15.097 Prediction Project Report

Online k-Means Clustering of Nonstationary Data

Angie King

May 17, 2012

1 Introduction and Motivation

Machine learning algorithms often assume we have all our data before we try to learn from it; however, we often need to learn from data as we gather it over time. In my project I will focus on the case where we want to learn by clustering. The setting where we have all the data ahead of time is called "batch clustering"; here we will be interested in "online clustering." Specifically, we will investigate algorithms for online clustering when the data is non-stationary.

Consider a motivating example of a t-shirt retailer that receives online data about their sales. The retailer sells women's, men's, girls' and boys' t-shirts. Fashion across these groups is different; the colors, styles, and prices that are popular in women's t-shirts are quite different from those that are popular in boys' t-shirts. Therefore the retailer would like to find the "average" set of characteristics for each demographic in order to market appropriately to each. However, as fashion changes over time, so do the characteristics of each demographic. At every point in time, the retailer would like to be able to segment its market data into correctly identified clusters and find the appropriate current cluster average.

This is a problem of online clustering of non-stationary data. More formally, consider d-dimensional data points which arrive online and need to be clustered. While the cluster should remain intact, its center essentially shifts in d-space over time.

In this paper I will argue why the standard k-means cost objective is unsuitable in the case of non-stationary data, and will propose an alternative cost objective. I will discuss a standard algorithm for online clustering, and its shortcomings when data is non-stationary. I will introduce a simple variant of this algorithm which takes into account nonstationarity, and will compare the performance of these algorithms with respect to the optimal clustering for a simulated data set. I will then discuss performance guarantees, and provide a practical, rather than a theoretical, way of measuring an online clustering algorithm's performance.

2 Background and Literature Review

Before we delve into online clustering of time-varying data, we will build a baseline for this problem by providing background and reviewing relevant literature.

As in the t-shirt example, the entire data set to be processed may not be available when we need to begin learning; indeed, the data in the t-shirt example is not even finite. There are several modes in which the data may be available and we now define precisely what those are. In batch mode, a finite dataset is available from the beginning. This is the most commonly analyzed setting. Streaming mode refers to a setting where there is a finite amount of data to be processed but it arrives one data point at a time, and the entire dataset cannot be stored in memory. The online setting, which we will be concerned with here, departs from the previous two settings in that the there is an infinite amount of data to be analyzed, and it arrives one data point at a time. Since there is an infinite amount of data, it is impossible to store all the data in memory.

When clustering data in the batch setting, several natural objectives present themselves. The "k-center" objective is to minimize the maximum distance from any point to its cluster's center. The "k-means" objective is to minimize the mean squared distance from all points to their respective cluster centers. The "k-median" objective is to minimize the distance from all points to their respective cluster centers. Depending on the data being analyzed, different objectives are appropriate in different scenarios.

In our case we will focus on the k-means objective. Even in the batch setting, finding the optimal k-means clustering is an NP-hard problem [1]. Therefore heuristics are often used. The most common heuristic is often simply called "the k-means algorithm," however we will refer to it here as Lloyd's algorithm [7] to avoid confusion between the algorithm and the k-clustering objective.

In the batch setting, an algorithm's performance can be compared directly to the optimal clustering as measured with respect to the k-means objective. Lloyd's algorithm, which is the most commonly used heuristic, can perform arbitrarily badly with respect to the cost of the optimal clustering [8]. However, there exist other heuristics for the k-means objective which have a performance guarantee; for example, k-means++ is a heuristic with an $O(\log k)$-approximation for the cost of the optimal clustering [2]. Another algorithm, developed by Kanungo et al. in [6], gives a constant factor 50-approximation to the cost of the optimal clustering.

However, in the online setting, it is much less clear how to analyze an algorithm's performance. In [5], Dasgupta proposed two possible methods for evaluating the performance of an online clustering scheme. The first is an approximation algorithm style bound: if at time t an algorithm sees a new data point x_t and then outputs a set of k centers Z_t then an approximation algorithm would find some $\alpha \geq 1$ such that $\text{Cost}(Z_t) \leq \alpha$ OPT, where OPT is the cost for the best k centers for $x_1, ..., x_t$.

