur and Vassilvitskii, 2007), and clustering (finite) data streams (Ailon et al., 2009). In describing the data sets, m denotes the number of points in the data, d denotes the dimension, and k denotes the number of clusters. The simulated data was generated by Ailon et al. (2009), and consists of samples from a mixture of 25 Gaussians in $\mathbb{R}^{15}$; the "true" k is therefore 25. We report results on 1000 samples. We also experimented on the UCI data sets Cloud, and Spambase (Asuncion and Newman, 2007). For Cloud $m = 1024$ and $d = 10$. Spambase contains 4601 examples in 58 dimensions; we experimented on 2000 of them.[6] As in Ailon et al. (2009), since the true k is unknown, we vary k from 5 to 25 in multiples of 5.

We instantiated the experts with 6 clustering algorithms, numbered as follows:

1. Lloyd's algorithm.
2. k-means++.
3. k-means# that outputs $3 \cdot k \cdot \log k$ centers.
4. k-means# that outputs $2.25 \cdot k \cdot \log k$ centers.

6. We found similar results on the full Spambase data set, as well as on 10,000 points from the Gaussian simulation, in experiments with just the first 3 experts.

k	5	10	15	20	25
Exp1	3.8975	2.5061	1.5337	1.1316	0.7257
Exp2	3.8726	2.2544	1.1937	0.4219	0.0021
Exp3	0.8526	0.0002	0.0002	0.0002	0.0001
Exp4	1.4298	0.0002	0.0002	0.0002	0.0002
Exp5	4.4704	0.0328	0.0002	0.0002	0.0002
Exp6	5.4353	4.9212	3.5766	2.9348	2.1550
OCE	0.8561	0.0002	0.0002	0.0002	0.0001

k	5	10	15	20	25
Exp1	0.1254	0.0680	0.0329	0.0209	0.0174
Exp2	0.1095	0.0349	0.0195	0.0128	0.0090
Exp3	0.0833	0.0608	0.0555	0.0536	0.0529
Exp4	0.1034	0.0647	0.0583	0.0547	0.0538
Exp5	0.1851	0.0769	0.0615	0.0587	0.0554
Exp6	1.1388	0.7919	0.7905	0.6060	0.6038
OCE	0.0825	0.0607	0.0324	0.0128	0.0090

k	5	10	15	20	25
Exp1	0.1490	0.0742	0.0492	0.0384	0.0363
Exp2	0.0748	0.0182	0.0086	0.0050	0.0031
Exp3	0.4690	0.4638	0.4636	0.4635	0.4635
Exp4	0.4740	0.4642	0.4636	0.4636	0.4635
Exp5	0.5150	0.4656	0.4639	0.4637	0.4636
Exp6	1.2761	1.2745	1.2658	1.2656	1.2655
OCE	0.1084	0.0182	0.0086	0.0050	0.0031

Table 1: Cumulative loss versus k. Top: 25 Gaussian simulation; units of 10^4. Middle: UCI Cloud data; units of 10^4. Bottom: UCI Spambase data; units of 10^5.

5. k-means# that outputs $1.5 \cdot k \cdot \log k$ centers.
6. An heuristic often referred to as "Online k-means."[7]

The algorithms are all batch clustering algorithms, except for Online k-means; we also limited it to operate only on the current data in the sliding window. Note that only the k-means++ and k-means# variants have approximation guarantees with respect to the k-means objective. Our Theorem 6 holds regardless, but Theorem 10 requires at least the best performing expert (in hind-sight) to be b-approximate with respect to the k-means objective. Conveniently, our experiments reveal that, on these data sets, usually k-means++ or a k-means# variant is the best in hindsight for the data. Lloyd's algorithm requires initialization; the centers were initialized randomly, per sliding window.

