# Evolutionary Clustering

Deepayan Chakrabarti Ravi Kumar Andrew Tomkins
Yahoo! Research, 701 First Ave, Sunnyvale, CA 94089.
{deepay, ravikumar, atomkins}@yahoo-inc.com

## ABSTRACT

We consider the problem of clustering data over time. An *evolutionary clustering* should simultaneously optimize two potentially conflicting criteria: first, the clustering at any point in time should remain faithful to the current data as much as possible; and second, the clustering should not shift dramatically from one timestep to the next. We present a generic framework for this problem, and discuss evolutionary versions of two widely-used clustering algorithms within this framework: $k$-means and agglomerative hierarchical clustering. We extensively evaluate these algorithms on real data sets and show that our algorithms can simultaneously attain both high accuracy in capturing today's data, and high fidelity in reflecting yesterday's clustering.

**Categories and Subject Descriptors:** H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

**General Terms:** Algorithms, Experimentation, Measurements

**Keywords:** Clustering, Temporal Evolution, Agglomerative, $k$-means

## 1. INTRODUCTION

*Evolutionary clustering* is the problem of processing timestamped data to produce a sequence of clusterings; that is, a clustering for each timestep of the system. Each clustering in the sequence should be similar to the clustering at the previous timestep, and should accurately reflect the data arriving during that timestep.

The primary setting for this problem is the following. Every day, new data arrives for the day, and must be incorporated into a clustering. If the data does not deviate from historical expectations, the clustering should be "close" to that from the previous day, providing the user with a familiar view of the new data. However, if the structure of the data changes significantly, the clustering must be modified to reflect the new structure. Thus, the clustering algorithm must trade off the benefit of maintaining a consistent clustering over time with the cost of deviating from an accurate representation of the current data.

The benefits of evolutionary clustering compared to traditional clustering appear in situations in which the current (say, daily) clustering is being consumed regularly by a user or system. In such a setting, evolutionary clustering is useful for the following reasons:

(1) *Consistency:* A user will find each day's clustering familiar, and so will not be required to learn a completely new way of segmenting data. Similarly, any insights derived from a study of previous clusters are more likely to apply to future clusters.

(2) *Noise removal:* Providing a high-quality and historically consistent clustering provides greater robustness against noise by taking previous data points into effect. As we describe later, our method subsumes standard approaches to windowing and moving averages.

(3) *Smoothing:* If the true clusters shift over time, evolutionary clustering will naturally present the user with a smooth view of the transition.

(4) *Cluster correspondence:* As a side effect of our framework, it is generally possible to place today's clusters in correspondence with yesterday's clusters. Thus, even if the clustering has shifted, the user will still be situated within the historical context.

Overview of framework. Formally, let $C_t$ be the clustering produced by the algorithm for data arriving at timestep $i$. The *snapshot quality* of $C_t$ measures how well $C_t$ represents the data at timestep $t$. The *history cost* of the clustering is a measure of the distance between $C_t$ and $C_{t-1}$, the clustering used during the previous timestep. Typically, the snapshot quality is defined in terms of the data points themselves, while the history cost is a function of the cluster models. The overall cost of the clustering sequence is a combination of the snapshot quality and the history cost at each timestep.

For intuition, we consider an extreme example to show why the introduction of history cost may have a significant impact on the clustering sequence. Consider a data set in which either of two features may be used to split the data into two clusters: feature A and feature B. Each feature induces an orthogonal split of the data, and each split is equally good. However, on odd-numbered days, feature A provides a slightly better split, while on even-numbered days, feature B is better. The optimal clustering on each day will shift radically from the previous day, while a consistent clustering using either feature will perform arbitrarily close


*KDD'06,* August 20–23, 2006, Philadelphia, Pennsylvania, USA.

to optimal. In this case, a clustering algorithm that does not take history into account will produce a poor clustering sequence.

This approach may be contrasted to *incremental clustering*, in which a model is incrementally updated as new data arrive, primarily to avoid the cost of storing all historical similarities [9]. In evolutionary clustering, however, the focus is upon optimizing a new quality measure which incorporates deviation from history.