Another type of performance metric is regret, which is a common framework in online learning. Under a regret framework, an algorithm would announce a set of centers Z_t at time t, then recieve a new data point x_t and incur a loss for that point equal to the squared distance from x_t to its closest center in Z_t. Then the cumulative regret up to time t is given by

$$\sum_{\tau \leq t} \min_{z \in Z_\tau} ||x_\tau - z||_2^2$$

and we can compare this to the regret that would be achieved under the best k centers for $x_1, .., x_t$.

Dasgupta acknowledges that "it is an open problem to develop a good online algorithm for k-means clustering" [5] under either of these performance metrics. Indeed, although several online algorithms exist, almost nothing theoretical is known about performance.

There is a standard online variant of Lloyd's algorithm which we will describe in detail in Section 4; like Lloyd's algorithm in the batch case it can perform arbitrarily badly under either of the performance frameworks outlined above.

The only existing work that proves performance guarantees for online k-means clustering is by Choromanska and Monteleoni [4]. They consider a traditional online learning framework where there are experts that the algorithm can learn from. In their setting, the experts are batch algorithms with known performance guarantees. To get an approximation guarantee in the online sense, they use an analog to Dasgupta's concept of regret defined above [5] and prove the approximation bounds by comparing the online performance to the batch performance, which has a known performance guarantee with respect to the cost of the optimal clustering.

Although [4] is the only paper which provides performance bounds for online clustering, there is a growing literature in this area which focuses on heuristics for particular applications. In [3], rather than trying to minimize the k-means objective, Barbakh and Fyfe consider how to overcome the sensitivity of Lloyd's algorithm to initial conditions when Lloyd's algorithm is extended to an online setting.

As motivated in Section 1, here we will be concerned with developing a heuristic for online clustering which performs well on time-varying data.

The traditional k-means objective is inadequate in the non-stationary setting, and it is not obvious what it should be replaced by. In this project, we will propose a performance objective for the analog of k-means clustering in the the non-stationary setting and provide justification for the departure from the traditional k-means performance objective. We then introduce a heuristic algorithm, and in lieu of proving a performance guarantee (which it should be clear by now is an open, hard problem in the online setting, let alone when the data is non-stationary) we will test empirical performance of the algorithm on a simulated non-stationary data set. By using a simulated data set we have ground truth and can therefore calculate the optimal clairvoyant clustering of the data with respect to our performance measure. However, the findings would apply to any real data set which has similar properties of clusterability and nonstationarity; we discuss this in more detail in Section 4 and Section 5.

3 Cost objective in the non-stationary setting

The traditional batch k-means cost function is given by

$$\text{Cost}(C_1, ..., C_k, z_1, ..., z_k) = \sum_{k} \sum_{\{i: x_i \in C_k\}} ||x_i - z_k||_2^2$$

where $C_1, ..., C_k$ are the k clusters and $z_1, ..., z_k$ are the cluster centers [8].

In the online setting, we must consider the cost at any point in time. At time t, we have centers $z_1^t, ..., z_k^t$ and clusters $C_1^t, ..., C_k^t$. Then

$$\text{Cost}_t(C_1^t, ..., C_k^t, z_1^t, ..., z_k^t) = \sum_k \sum_{\{i:x_i \in C_k^t\}} ||x_i - z_k^t||_2^2$$

I claim that this objective is inappropriate in the non-stationary setting; this objective implicitly assumes that we want to find the best online clustering for all of the points seen so far, and at time t all points $x_1, ..., x_t$ contribute equally to the cost of the algorithm. However, in the non-stationary setting, old data points should not be worth as much as new data points; the data may have shifted significantly over time.

In order to take this into account, I propose the following cost function for the non-stationary setting:

$$\text{Cost}_t(C_1^t, ..., C_k^t, z_1^t, ..., z_k^t) = \sum_k \sum_{\{i:x_i \in C_k^t\}} \delta^{t-i} ||x_i - z_k^t||_2^2$$

where $\delta \in (0, 1)$. Under this cost function, we want to find k clusters which minimize the *weighted* squared distance from each point to its cluster center, where the weight is exponentially decreasing in the age of the point.

4 Algorithms

Since we are in an online setting with a theoretically infinite stream of data, it is impossible to store all of the data points seen so far. Therefore we assume that from one time period to the next, we can only keep $O(k)$ pieces of information where k is the number of cluster centers.

