Algorithm for Online K-Means Clustering

Edo Liberty [*] Ram Sriharsha [†] Maxim Sviridenko [‡]

Abstract

This paper shows that one can be competitive with with the k-means objective while operating online. In this model, the algorithm receives vectors $v_1, \ldots, v_n$ one by one in arbitrary order. For each vector v_i the algorithm outputs a cluster identifier before receiving v_{i+1}. Our online algorithm generates $\tilde{O}(k)$ clusters whose k-means cost is $\tilde{O}(W^*)$ where W^* is the optimal k-means cost using k clusters.[1] We also show that, experimentally, it is not much worse than k-means++ while operating in a strictly more constrained computational model.

1 Introduction

One of the basic and well-studied optimization models in unsupervised Machine Learning is k-means clustering. In this problem we are given the set V of n points (or vectors) in R^m. The goal is to partition V into k sets called clusters $C_1, \ldots, C_k$ and choose one point $z_i \in R^m$ for each cluster C_i to minimize

$$\sum_{i=1}^{k} \sum_{x \in C_i} ||x - z_i||_2^2.$$

In the offline setting where the set of all points is known in advance, Lloyd's algorithm [**?**] provides popular heuristics. It is so popular that practitioners often simply refer to it as k-means. Yet, only recently some theoretical guaranties were proven for its performance on "well clusterable" inputs [**?**]. The k-means++ [**?**] algorithm provides an expected $O(\log(k))$ approximation or an efficient seeding algorithm. A well know theoretical algorithm is due to Kanungo et al. [**?**]. It gives a constant approximation ratio and is based on local search ideas popular in the related area of design and analysis of algorithms for facility location problems (e.g. [**?**]).

In the standard streaming model the algorithm must consume the date in one pass and is allowed to keep only a small (typically constant or logarithmic) amount of information. Nevertheless, it must output its final decisions when the stream has ended. For example, the location of the centers for k-means. Such model was considered by Ailon et al. [**?**]. In such a model, the best assignment of points to the centers chosen by the algorithm is only known in hindsight.

In contrast, the online model of computation does not allow to postpone clustering decisions. In this setting, an a priori unknown number of points arrive one by one in an arbitrary order. When a new point arrives the algorithm must either put it in one of the existing clusters or open a new cluster (consisting of a single point). Note that this problem is conceptually non trivial even if one could afford unbounded computational power at every iteration. This is because the quality of current choices depend on the unknown (yet unseen) remainder of the stream.

In this paper, we consider the very restricted setting in the intersection of these two models. We require the algorithm outputs a single cluster identifier for each point online while using space and time at most poly-logarithmic in the length of the stream. This setting is harder than the streaming model. On the one hand, any space efficient online algorithm is trivially convertible to a streaming algorithm. One could trivially keep sufficient statistics for each cluster such that the centers of mass could be computed at the end of the stream. The computational and space overhead are independent of the length of the stream. On the other hand, the online problem resists approximation even for the case of 3 points in one dimension and $k = 2$. Consider the stream of one dimensional vectors (scalars) $x_1 = 0$, $x_2 = 1$ and $x_3 = c$ or $x_3 = 1/c$ for some $c \gg 1$. If the online algorithm

[*] Yahoo Labs, New York, NY

[†] Yahoo Labs, Sunnyvale, CA

[‡] Yahoo Labs, New York, NY

[1] The notation $\tilde{O}(\cdot)$ depresses poly-logarithmic factors.

separates x_1 and x_2 its cost is potentially $\Omega(c)$ but the optimal solution has cost $O(1)$. If the online algorithm puts x_1 and x_2 in the same cluster it exhibits cost in $\Omega(1)$. But, the optimal cost is potentially $O(1/c)$.

In the context of machine learning, the results of k-means were shown to provide powerful unsupervised features [**?**] on par, sometimes, with neural nets for example. This is often referred to as (unsupervised) feature learning. Intuitively, if the clustering captures most of the variability in the data, assigning a single label to an entire cluster should be pretty accurate. It is not surprising therefore that cluster labels are powerful features for classification. In the case of online machine learning, these cluster labels must also be assigned online. The importance of such an online k-means model was already recognized in machine learning community [**?**, **?**].

A closely related family of optimization problems is known as facility location problems. Two standard variants are the uncapacitated facility location problem (or the simple plant location problem in the Operations Research jargon) and the k-median problem. These problems are well-studied both from computational and theoretic viewpoints (a book [**?**] and a survey [**?**] provide the background on some of the aspects in this area). Meyerson [**?**] suggested a simple and elegant algorithm for the online uncapacitated facility location with competitive ratio of $O(\log n)$. Fotakis [**?**] suggested a primal-dual algorithm with better performance guarantee of $O(\log n/ \log\log n)$. Anagnostopoulos et al. [**?**] considered a different set of algorithms based on hierarchical partitioning of the space and obtained similar competitive ratios. The survey [**?**] summarizes the results in this area.

In this work we consider two online k-means models. In the *semi-online* model we assume that we know the total number of vectors that we are going to process. And, a lower bound w^* for the total optimal cost of k-means W^*. We design an algorithm (Algorithm 1) that finds an approximate solution with at most $O(k \log n \log(W^*/w^*))$ clusters in expectation and has an expected objective value $O(W^*)$.