Summary of Contributions. We consider two classical clustering algorithms and extend them to the evolutionary setting: (1) the traditional $k$-means algorithm that provides a *flat* clustering of points in a vector space, and (2) a bottom-up agglomerative *hierarchical* clustering algorithm. These represent two major categories of clustering methods and we chose to use them to demonstrate the generality of our framework. These algorithms are applied to a large evolving dataset of user-tags placed on images from `flickr.com`, tracked over 68 weeks. Our experiments demonstrate all the advantages of evolutionary clustering mentioned previously, namely, stable and consistent clusterings are obtained even when the data exhibits noisy behavior, and a smooth sequence of clusters with very low history cost can be obtained for only a small reduction in total snapshot quality.

Our framework for evolutionary clustering is in fact quite general. With suitable definitions of history cost and snapshot quality, many other static clustering algorithms can now be extended to perform evolutionary clustering under our framework. Furthermore, we focused on a setting in which today's data must be clustered before tomorrow's data is available. However, there are other natural settings. For example, the entire sequence may be available to the algorithm at once, but the algorithm must again produce a sequence of clusterings that accurately reflect the data at each timestep while shifting as gently as possible over time. This formulation applies when the data should be clustered retroactively but interpretability across time is an important consideration. Similarly, there are settings in which the numerical variable over which we track evolution is not time; for example, we may cluster products by dollar value, and ask that the clusterings for similar price buckets be as similar as possible. Our formulation can capture such directions as well, but we will not explore them in this paper.

## 2. RELATED WORK

Clustering is a well-studied problem; see, for instance [16, 8]. However, to the best of our knowledge, evolutionary clustering has not been considered before.

In the following, we briefly review the interplay of time and notions related to clustering, including classification and online topic detection and tracking.

Temporal aspects have been considered in some classification problems. Some work in online machine learning considers learning tasks such as classification in which the model evolves over time, and the algorithm is penalized for both mistakes and shifts in the model from one timestep to the next [11, 3]. However, it is unclear how this could be used in the unsupervised learning setting.

Temporal aspects have also been considered in online document clustering setting. Online document clustering applies clustering sequentially to a time series of text documents, in order to detect novelty in the form of certain types of outliers [4]. Zhang et al. [18] proposed a probabilistic model for online document clustering, where the emphasis is again on detecting novelty. Clustering was also used in automatic techniques for discovering and retrieving topically related material in data streams, and to detect novel events from a temporally ordered collection of news stories [17, 2]. However, the main goal of topic detection and tracking is to detect new events in a time-line using methods such as clustering, and not to produce clusterings that incorporate history into the clustering objective.

In some data stream applications, time has played a different role with respect to clustering. Aggarwal et al. [1] study the problem of clustering data streams at different time horizons using an online statistical aggregation stage and then an offline clustering stage. For more details on clustering from a data stream analysis perspective, see [10].

The notion of clustering time-series has been considered in statistics, data mining, and machine learning. Temporal correlation is perhaps the best-known approach to time-series similarity [5]. Smyth [14] considers general clustering of sequence data using a probabilistic approach based on hidden Markov models; see also [12]. Immorlica and Chien [6] propose a low-dimensional representation of time-series for clustering. They use a variety of basis functions including piecewise-constant, piecewise-linear, and piecewise-aggregate approximations. Vlachos et al. propose a Fourier approach to this problem [15]. Our work, however, has much broader scope; we must consider object feature similarity in addition to the similarities in their time series, as will be explained in the next section.

## 3. FRAMEWORK

In this section, we present our framework for evolutionary clustering. We begin by differentiating between algorithms that must cluster data as it arrives, versus algorithms that have access to the entire time series before beginning work. An evolutionary clustering algorithm is *online* if it must provide a clustering of the data during timestep $t$ before seeing any data for timestep $t+1$. If the algorithm has access to all data beforehand, it is *offline*. Offline algorithms may "see the future," and should perform at least as well as their online counterparts. However, the online setting is arguably more important for real-world applications, and there are no natural, efficient offline algorithms that may be easily employed as lower bounds. We therefore leave the offline problem for future work, and focus on the online version.