In Online Lloyd's algorithm, at time step t, each point $x_1, ..., x_t$ contributes equally to determine the updated centers. Here is the psuedocode for Online Lloyd's Algorithm:

```
initialize the $k$ cluster centers $z_1, ..., z_k$ in any way
create counters $n_1, ..., n_k$ and initialize them to zero
loop
   get new data point $x$
   determine the closest center $z_i$ to x
   update the number of points in that cluster:  $n_i \leftarrow n_i + 1$
   update the cluster center:  $z_i \leftarrow z_i + \frac{1}{n_i}(x - z_i)$
end loop
```

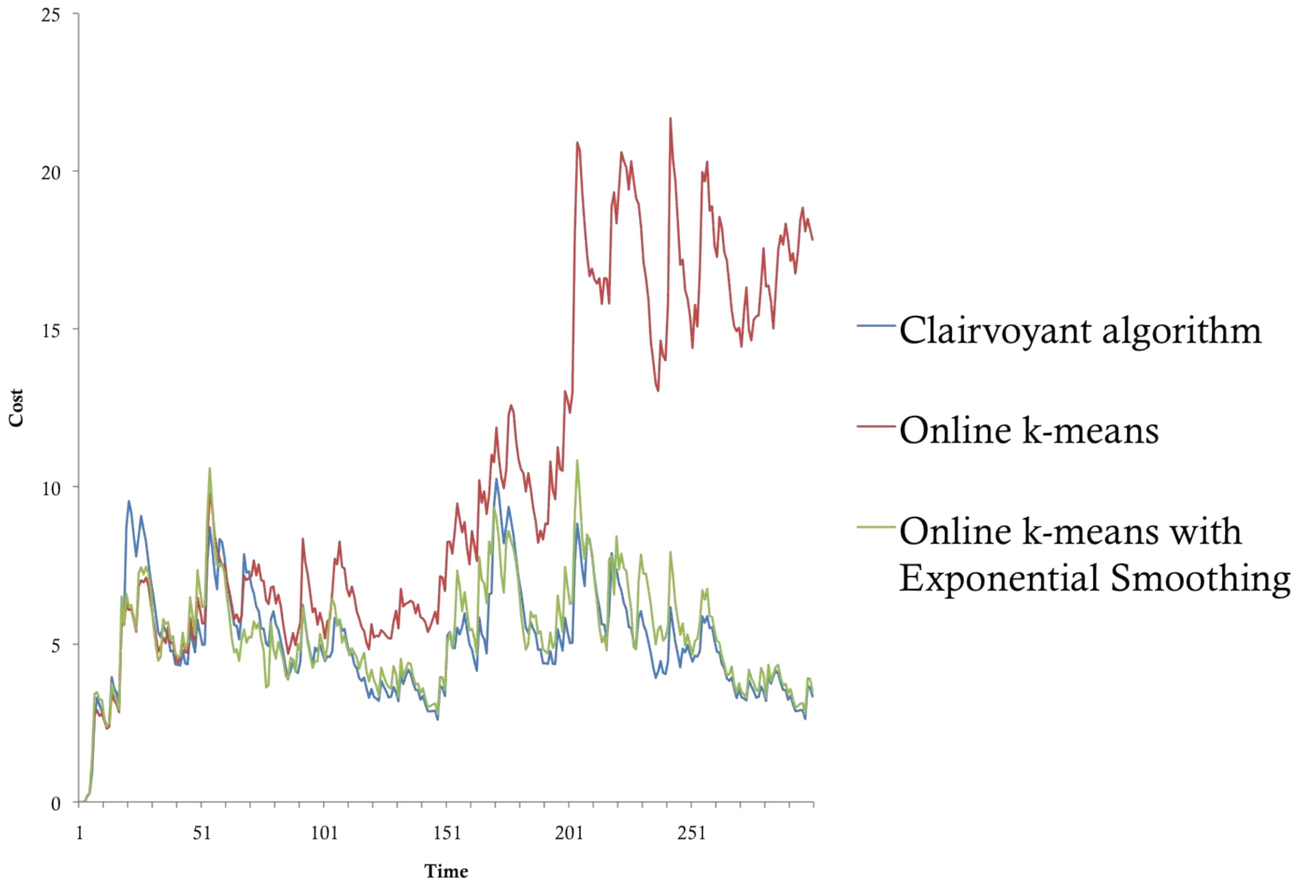
I propose an alternative to updating the centers by the update rule $z_i \leftarrow z_i + \frac{1}{n_i}(x - z_i)$. Instead, I suggest using a discounted updating rule $z_i \leftarrow z_i + \alpha(x - z_i)$ for $\alpha \in (0, 1)$.