In the pure online model we do not assume any prior knowledge. Our algorithm (Algorithm 2) opens a comparable number of clusters as in the semi-online model but the objective function quality of the solution could be worse by a $\log n$-factor. From the practical viewpoint having an estimate for the total number of input vectors is a reasonable assumption. The performance of the semi-online algorithm will degrade significantly only if such an estimate is wrong by many orders of magnitude.

2 Online and Semi-Online Algorithms for k-means

The algorithm uses ideas from the online facility location algorithm of Meyerson [**?**]. The intuition is as follows; think about k-means and a facility location problem where the service costs are squared Euclidean distances. For the facility cost, start with f_1 which is known to be too low. By doing that the algorithm is "encouraged" to open many facilities (centers) which keeps the service costs low. If the algorithm detect that too many facilities were opened, it can conclude that the current facility cost is too low. It therefore doubles the facility cost of opening future facilities (centers). It is easy to see that the facility cost cannot be doubled many times without making opening new clusters prohibitively expensive. In Algorithm 2 we denote the distance of a point v to a set C as $D(v, C) = \min_{c \in C} \|v - c\|$. As a convention, if $C = \emptyset$ then $D(v, C) = \infty$ for any v.

Algorithm 1 semi-online k-means algorithm

```
input: $k$, $v_1, \ldots, v_n$, $w^*$, $n$
$C \leftarrow \emptyset$;  $r \leftarrow 1$;  $q_1 \leftarrow 0$
$f_1 \leftarrow w^*/k \log(n)$
for $i \in 1, \ldots, n$ do
    with probability $p_i = \min(D^2(v_i, C)/f_r, 1)$
        $C \leftarrow C \cup \{v_i\}$
    if $q_r \geq 3k(1 + \log_2(n))$ then
        $r \leftarrow r + 1$;  $f_r \leftarrow 2 \cdot f_{r-1}$;  $q_r \leftarrow 0$
    yield: $c = \arg\min_{c \in C} \|v_i - c\|^2$
```

3 Analysis

Consider some optimal solution consisting of clusters $C_1^*, \ldots, C_k^*$ with cluster centers $z_1^*, \ldots, z_k^*$. Let

$$W_i^* = \sum_{v \in C_i^*} ||v - z_i^*||_2^2$$

Algorithm 2 Online k-means algorithm

```
input: $k, v_1, \ldots, v_n$
$C \leftarrow \emptyset$;   $r \leftarrow 1$;   $q_1 \leftarrow 0$
for $i \in k'$     (for some $k' > k$) do
    $C \leftarrow C \cup \{v_i\}$
    yield: $c = i$
$w^* \leftarrow$ the best $k$-means cost on $v_1, \ldots, v_{k'}$
$f_1 \leftarrow w^*/k$
for $i \in k'+1, \ldots, n$ do
    with probability $p_i = \min(D^2(v_i, C)/f_r, 1)$
        $C \leftarrow C \cup \{v_i\}$
    yield: $c = \arg\min_{c \in C} \|v_i - c\|^2$
    if $q_r \geq 3k(1 + \log_2(i))$ then
        $r \leftarrow r + 1$;   $f_r \leftarrow 2 \cdot f_{r-1}$;   $q_r \leftarrow 0$
```

be the cost of the i-th cluster in the optimal solution and $W^* = \sum_{i=1}^{k} W_i^*$ be the value of the optimal solution. Let $A_i^* = W_i^*/|C_i^*|$ be the average distance to the cluster center from the vectors in the i-th optimal cluster. We define a partition of the cluster C_i^* into subsets that we call *rings*:

$$S_{0,i} = \{v \in C_i^* : ||v - z_i^*||_2^2 \leq A_i^*\}, \tag{1}$$

$$S_{\tau,i} = \left\{v \in C_i^* : ||v - z_i^*||_2^2 \in (2^{\tau-1}A_i^*, 2^{\tau}A_i^*]\right\} \text{ for } \tau \geq 1. \tag{2}$$

Note that $S_{\tau,i} = \emptyset$ for $\tau > \log_2(|C_i^*|)$ because otherwise the average squared distances is larger than A_i^*. For any two vectors $v, v' \in S_{\tau,i}$ we have

$$||v - v'||_2^2 \leq 2||v - z_i||_2^2 + 2||v' - z_i||_2^2 \leq 4 \cdot 2^{\tau} A_i^*. \tag{3}$$

Let $c_{min} = f_1$ be the smallest non-zero squared distance between two vectors in our problem instance and $c_{max} = \max_{i,\tau} \max_{v \in S_{\tau,i}} ||v - z_i||_2^2$ is the maximal distance between a vector and a cluster center in the optimal solution. Let R denote the index of the last phase of our algorithm. First we estimate the expected number of clusters opened after certain phase.