### 3.1 Overview of the framework

Recall that an evolutionary clustering algorithm must produce a sequence of clusterings, one for each timestep. The clustering produced during a particular timestep should measure well along two distinct criteria: it should have low history cost, meaning it should be similar to the previous clustering in the sequence, and it should have high snapshot quality, meaning it should be a high-quality clustering of the data that arrived during the current timestep. The snapshot quality simply reflects the underlying figure of merit driving a particular clustering algorithm in a non-evolutionary setting. The history cost, however, must allow a comparison of clusterings, in order to determine how much the later one has shifted from the earlier. This comparison must address the issue that some data will appear for the first time in the later clustering, while some data will be seen in the earlier clus-

tering but not the later one, and so forth. There are many examples of measures for comparing clusterings — see [13] for a discussion of the complexities that arise even in the case of flat clusterings. But this comparison may be more sophisticated than simply a comparison of two partitions of the universe. The comparison may occur at the data level, for instance by comparing how similar pairs of data objects are clustered, or at the model level, for instance by comparing the best matching between two sets of centroids in the $k$-means setting. In our setting, comparisons at the model level make the most intuitive sense, and this is what we use.

First, we require a few high-level definitions. Let $\mathcal{U} = \{1, \ldots, n\}$ be the universe of objects to be clustered. Let $U_t \subseteq \mathcal{U}$ be the set of all objects present at timestep $t$, and let $U_{\leq t} = \cup_{t' \leq t} U_{t'}$ be the set of all objects present in any timestep up to and including step $t$.

An evolutionary clustering algorithm must behave as follows. At each timestep $t$, it should produce a clustering $C_t$ of $U_{\leq t}$, the objects seen by the algorithm so far. The distance from $C_t$ to $C_{t-1}$ will be evaluated with respect to $U_{\leq t-1}$ in order to determine the historical accuracy of $C_t$. Specifically, if new data arrived for the first time during timestep $t$, the clustering will not be penalized for deviating from the previous clustering with respect to the new data unless this deviation also impacts the clustering of historical data. The history cost is computed by projecting $C_t$ onto $U_{\leq t-1}$.

At the same time, the snapshot quality of $C_t$ will be evaluated with respect to $U_t$, the objects that actually appear during step $t$. These two measures, over all timesteps, will provide an overall evaluation of the entire cluster sequence.

Observe that, in order to perform well, clustering $C_t$ must include objects in $U_{\leq t} \backslash U_t$, that is, the objects that have been seen in the past but have not appeared during the current timestep. So either implicitly or explicitly, an evolutionary clustering algorithm must "carry along" information about the appropriate location of historical information.

## 3.2 Input to the framework

Recall that $\mathcal{U} = \{1, \ldots, n\}$ is the universe of objects to be clustered. At each timestep $t$ where $1 \leq t \leq T$, a new set of data arrives to be clustered. We assume that this data can be represented as an $n \times n$ matrix $M_t$ that expresses the relationship between each pair of data objects. The relationship expressed by $M_t$ can be either based on similarity or based on distance depending on the requirements of the particular underlying clustering algorithm. If the algorithm requires similarities (resp., distances), we will write $\mathrm{sim}(i, j, t)$ (resp., $\mathrm{dist}(i, j, t)$) to represent the similarity (resp., distance) between objects $i$ and $j$ at time $t$.

At each timestep $t$, an online evolutionary clustering algorithm is presented with a new matrix $M_t$, either $\mathrm{sim}(\cdot, \cdot, t)$ or $\mathrm{dist}(\cdot, \cdot, t)$, and must produce $C_t$, the clustering for time $t$, based on the new matrix and the history so far.

A user of the framework must specify a snapshot quality function $\mathrm{sq}(C_t, M_t)$ that returns the quality of the clustering $C_t$ at time $t$ with respect to $M_t$. The user must also provide a history cost function $\mathrm{hc}(C_{t-1}, C_t)$ that returns the historical cost of the clustering at time $t$. The total quality of a cluster sequence is then

$$\sum_{t=1}^{T} \mathrm{sq}(C_t, M_t) - \mathrm{cp} \cdot \sum_{t=2}^{T} \mathrm{hc}(C_{t-1}, C_t), \tag{1}$$

where the "change parameter" $\mathrm{cp} > 0$ is a user-defined parameter which trades-off between the two. As cp increases, more and more weight is placed on matching the historical clusters.

Given this framework, an evolutionary clustering algorithm takes as input $M_1, \ldots, M_t$ and produces $C_1, \ldots, C_t$. We can compute the quality of the resulting cluster sequence, and hence we may compare evolutionary clustering algorithms and speak of optimal algorithms and optimal quality.