This exponential smoothing heuristic should work well when the data is naturally clusterable and the cluster centers are moving over time. However, as with Lloyd's algorithm, it may perform arbitrarily badly and is particularly sensitive to the initial cluster centers. Also, it may not be a good choice of algorithm if you do not know the number of clusters a priori.

5 Empirical Tests

In order to run empirical tests, I simulated a data set. I wanted the data set to have the properties of having natural clusters and for these clusters to drift over time. Since I needed a data set with ground truth, I modified the well known data set, iris, in order to have the time series property. Therefore the modified iris data had four dimensions, with 3 natural clusters whose centers drift over time.

I tested Online Lloyd's algorithm on this dataset, as well as my exponentially smoothed version of Online Lloyd's. Additionally, I calculated the optimal clairvoyant cost. A plot of costs appears below for $\alpha = 0.5$:



As expected, Online Lloyd's performs poorly as time goes on, but my exponential smoothing version of Online Lloyd's forgets the initial points and performs close to the optimal clairvoyant solution.

6 "Practical Performance Guarantee"

It is a hard, open problem to make performance guarantees for online clustering, let alone when the data is non-stationary. However, if we can keep track of the cost that an algorithm is incurring over time, we can give a practical performance guarantee by comparing costs of different algorithms against each other. I claim that it is possible to keep track of the cost without holding all points ever seen in memory. Specifically, I give the following result:

Result 1. *It is possible to calculate the cost in time period $t+1$ recursively with only $O(k)$ pieces of information from the previous time period.*

Proof. We will need to keep track of the following pieces of information:

- $\text{Cost}^{\mathbf{t}} = (\text{Cost}_1^t, \text{Cost}_2^t, ...\text{C}_k^t)$ = the cost at time t for each center
- $\mathbf{z^t} = (z_1^t, z_2^t, ...z_k^t)$, the cluster centers at time t
- $\text{sum-delta}_t = (\text{sum-delta}_t^1, ..., \text{sum-delta}_t^k)$ where $\text{sum-delta}_t^k = \sum_{\{i:x_i \in C_k^t\}} \delta^{t-i}$ is a scalar keeping track of the sum of the discount factors of every point that has ever entered cluster k up to time period t. At every time period we will multiply this number by δ.
- $\text{sum-delta-x}_t = (\text{sum-delta-x}_t^1, ..., \text{sum-delta-x}_t^k)$ where $\text{sum-delta-x}_t^k = \sum_{\{i:x_i \in C_k^t\}} \delta^{t-i} x_i$ is a vector keeping track of the sum of every point that has ever entered cluster k up to time period t, weighted by its discount factor.

Let d be the dimension of the data points.

Note that $\text{Cost}_k^t = \sum_{\{i:x_i \in C_k^t\}} \delta^{t-i} ||x_i - z_k^{t+1}||_2^2 = \sum_{\{i:x_i \in C_k^t\}} \delta^{t-i} \sum_{j=1}^d (x_{i_j} - z_{k_j}^t)^2$ for every center k. Suppose that at time $t+1$ a new data point x_{t+1} arrives to cluster k.

First we update:

$\text{sum-delta}_{t+1} \leftarrow \delta \text{sum-delta}_t$
$\text{sum-delta}_{t+1}^k \leftarrow \delta \text{sum-delta}_{t+1}^k + 1$
$\text{sum-delta-x}_{t+1} \leftarrow \delta \text{sum-delta-x}_t$
$\text{sum-delta-x}_{t+1}^k \leftarrow \delta \text{sum-delta-x}_{t+1}^k + x_{t+1}$

The new center z_k^{t+1} can be chosen in any way. Then the cost in time period $t+1$ is given by $\text{Cost}_k^{t+1} + \delta \sum_{j \neq k} \text{Cost}_j^t$. We can calculate Cost_k^{t+1} as follows.