Lemma 1 *The expected number of clusters opened during phases $r', r'+1, \ldots, R$ by the online or the semi-online versions of our algorithm is at most*

$$k(1 + \log_2 n) + \frac{24W^*}{f_{r'}}.$$

Proof. We upper bound the expected number of clusters initiated by vectors in the ring $S_{\tau,i}$ during phases $r', \ldots, R$ by

$$1 + \sum_{r \geq r'} \frac{4 \cdot 2^{\tau} A_i^*}{f_r} |S_{\tau,i}|$$

because once we open the first cluster with center at some $v \in S_{\tau,i}$, the probability to open a cluster for each subsequent vector $v' \in S_{\tau,i}$ during phase r is upper bounded by

$$\frac{||v - v'||_2^2}{f_r} \leq \frac{4 \cdot 2^{\tau} A_i^*}{f_r}$$

by inequality (3). Therefore the expected number of clusters opened by the vectors in C_i^* after phase r' is at most

$$\begin{aligned}
\sum_{\tau=0}^{\log_2 n} \left(1 + \sum_{r \geq r'} \frac{4 \cdot 2^{\tau} A_i^*}{f_r} |S_{\tau,i}|\right) &\leq 1 + \log_2 n + 8\frac{A_i^* |S_{0,i}|}{f_{r'}} + 8 \sum_{\tau=1}^{\log_2 n} \frac{2^{\tau} A_i^* |S_{\tau,i}|}{f_{r'}} \\
&\leq 1 + \log_2 n + 8\frac{W_i^*}{f_{r'}} + 16 \sum_{\tau=1}^{\log_2 n} \frac{2^{\tau-1} A_i^* |S_{\tau,i}|}{f_{r'}} \\
&\leq 1 + \log_2 n + \frac{24 W_i^*}{f_{r'}}.
\end{aligned}$$

Summing up over all $i = 1, \dots, k$ we obtain that the statement of lemma. □

The next lemma is similar to Lemma 1, it will be used in the analysis of the pure online algorithm only.

Lemma 2 *Consider a phase r' of our online algorithm. Let $n_{r'}$ be the total number of vectors processed by the online algorithm up to and including phase r'. The number of clusters opened during phase r' is a sum of two random numbers $p_1(r') + p_2(r')$ where $p_1(r') \leq k(1 + \log_2 n_{r'})$ and $E[p_2(r')] \leq \frac{24W^*}{f_{r'}}$.*

Proof. The proof is identical to the proof of the Lemma 1, the only difference is that the number of rings $p_1(r')$ is always upper bounded by $\log_2 n_{r'}$ instead of $\log_2 n$. The second term

$$p_2(r') = \sum_{\tau=0}^{\log_2 n} \sum_{r \geq r'} \frac{4 \cdot 2^\tau A_i^*}{f_r} |S_{\tau,i}|$$

is upper bounded in expectation as in the previous lemma. □

Theorem 1 *Let q be the number of clusters defined by our algorithm (online or semi-online). Then $E[q] = O\left(k \log_2 n \log_2 \frac{nc_{max}}{c_{min}}\right)$.*

Proof. Consider the first phase r' of our algorithm when

$$f_{r'} \geq \frac{W^*}{k \log_2 n}.$$

The total number of clusters opened before phase r' is upper bounded by $3k(1 + \log_2 n) \log_2 \frac{f_{r'}}{f_1}$. Combining that with Lemma 1 we derive that the expected number of clusters opened by our online algorithm is at most

$$k(1 + 25 \log_2 n) + 3k(1 + \log_2 n) \log_2 \frac{f_{r'}}{f_1} = O\left(k \log_2 n \log_2 \frac{nc_{max}}{c_{min}}\right).$$

□

Before we proceed estimating the expected cost of clusters opened by our online algorithm we prove the following technical lemma.

Lemma 3 *We are given a sequence $X_1, \dots, X_n$ of n independent boolean random variables with expectations $p_i \geq \min\{A_i/B, 1\}$ where $B \geq 0$ and $A_i \geq 0$ for $i = 1, \dots, n$. We sample the boolean variables sequentially from $X_1, \dots, X_n$. Let t be the random index of the first boolean random variable producing one or $t = n + 1$ if all of them are equal to zero then*

$$E\left[\sum_{i=1}^{t-1} A_i\right] \leq B.$$

Proof. Let T be the first index such that $p_T = 1$. If there is no such index then $T = n + 1$.

Consider the boolean variable X_i for $i = 1, \dots, T - 1$. The probability that it contributes to the sum $E\left[\sum_{i=1}^{t-1} A_i\right]$ is upper bounded by $\prod_{j=1}^{i}\left(1 - \frac{A_j}{B}\right)$. Therefore, by linearity of expectation we have

$$\begin{aligned} E\left[\sum_{i=1}^{t-1} A_i\right] &\leq \sum_{i=1}^{T-1} A_i \prod_{j=1}^{i}\left(1 - \frac{A_j}{B}\right) \\ &= B \sum_{i=1}^{T-1} \frac{A_i}{B} \prod_{j=1}^{i}\left(1 - \frac{A_j}{B}\right) \\ &\leq B \sum_{i=1}^{T-1} \frac{A_i}{B} \prod_{j=1}^{i-1}\left(1 - \frac{A_j}{B}\right) \\ &\leq B. \end{aligned}$$

□

Lemma 4 *The expected service cost for each ring $S_{\tau,i}$ in both online and semi-online algorithms is upper bounded by*

$$f_R + 4 \cdot 2^\tau A_i^* \cdot |S_{\tau,i}|$$

where R is the total number of rounds for our algorithm.