7. This algorithm has apparently been used in practice for a while; pseudocode is stated in Dasgupta (2008).

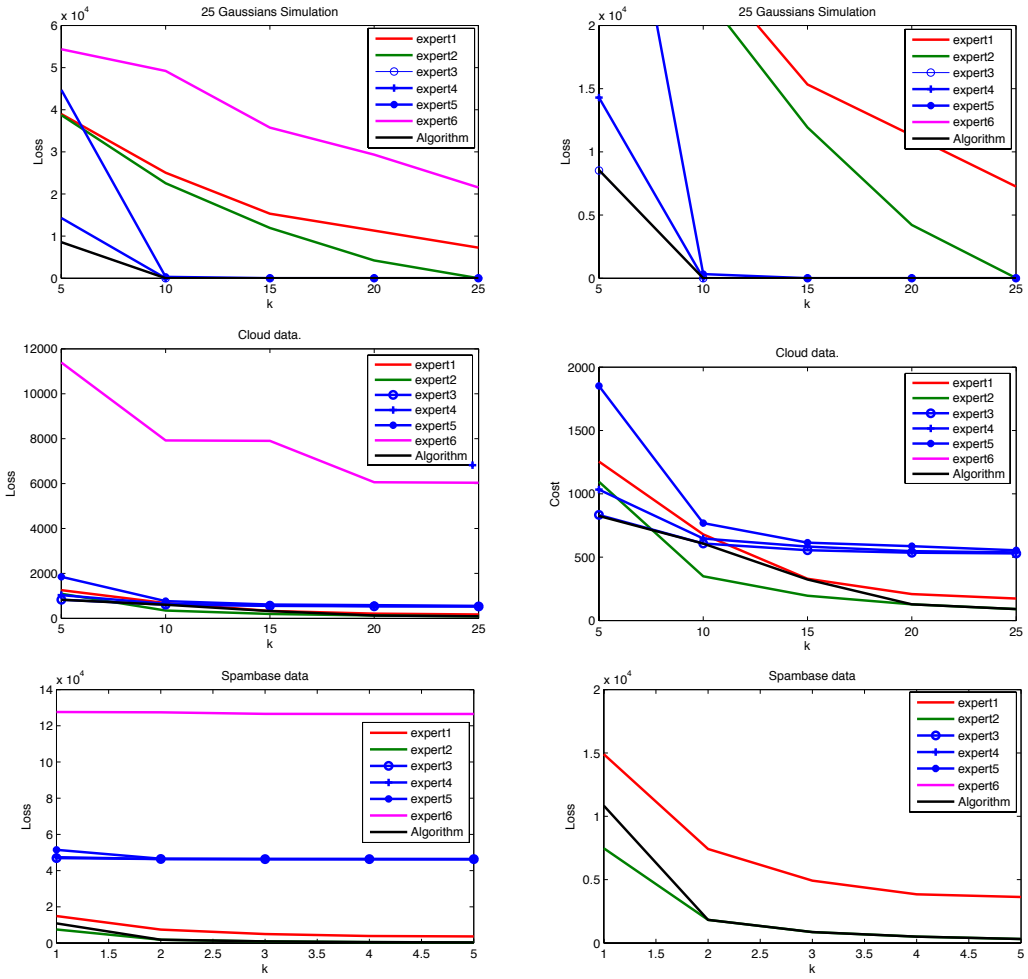

Figure 1: Cumulative loss vs. k. Top: mixture of 25 Gaussians simulation. Middle: UCI Cloud data. Bottom: UCI Spambase data. For each experiment, the right plot zooms in on the lower part of the loss axis.

We made no attempt to tune the size of the sliding window; it was set arbitrarily to 200 for all algorithms. Although the algorithm can start training from the first observation, via the analysis treating smaller window sizes (encountered at the beginning and the end of the sequence), in the experiments we used the first batch of 200 points as input to all the clustering algorithms, and started training the algorithm after that. Our performance guarantees hold when R upper bounds the ℓ_2 norm of any data point, and any cluster center output by an expert. We chose $R = 10,000$ arbitrarily; for better performance it can either be tuned, or informed by the choice of data and expert clustering algorithms.

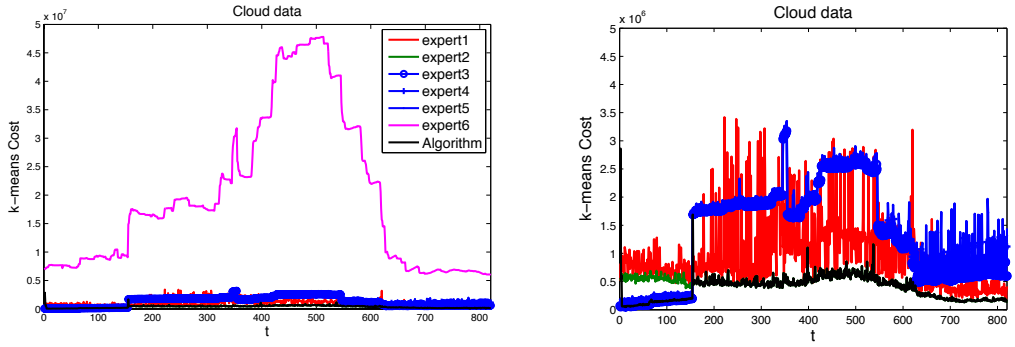

Figure 2: Clustering analogs to learning curves. k-means cost on all points seen so far, versus t. UCI Cloud, when $k = 15$. Legend refers to both figures. Right plot zooms in on the y-axis (k-means cost).

5.1. Results

Table 1 reports cumulative loss over the entire stream (where loss is defined in Definition 3), for each expert, and our algorithm, Online Clustering with Experts (OCE), per k, on the 3 data sets. Dividing by the m per experiment yields average loss. Reporting cumulative or average loss is standard in the evaluation of online learning algorithms, in the supervised setting. Cumulative loss relates directly to regret bounds, and average loss relates to a form of progressive validation error that was analyzed in Blum et al. (1999), which formally related it, as well as k-fold cross-validation error, to standard batch holdout error.