As we stated earlier, our focus is on the online setting. Clearly a sequence of online decisions to find $C_1, \ldots, C_t$ may not be the best offline solution to (1), but nevertheless, it is a defensible alternative given no knowledge about the future data. Our algorithms therefore try to find an optimal cluster sequence by finding at each timestep $t$ a clustering $C_t$ that optimizes the incremental quality

$$\mathrm{sq}(C_t, M_t) - \mathrm{cp} \cdot \mathrm{hc}(C_{t-1}, C_t). \tag{2}$$

## 3.3 Constructing $M_t$ from raw data

In a traditional clustering setting, the data objects are all available at once to be clustered, and some measure of similarity or distance between objects may be applied. This is also true in our setting: all other timesteps may be ignored, and local information may be used to compute similarity between objects during a particular timestep. However, there is also another type of similarity which is unique to our setting — temporal similarity. If objects recur over time, the algorithm to cluster them during a particular timestep may make use of their historical occurrence patterns. This type of similarity should not be confused with similarity between time series, as in a time series clustering problem: the object in our universe is the point of the time series, rather than the series itself. Below, we describe these, and how to combine them to form the input matrix $M_t$.

Local information similarity. In some cases, the input to evolutionary clustering might already be in the form of an inter-object similarity matrix $S_t$; in other cases, we must infer it. One common scenario involves input as a graph, for example, a bipartite graph linking the data objects of interest to a set of features. Then, similarity between objects is related to the number of features they share.

Let $\mathcal{B}(t)$ represent this bipartite graph at time $t$, and let $\mathcal{R}(t) = r_{j,j'}$ encode the number of features shared by both objects $j$ and $j'$. Then, we can say

$$\mathcal{R}(t) = \mathcal{B}(t)\mathcal{B}'(t).$$

However, notice that a similarity measure based purely on $\mathcal{R}(t)$ is "memoryless," i.e., if an object fails to appear on one snapshot, all information about its similarities with other objects is lost completely. This is not desirable since we wish to capture history to some extent. Therefore, we use

$$\mathcal{R}(t) = (1 - \beta) \cdot \mathcal{B}(t)\mathcal{B}'(t) + \beta \cdot \mathcal{R}(t-1), \quad \text{for } t > 0$$

where the parameter $\beta$ is chosen to retain enough information on relative similarities of old objects with all other objects, but not swamp the data from the current snapshot. In our experiments, we set $\beta = 0.1$.

Observe that this approach incorporates an exponentially decaying moving average of data, and could easily be modified to support other windowing techniques.

Now, we can use cosine similarity between objects to generate the local similarity matrix $S_t$:

$$S_t(j, j') = \frac{\langle r_j, r_{j'} \rangle}{|r_j| \cdot |r_{j'}|}.$$

Similar steps could be followed if the underlying clustering algorithm required a distance matrix instead of a similarity matrix.

TEMPORAL SIMILARITY. This is given by the standard definition of correlation of the two time series up to and including time $t_0$:

$$\mathrm{Corr}(i, j, t_0) = \frac{\sum_{t=1}^{t_0}(x_{i,t} - \mu(i,t))(x_{j,t} - \mu(j,t))}{\sqrt{\mathrm{Var}(i,t) \cdot \mathrm{Var}(j,t)}},$$

where $x_{i,t}$ represents the number of occurrences of data object $i$ in timestep $t$, and the means and variances are defined on $x_{i,1}, \ldots, x_{i,t_0}$ and $x_{j,1}, \ldots, x_{j,t_0}$.

TOTAL SIMILARITY. We combine these two types of similarity information into the final similarity matrix $M_t$, taking the overall similarity between two objects at time $t$ to be

$$M_t(i, j) = \alpha \cdot S_t(i, j) + (1 - \alpha) \cdot \mathrm{Corr}(i, j, t),$$

where $\alpha$ controls the contribution of correlation and temporal similarities.

# 4. ALGORITHMS

We now present two instantiations of our framework. Section 4.1 describes an evolutionary version of the bottom-up agglomerative hierarchical clustering algorithm and Section 4.2 discusses an evolutionary version of the traditional $k$-means. The above two choices were motivated by the significant differences between the underlying clustering algorithms: $k$-means produces a flat rather than a hierarchical clustering, implicitly requires the data to lie in a vector space, and creates a model based on pseudo-objects that lie in the same space as the actual objects being clustered. These two very different approaches thus show the generality of our framework.