Proof. After one of the vectors in $S_{\tau,i}$ opens a new cluster each vector in this ring pays at most $4 \cdot 2^\tau A_i^*$ in expectation. Before that, we could view the service cost accumulation as the process in Lemma 3 where for each new vector v_j we either open a new cluster with probability at least $\min\{d_j^2/f_R, 1\}$ and incur cost 0 or put it in the existing cluster and incur the cost of d_j^2. Combining we derive that the expected service costs for vectors in $S_{\tau,i}$ is upper bounded by $f_R + 4 \cdot 2^\tau A_i^* \cdot |S_{\tau,i}|$. □

Theorem 2 *Let W be the cost of the approximate solution obtained by our semi-online algorithm. Then $E[W] = O(1)\, W^*$.*

Proof. Consider the first phase r'' of the algorithm such that

$$f_{r''} \geq \frac{72W^*}{k(1+\log_2 n)}.$$

By Lemma 1 the expected number of clusters opened after the start of the phase r'' is at most $k(1+\log_2 n) + \frac{24W^*}{f_{r''}} \leq \frac{4}{3}k(1+\log_2 n)$. By Markov's inequality the probability to open more than $3k(1+\log_2 n)$ clusters is at most $4/9$. Therefore, with probability at least $5/9$ the algorithm will stop at phase r''.

By Lemma 4 the expected service costs for vectors in $S_{\tau,i}$ is upper bounded by $f_R + 4 \cdot 2^\tau A_i^* \cdot |S_{\tau,i}|$. Summing up over all the rings we derive and upper bound of $f_R k(1+\log n) + 12W^*$. We now estimate $E[f_R]$. Let p be the probability that our algorithm terminates before round r''. Since the probability of concluding the execution at each of the rounds after r'' is at least $5/9$ we derive an upper bound

$$\begin{aligned} E[f_R] &\leq p f_{r''-1} + (1-p) \sum_{r=r''}^{+\infty} f_r \cdot \frac{5}{9} \cdot \left(\frac{4}{9}\right)^{r-r''} \\ &= p f_{r''-1} + (1-p) f_{r''} \cdot \frac{5}{9} \sum_{i=0}^{+\infty} 2^i \cdot \left(\frac{4}{9}\right)^i \\ &\leq 5 f_{r''} \leq 5 \cdot 2 \cdot \frac{72W^*}{k(1+\log_2 n)}. \end{aligned}$$

Therefore, the total expected cost of the approximate solution is upper bounded by $732W^* = O(1)W^*$. □

Theorem 3 *Let W be the cost of the approximate solution obtained by our online algorithm. Then $E[W] = O(\log_2 n)\, W^*$.*

Proof. Consider the first phase r'' of the algorithm such that

$$f_{r''} \geq \frac{72W^*}{k}.$$

By Lemma 2 the number of clusters opened during the phase $r \geq r''$ is a sum of two random numbers $p_1(r)+p_2(r)$ where $p_1(r) \leq k(1+\log_2 n_r)$ and

$$E[p_2(r)] \leq \frac{24W^*}{f_r} \leq \frac{k}{3 \cdot 2^{r-r''}}.$$

The algorithm can open $3k(1+\log_2 n_r)$ clusters during round r only if $p_2(r'') \geq 2k(1+\log_2 n_r) > 2k$. By Markov's inequality the probability to open $3k(1+\log_2 n_r)$ clusters on iteration $r \geq r''$ is at most $\frac{1}{6 \cdot 2^{r-r''}} \leq \frac{1}{6}$.

By Lemma 4 the expected service costs for vectors in the ring $S_{\tau,i}$ is upper bounded by $f_R + 4 \cdot 2^\tau A_i^* \cdot |S_{\tau,i}|$. Summing up over all the rings we derive and upper bound of $f_R k(1+\log n) + 12W^*$.

We now estimate $E[f_R]$. Let p be the probability that our algorithm finishes before round r''. We have

$$\begin{aligned}
E[f_R] &\leq pf_{r''-1} + (1-p)\sum_{r=r''}^{+\infty} f_r \cdot \frac{5}{6} \cdot \left(\frac{1}{6}\right)^{r-r''} \\
&= pf_{r''-1} + (1-p)f_{r''} \cdot \frac{5}{6}\sum_{i=0}^{+\infty} 2^i \cdot \left(\frac{1}{6}\right)^{i} \\
&\leq 5f_{r''}/4 \leq \frac{5}{2} \cdot \frac{72W^*}{k}.
\end{aligned}$$

Therefore, the total expected cost of the approximate solution is upper bounded by $O(\log_2 n)W^*$.

□

4 Experimental Analysis of the Algorithm

4.1 Practical modifications to the algorithm

While experimenting with the algorithm, we discovered that some log factors were, in fact, too pessimistic in practice. We also had to make some pragmatic decisions about, for example, how to set the initial facility cost lower bound. As another practical adjustment we introduce the nation of k_{target} and k_{actual}. The value of k_{target} is the number of clusters we would like the algorithm to output while k_{actual} is the actual number of clusters generated. Internally, the algorithm operates with a value of $k = \lceil (k_{target} - 15)/5 \rceil$. This is a heuristic ad-hoc conversion that compensates for the k_{actual} being larger that k by design.