We provide visualizations of these tabular results in Figure 1. In the top two figures, corresponding to the top table in Table 1, the data is fake and the true k is 25. At that k, the algorithm's performance matches that of the best performing method, k-means#. This property holds for most other k values as well. In the middle two figures, corresponding to the middle table in Table 1, the true k is unknown. The algorithm matches the loss of the best expert for two k values, and beats it for one k value. There are two k values for which the best expert beats it, however the algorithm still beats all remaining experts. This is notable, especially since the identity of the best expert differs per k. Finally, in the bottom two figures, corresponding to the bottom table in Table 1, the true k is again unknown, and the algorithm tracks the best expert, k-means++ in this case, matching its loss at all k values but one.

To provide qualitative results on OCE's performance, in Figure 2 we show clustering analogs to learning curves. We plot the (batch) k-means cost of each expert on all the data seen so far, versus t. The algorithm OCE (which outputs k centers as discussed in Section 3) achieves lower k-means cost than all experts in all but the earlier iterations. While our present analysis upper bounds our loss function, we will explore these apparently stronger guarantees in future work.

6. Future work

In future work we plan to experiment on data known to be time-varying, using our online clustering extensions of the Fixed-Share (Herbster and Warmuth, 1998), and Learn-α (Monteleoni and Jaakkola, 2003) algorithms, to empirically compare how the exploration/exploitation tradeoffs are managed by the different techniques. Another interesting area for future work is to allow k to vary with time. This explicitly addresses the exploration/exploitation tradeoff, that is, whether to update the parameters of an existing cluster (exploit), to better decribe the current observation, versus whether to start growing a new cluster from scratch (explore).

Acknowledgments

We would like to acknowledge Ragesh Jaiswal for sharing his implementations of the k-means++, k-means#, and Lloyd's algorithms.

References

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

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

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

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

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

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

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

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

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

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

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.

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

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

Anna Choromanska and Claire Monteleoni. Online clustering with experts. In *AISTATS*, 2012.

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

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.

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

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.

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.

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

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.

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.

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

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

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

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

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

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

Appendix A. Proof of Theorem 6

Proof For ease of notation, we denote $L(i,t) = L(x_t, C_t^i)$. We can proceed by applying Theorem 5 to bound the cumulative loss of the algorithm.

$$\begin{aligned}
& L_T(\texttt{alg}) = \sum_{t=1}^{T} L(\texttt{clust}(x_t)) \\
\leq\ & -\sum_{t=1}^{T} 2\log \sum_{i=1}^{n} p_t(i) e^{-\frac{1}{2}L(i,t)} \\
=\ & -2\sum_{t=1}^{T} \log \sum_{i=1}^{n} p_t(i) P(x_t|a_i, x_1, ..., x_{t-1}) \\
=\ & -2\sum_{t=1}^{T} \log P(x_t|x_1, ..., x_{t-1}) \\
=\ & -2\log p_1(x_1) \prod_{t=2}^{T} P(x_t|x_1, ..., x_{t-1}) \\
=\ & -2\log P(x_1, ..., x_T) \\
=\ & -2\log \sum_{i=1}^{n} P(x_1, ..., x_T|a_{i,1}, ..., a_{i,T}) P(a_{i,1}, ..., a_{i,T}) \\
=\ & -2\log \sum_{i=1}^{n} p_1(i) P(x_1|a_{i,1}) \prod_{t=2}^{T} P(x_t|a_i, x_1, ..., x_{t-1}) \\
=\ & -2\log \frac{1}{n} \sum_{i=1}^{n} e^{-\frac{1}{2}L(i,1)} \prod_{t=2}^{T} e^{-\frac{1}{2}L(i,t)} \\
=\ & -2\log \frac{1}{n} \sum_{i=1}^{n} e^{-\frac{1}{2}\sum_{t=1}^{T} L(i,t)} \\
=\ & -2\log \frac{1}{n} - 2\log \sum_{i=1}^{n} e^{-\frac{1}{2}\sum_{t=1}^{T} L(i,t)}
\end{aligned}$$

The last inequality holds for any a_i, so in particular for a_i^*. ■

Appendix B. Proof of Lemma 7

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:

$$\begin{aligned} OPT_{W_1 \cup W_2} &= \sum_{x \in W_1 \cup W_2} \min_{z \in C_3} \|x - z\|^2 \\ &= \sum_{x \in W_1} \min_{z \in C_3} \|x - z\|^2 + \sum_{x \in W_2} \min_{z \in C_3} \|x - z\|^2 \\ &\geq \sum_{x \in W_1} \min_{z \in C_1} \|x - z\|^2 + \sum_{x \in W_2} \min_{z \in C_2} \|x - z\|^2 \end{aligned}$$