$$\begin{aligned}
\text{Cost}_k^{t+1} &= \sum_k \sum_{\{i:x_i\in C_k^{t+1}\}} \delta^{t+1-i}||x_i - z_k^{t+1}||_2^2 \\
&= \sum_{\{i:x_i\in C_k^{t+1}\}} \delta^{t+1-i}\sum_{j=1}^{d}(x_{i_j} - z_{k_j}^{t+1})^2 \\
&= \sum_{\{i:x_i\in C_k^{t+1}\}} \delta^{t+1-i}\sum_{j=1}^{d}(x_{i_j} - z_{k_j}^{t} + z_{k_j}^{t} - z_{k_j}^{t+1})^2 \\
&= \sum_{\{i:x_i\in C_k^{t+1}\}} \delta^{t+1-i}\sum_{j=1}^{d}(x_{i_j} - z_{k_j}^{t})^2 + 2(x_{i_j} - z_{k_j}^{t})(z_{k_j}^{t} - z_{k_j}^{t+1}) + (z_{k_j}^{t} - z_{k_j}^{t+1})^2 \\
&= \delta\left(\sum_{\{i:x_i\in C_k^{t}\}} \delta^{t-i}\sum_{j=1}^{d}(x_{i_j} - z_{k_j}^{t})^2\right) + \sum_{j=1}^{d}(x_{t+1_j} - z_{k_j}^{t})^2 \\
&\quad + \sum_{\{i:x_i\in C_k^{t}\}} \delta^{t+1-i}\sum_{j=1}^{d}2(x_{i_j} - z_{k_j}^{t})(z_{k_j}^{t} - z_{k_j}^{t+1}) + \sum_{\{i:x_i\in C_k^{t}\}} \delta^{t+1-i}\sum_{j=1}^{d}(z_{k_j}^{t} - z_{k_j}^{t+1})^2 \\
&= \delta\text{Cost}_k^t + \sum_{j=1}^{d}(x_{t+1_j} - z_{k_j}^{t})^2 + \text{sum-delta}_{t+1}^{k}\left(\sum_{j=1}^{d}(z_{k_j}^{t} - z_{k_j}^{t+1})^2\right) \\
&\quad + 2\sum_{j=1}^{d}(z_{k_j}^{t} - z_{k_j}^{t+1})^2\left(\text{sum-delta-x}_{t+1}^{k} - z_{k_j}^{t}\text{sum-delta}_{t+1}^{k}\right)
\end{aligned}$$

Therefore, counting a point as a single piece of information, we can calculate the cost recursively from time t to time $t+1$ by storing $2 + 2k = O(k)$ pieces of information from one period to the next.

□

Therefore, even without knowing the optimal cost, it is possible to run several online clustering algorithms in parallel, calculate the cost each one is incurring, and choose the best one for a specific dataset.

7 Conclusion

In this paper we have explored online clustering, specifically for non-stationary data. Online clustering is a hard problem and performance guarantees are rare. When non-stationarity is introduced, there is not even a standard agreed-upon framework for measuring performance. Here, I have defined a cost function for the non-stationary setting, and demonstrated a simple exponential smoothing algorithm which performs well in practice. There is lots of room to invent algorithms for online clustering of non-stationary data, but little theoretical machinery to give performance guarantees. In lieu of this, I have shown that at the very least, we can compare algorithms against each other by keeping track of the cost with only $O(k)$ pieces of information. This gives a "practical performance guarantee."

This paper scratches the surface of online clustering for non-stationary data. In general, this corner of clustering research contains many open questions.

References

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

[2] D. Arthur and S Vassilvitskii. "k-means++: the advantages of careful seeding". In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–2035, 2007.

[3] W. Barbakh and C. Fyfe. "Online Clustering Algorithms". *International Journal of Neural Systems (IJNS)*, 18(3):1–10, 2008.

[4] Anna Choromanska and Claire Monteleoni. "Online Clustering with Experts". In *Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings*, ICML 2011 Workshop on Online Trading of Exploration and Exploitation 2, 2011.

[5] Sanjoy Dasgupta. "Clustering in an Online/Streaming Setting". Technical report, Lecture notes from the course CSE 291: Topics in Unsupervised Learning, UCSD, 2008. Retrieved at cseweb.ucsd.edu/~dasgupta/291/lec6.pdf.

[6] T. Kanungo and D. Mount, N. Netanyahu, C. Piatko, R. Silverman and A. Wu. "A Local Search Approximation Algorithm for k-Means Clustering". *Computational Geometry: Theory and Applications*, 2004.

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

[8] Cynthia Rudin and Seyda Ertekin. "Clustering". Technical report, MIT 15.097 Course Notes, 2012.

[9] Shi Zhong. "Efficient Online Spherical K-means Clustering". In *Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings*, IEEE Int. Joint Conf. Neural Networks (IJCNN 2005), pages 3180–3185, Montreal, August 2005.