Algorithm 3 Heuristic online k-means, practical but unprovable.

```
input: $k_{target}, v_1, \ldots, v_n$
$C \leftarrow \emptyset$;  $r \leftarrow 1$;  $q_1 \leftarrow 0$
$k \leftarrow \lceil (k_{target} - 15)/5 \rceil$
$k' \leftarrow k + 10$
for $i \in 1, \ldots, k'$ do
    $C \leftarrow C \cup \{v_i\}$
$D_{\min} = \{D^2(v_i, C \setminus \{i\}) | v_i \in C\}$
$f_r \leftarrow$ half the sum of the $k' - k$ smallest elements in $D_{\min}$.
for $i \in k' + 1, \ldots, n$ do
  with probability $p_i = \min(D^2(v_i, C)/f_r, 1)$
    $C \leftarrow C \cup \{v_i\}$
  Assign $v_i$ the cluster center $c = \arg\min_{c \in C} \|v_i - c\|^2$.
  if $q_r \geq k$ then
    $r \leftarrow r + 1$;  $f_r \leftarrow 10 \cdot f_{r-1}$;  $q_r \leftarrow 0$
$k_{actual} \leftarrow |C|$
```

4.2 Datasets

To evaluate our algorithm we executed it on 12 different datasets. All the datasets that we used are conveniently aggregated on the Libsvm website [?] and on the UCI dataset collection [?] While they are originally organized as classification benchmarks, we simply striped the label and used the vectors for clustering. Some more information is given in Table 1. For each dataset, "nnz" gives to total number of non zero features in the dataset, n the number of examples and d the dimension of the problem.

Feature engineering for the sake of online learning is one of the motivations for this work. For that reason, we apply standard stochastic gradient descent linear learning with the squared loss on these data. Once with the raw features and once with the k-means features added. In some cases we see a small decrease in accuracy due to slower convergence of the learning on a larger feature set. This effect should theoretically be nullified in the presence of more data. In other case, however, we see a significant uptick in classification accuracy. This is agreement with the observations of Coates et al. [?].

dataset	nnz	n	d	classification accuracy with raw features	classification accuracy with added k-means features	Sparse?
20news-binary	2.44E+6	1.88E+4	6.12E+4	0.9532	0.9510	yes
adult	5.86E+5	4.88E+4	1.04E+2	0.8527	0.8721	yes
ijcnn1	3.22E+5	2.50E+4	2.10E+1	0.9167	0.9405	no
letter	2.94E+5	2.00E+4	1.50E+1	0.7581	0.7485	no
magic04	1.71E+5	1.90E+4	9.00E+0	1.0000	1.0000	no
maptaskcoref	6.41E+6	1.59E+5	5.94E+3	0.8894	0.8955	yes
nomao	2.84E+6	3.45E+4	1.73E+2	0.5846	0.5893	no
poker	8.52E+6	9.47E+5	9.00E+0	0.5436	0.6209	no
shuttle	2.90E+5	4.35E+4	8.00E+0	0.9247	0.9973	no
skin	4.84E+5	2.45E+5	2.00E+0	0.9247	0.9988	no
vehv2binary	1.45E+7	2.99E+5	1.04E+2	0.9666	0.9645	no
w8all	7.54E+5	5.92E+4	2.99E+2	0.9638	0.9635	yes

Table 1: The table gives some basic information about the datasets we experimented with. The column under nnz gives the number of non zero entries in the entire dataset, n the number of vectors and d their dimension. The column under "sparse?" simply indicates whether nnz is much smaller that $n \cdot d$.

4.3 The number of online clusters

One of the artifacts of applying our online k-means algorithm is that the number of clusters is not exactly known a priory. But as we see in Figure 1, the number of resulting clusters is rather predictable and controllable. Figure 1 gives the ratio between the number of clusters outputted by the algorithm, k_{actual}, and the specified target k_{target}. The results reported are mean values of 3 runs for every parameter setting. The observed standard deviation of k_{actual} is typically in the range $[0, 3]$ and never exceeded $0.1 \cdot k_{target}$ in any experiment. Figure 1 clearly shows that the ratio k_{actual}/k_{target} is roughly constant and close 1.0. Interestingly, the main differentiator is the dataset itself and not the value of k_{target}.

4.4 Online clustering cost

Throughout this section, we measure the online k-means clustering cost with respect to different baselines. We report averages of at least 3 different independent executions for every parameter setting. In Figure 2 the reader can see the online k-means clustering cost for the set of centers chosen online by our algorithm for different values of k_{target} and different datasets. For normalization, each cost is divided by f_0, the sum of squares of all vector norms in the dataset (akin to the theoretical k-means cost of having one center at the origin). Note that some datasets are inherently unclusterable. Even using many cluster centers, the k-means objective does not decrease substantially. Nevertheless, as expected, the k-means cost obtained by the online algorithm, f_{online}, decreases as a function of k_{target}.

The monotonicity of f_{online} with respect to k_{target} is unsurprising. In Figure 3 we plot the ratio f_{online}/f_{random} as a function of k_{target}. Here, f_{random} is the sum of squared distances of input points to k_{actual} input points chosen uniformly at random (as centers). Note that in each experiment the number of clusters used by the random solution and online k-means is identical, namely, k_{actual}. Figure 3 illustrates something surprising. The ratio between the costs remains relatively fixed per dataset and almost independent to k_{target}. Put differently, even when the k-means cost is significantly lower than picking k random centers, they improve in similar rates as k grows.