■

Appendix C. Proof of Lemma 8

Proof For ease of notation, we denote by $\Phi_{(t-W,t>}$ the k-means cost of algorithm a on the data seen in the window $(t-W, t>$ (omitting the argument, which is the set of centers output by algorithm a at time t). Since a is b-approximate then:

$$\forall_W \quad \Phi_{(t-W,t>} \leq b \cdot OPT_{(t-W,t>} \tag{3}$$

For any W such that $W < T$, we can decompose T such that $T = mW + r$ where $m \in \mathbb{N}$ and $r \leq W$. Notice then that for any $j \in \{0, ..., W-1\}$ the following chain of inequalities holds as the direct consequence of Equation 3:

$$\begin{aligned} \Phi_{(T-W+j,T>} \quad &+ \quad \Phi_{(T-2W+j,T-W+j>} \\ &+ \quad \Phi_{(T-3W+j,T-2W+j>} \\ &+ \ldots + \quad \Phi_{(T-mW+j,T-(m-1)W+j>} \\ &+ \quad \Phi_{<1,T-mW+j>} \\ &\leq \quad b \cdot OPT_{(T-W+j,T>} \\ &+ \quad b \cdot OPT_{(T-2W+j,T-W+j>} \\ &+ \quad b \cdot OPT_{(T-3W+j,T-2W+j>} \\ &+ \ldots + \quad b \cdot OPT_{(T-mW+j,T-(m-1)W+j>} \\ &+ \quad b \cdot OPT_{<1,T-mW+j>} \\ &\leq \quad b \cdot OPT_{<1,T>} \end{aligned}$$

where the last inequality is the direct consequence of Lemma 7.

Notice that different value of j refers to different partitioning of the time span $< 1, T >$. Figure 3 illustrates an example when T = 10 and W = 3.

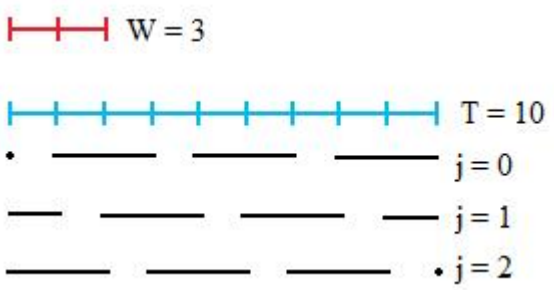

Figure 3: Different partitioning of time span T = 10 into time windows, W = 3, with respect to different values of j.

Let W_t refer to the data chunk seen in time span $< \max(1, t - W + 1), t >$. We finish by showing that,

$$\begin{aligned} \sum_{t=1}^{T} \Phi_{W_t} \quad &\leq \quad \sum_{j=0}^{W-1} \{\Phi_{(T-W+j,T>} \\ &+ \quad \Phi_{(T-2W+j,T-W+j>} \\ &+ \quad \Phi_{(T-3W+j,T-2W+j>} \\ &+ \ldots + \quad \Phi_{(T-mW+j,T-(m-1)W+j>} \\ &+ \quad \Phi_{<1,T-mW+j>}\} \\ &\leq \quad b \cdot W \cdot OPT_{<1,T>} \end{aligned} \tag{4}$$

The left hand side of the first inequality (4) sums the losses over only a subset of all the windows that are induced by partitioning the time span T using all possible values of j. The final inequality follows by applying (3). ■

To illustrate some of the ideas used in this proof, Figures 3 and 4 provide schematics. Figure 3 shows the windows over which the loss is computed on the left hand side of inequality (4), which is a subset of the set of all windows induced by all possible partitioning of time span T using all possible values of j, which is shown in Figure 3.

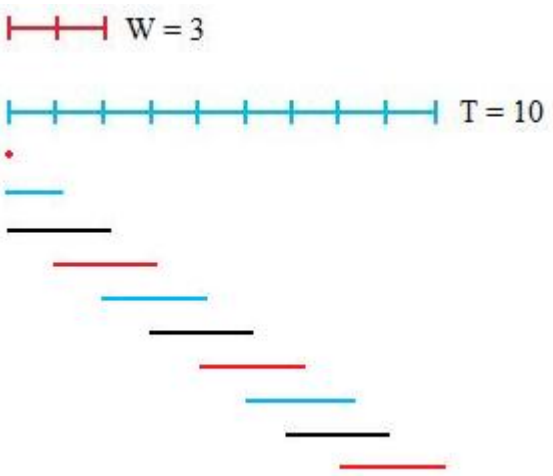

Figure 4: Illustration of the time spans over which the loss is being computed in Equation 4. T = 10, W = 3. Colors red, blue and black correspond to different partitionings, with respect to j, of time span T illustrated in Figure 3.