The next experiment compares online k-means to k-means++. For every value of k_{target} we ran online k-means to obtain both f_{online} and k_{actual}. Then, we invoke k-means++ using k_{actual} clusters and computed its cost, f_{kmpp}. This experiment was repeated 3 times for each dataset and each value of k_{target}. The mean results are reported in Figure 4. Unsurprisingly, k-means++ is usually better in terms of cost. But, the reader should keep in mind that k-means++ is an *offline* algorithm that requires k passes over the data compared with the online computational model of our algorithm.

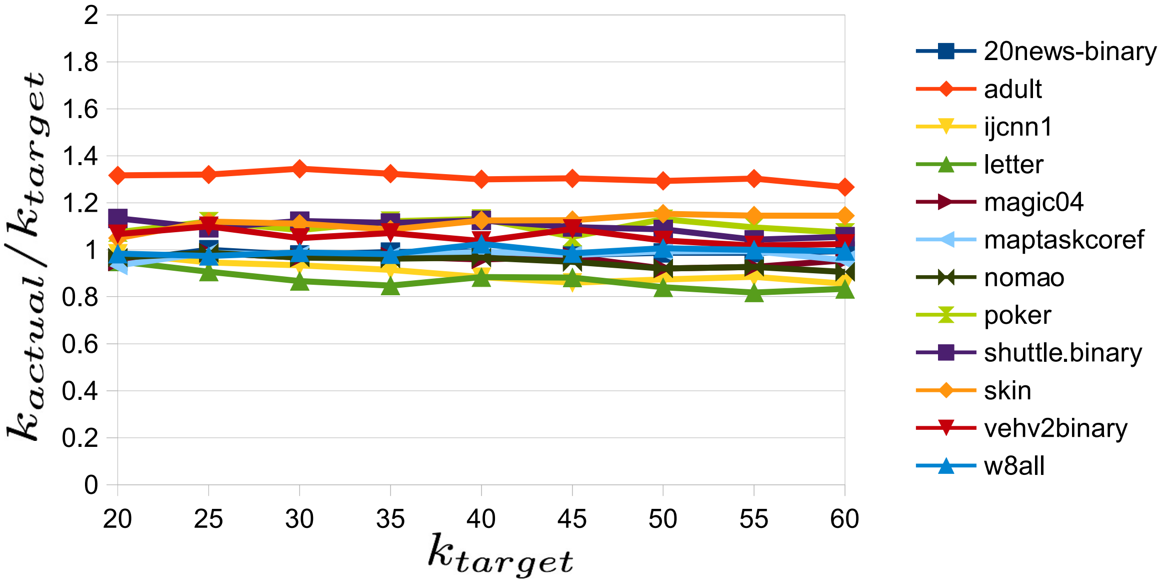

Figure 1: The figure gives the ratio k_{actual}/k_{target} on the y-axis as a function of k_{target} on the x-axis. The value k_{target} is given to the algorithm as input and k_{actual} is the resulting cardinality of the center set C. We clearly see that this ratio is roughly constant and close 1. Interestingly, the main differentiator is the dataset itself and not the value of k_{target}.

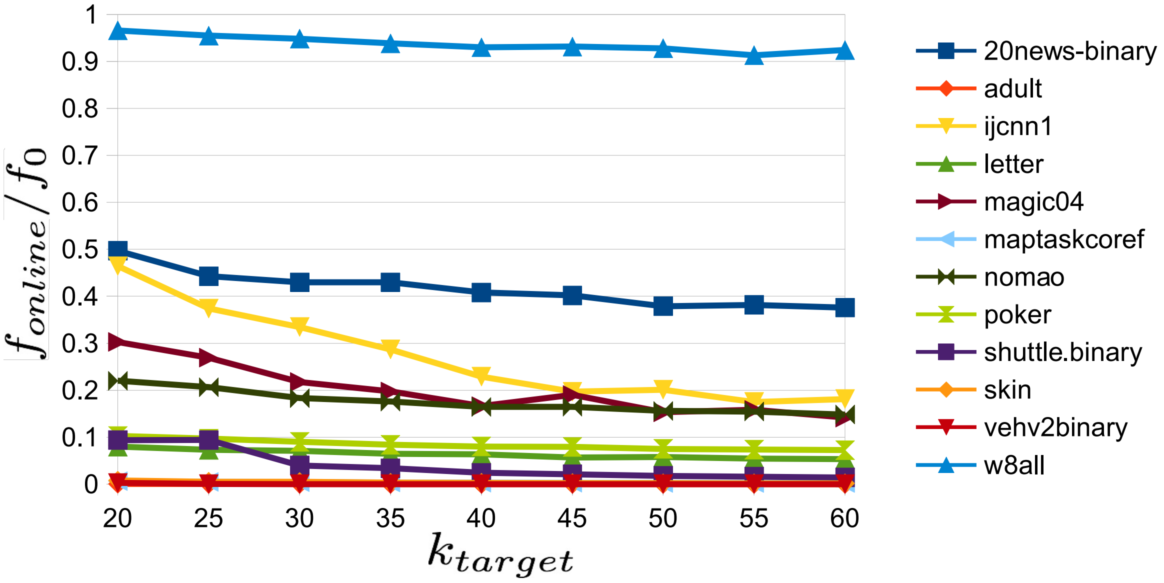

Figure 2: Online k-means clustering cost (f_{online}) as a sanction of k_{target} for the different datasets. For normalization, each cost is divided by f_0, the sum of squares of all vector norms in the dataset (akin to the k-cost of once center in the origin).

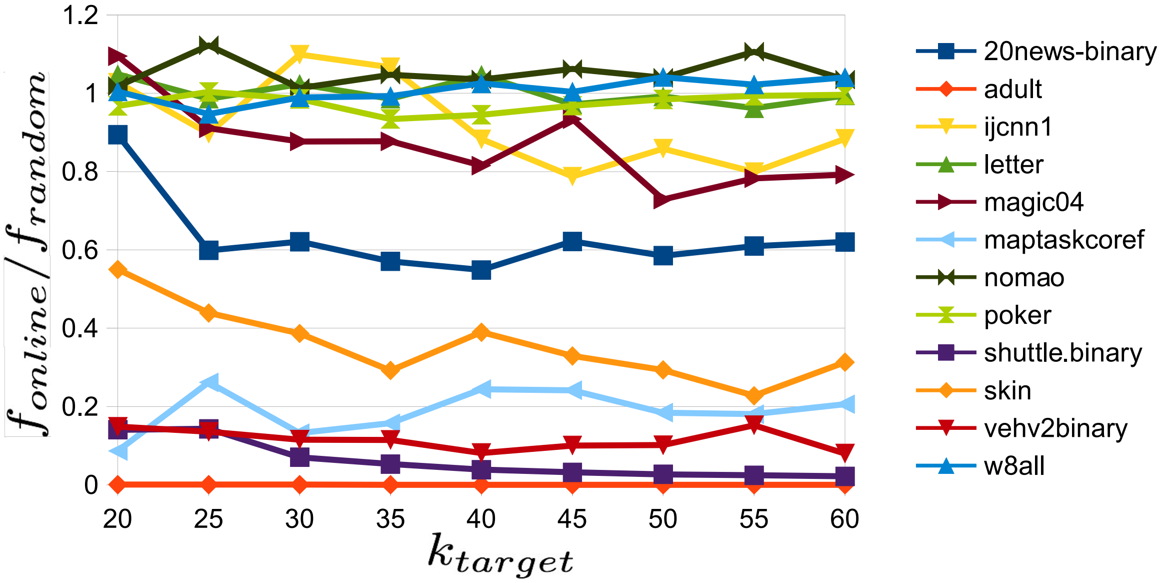

Figure 3: On the y-axis, the value of f_{online} divided by f_{random}. The latter is the cost of choosing, uniformly at random, as many cluster centers (from the data) as the online algorithm did. A surprising observation is that this ratio is almost constant for each dataset and almost independent of k_{target} (on the x-axis).

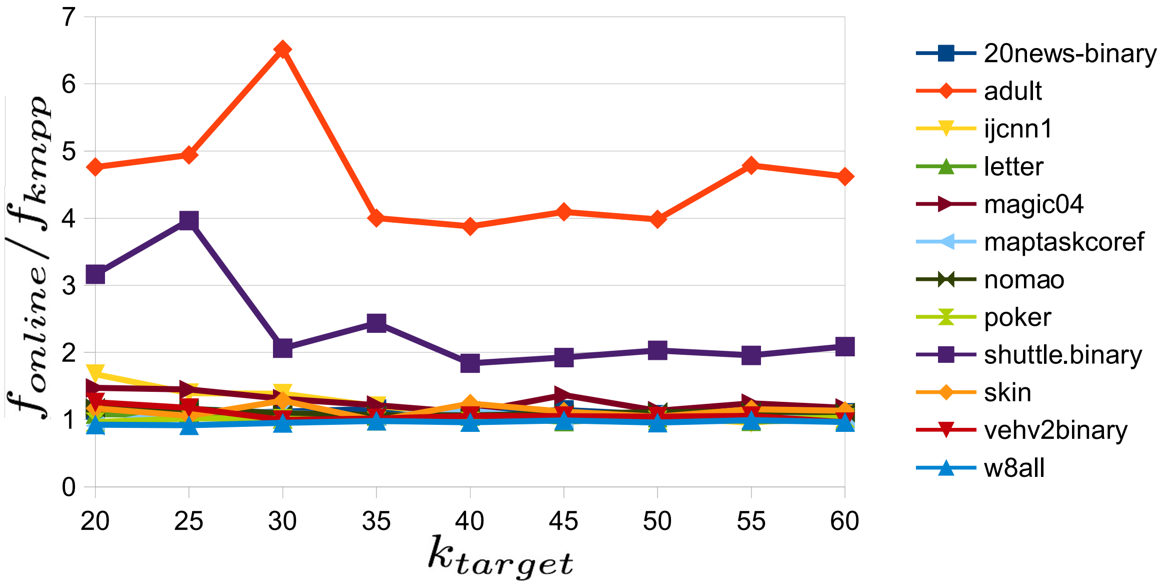

Figure 4: On the y-axis we plot f_{online}/f_{kmpp} as a function of k_{target} on the x-axis. The values of f_{online} is the cost of running Algorithm 3 with parameter k_{target}. The value of f_{kmpp} is the cost of running k-means++ with k_{actual} clusters, k_{actual} is the number of clusters online k-means actually used. We see that, except for the datasets *adult* and *shuttle.binary*, k-means++ and online k-means are comparable. For *adult* and *shuttle.binary* online k-means is worse by a small constant factor. Note (Figure 3) that both *adult* and *shuttle.binary* are datasets for which online k means is dramatically better than random.

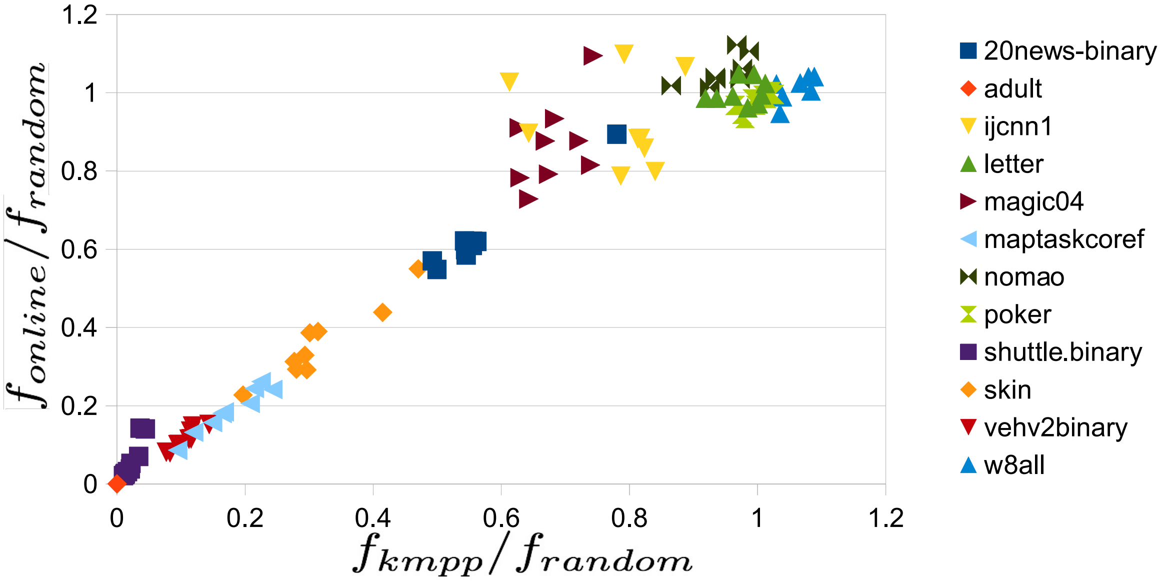

Figure 5: On the x-axis f_{kmpp}/f_{random} both using k_{actual} clusters. The value of k_{actual} is obtained by running online k-means with input k_{target} on the x-axis. The y-axis depicts f_{online}/f_{random}. Note that the performance of k-means++ and online k-means are very similar almost everywhere. The advantage of k-means++ (see Figure 4) occurs when the clustering cost is a minuscule fraction of random clustering.

5 Aknowledgements

We would like to thank Dean Foster for helping us with the proof of the Lemma 3.

A Online and Semi-Online Algorithms for k-means

Our algorithm will run in phases, the phases will be indexed by $r \geq 1$. On each phase let f_r denote the parameter used as a cost gauge for opening a new cluster. Initially f_1 is defined to be the smallest non-zero squared distance between two vectors in our problem instance and $f_r = 2 \cdot f_{r-1}$ for $r \geq 2$. The total number of demand points (or vectors) is n. Each input vector is indexed by $j = 1, \ldots, n$.

When a new vector $v_j \in R^m$ arrives the algorithm either decides to put it into an existing cluster or opens a new cluster centered at $z = v_j$. For $j = 1$ the algorithm always opens the first cluster centered at v_1.

When a new vector v_j for $j \geq 2$ arrives, we run a generic step for the algorithm:

1. Let $C_1, \ldots, C_q$ be the existing partition of the index set $\{1, \ldots, j-1\}$ into clusters. Let $z_1, \ldots, z_q$ be the corresponding cluster centers. Let $r \geq 1$ be the current phase of the algorithm. For each phase, let q_r be the number of clusters opened during that phase. Note $q = \sum_{i=1}^{r} q_r$.

2. Compute d_j^2 the squared Eucledian distance from the nearest cluster center to v_j, let $i(j)$ be the index of the closest cluster center.

3. Set
$$p = \min\left\{\frac{d_j^2}{f_r}, 1\right\}$$
and define a Boolean random variable Q with $E[Q] = p$.

4. If $Q = 1$ we open a new cluster $C_{q+1} = \{v_j\}$ centered at $z_{q+1} = v_j$, increment $q = q + 1$, $q_r = q_r + 1$.

5. If $Q = 0$ we assign vector v_j into the cluster indexed by $i(j)$, i.e. $C_{i(j)} = C_{i(j)} \cup \{v_j\}$.

6. (a) In the semi-online algorithm: If $q_r \geq 3k(1 + \log_2 n)$ we end the phase r and move to the next one, i.e. $r = r + 1$, the cluster cost is doubled and $q_r = 0$.

   (b) In the fully online algorithm: If $q_r \geq 3k(1 + \log_2 j)$ we end the phase r and move to the next one, i.e. $r = r + 1$, the cluster cost is doubled and $q_r = 0$.