## 4.1 Agglomerative hierarchical clustering

To develop an evolutionary hierarchical clustering, we first describe a standard agglomerative clustering at a particular fixed timestep $t$. Let $M = M_t = \mathrm{sim}(\cdot, \cdot, t), U = U_{\leq t}$. First, we select the pair $i, j$ of objects that maximizes $M(i, j)$. Next, we merge these two objects, creating a new object; we also update the similarity matrix $M$ by replacing the rows and columns corresponding to objects $i$ and $j$ by their average that represents the new object. We then repeat the procedure, building a bottom-up binary tree $T$ whose leaves are the objects in $U$; the tree $C_t = T_t = T$ represents the clustering of the objects at timestep $t$.

Let the internal nodes of $T$ be labeled $m_1, \ldots, m_{|U|-1}$, and let $\mathrm{sim}_M(m_i)$ represent the similarity of objects that were merged to produce the internal node $m_i$. Let $\mathrm{in}(T)$ be the set of all internal nodes of $T$. For an internal node $m$, let $m_\ell$ be the left child of $m$, $m_r$ be the right child of $m$, and $\mathrm{leaf}(m)$ be the set of leaves in the subtree rooted at $m$. Let $d(i, j)$ be the tree distance in $T$ between nodes $i$ and $j$. If $T', T$ are binary trees with $\mathrm{leaf}(T) \supseteq \mathrm{leaf}(T')$, then the tree $T'|T$ is the projection of $T'$ onto $T$, i.e., $T'|T$ is a binary tree obtained by first removing all leaves in $\mathrm{leaf}(T) \setminus \mathrm{leaf}(T')$ and then collapsing all unary internal nodes.

We define the snapshot quality of $T$ to be the sum of the qualities of all merges performed to create $T$:

$$\mathrm{sq}(T, M) = \sum_{m \in \mathrm{in}(T)} \mathrm{sim}_M(m).$$

We now define the history cost, which is the distance between two trees $T, T'$ with $\mathrm{leaf}(T) \supseteq \mathrm{leaf}(T')$. First, we define the distance between objects $i, j \in \mathrm{leaf}(T')$ to be the squared-error distance:

$$d_{T',T}(i, j) = \left(d_{T'}(i, j) - d_{T'|T}(i, j)\right)^2 \tag{3}$$

Then, the distance between $T$ and $T'$ is defined to be the average distance between all pairs of objects

$$\mathrm{hc}(T', T) = \mathop{\mathrm{E}}_{\substack{i,j \in \mathrm{leaf}(T') \\ i \neq j}} (d_{T',T}(i, j)). \tag{4}$$

As stated earlier, the goal is to find a clustering $C_t$ that minimizes (2). To do this, we first note that (4) can be rewritten as a sum of contributions from each internal node, where the contribution covers all pairs of points for whom that internal node is the least common ancestor. Thus,

$$\mathrm{hc}(T', T) = \mathop{\mathrm{E}}_{m \in \mathrm{in}(T')} \left( \mathop{\mathrm{E}}_{\substack{i \in \mathrm{leaf}(m_\ell) \\ j \in \mathrm{leaf}(m_r)}} (d_{T',T}(i, j)) \right).$$

Using this reformulation of history cost, we may write the incremental quality in (2) as

$$\sum_{m \in \mathrm{in}(T)} \mathrm{sim}_M(m) - \left( \mathrm{cp} \cdot \mathop{\mathrm{E}}_{\substack{m \in \mathrm{in}(T) \\ i \in \mathrm{leaf}(m_\ell), j \in \mathrm{leaf}(m_r)}} (d_{T',T}(i, j)) \right).$$

We propose four greedy heuristics to choose the order of merges. Let $T = T_t$ and $T' = T_{t-1}$.

In the first heuristic, we choose the merge $m$ whose contribution to this expression is maximal. In other words, pick the merge $m$ that maximizes

$$\mathrm{sim}_M(m) - \left( \mathrm{cp} \cdot \mathop{\mathrm{E}}_{\substack{i \in \mathrm{leaf}(m_\ell) \\ j \in \mathrm{leaf}(m_r)}} (d_{T',T}(i, j)) \right). \tag{5}$$

We refer to this heuristic as *Squared*, since it greedily minimizes the squared error in Equation 3.

However, we observe that a merge with a particular squared error may become better or worse if it is put off until later. To wit, if two objects are far away in $T'$, then perhaps we should *delay* the merge until they are similarly far away in $T$. However, if two objects are close in $T'$ but merging them would already make them far in $T$ then we should *encourage* the merge despite their high cost, as delaying will only make things worse. Based on this observation, we consider the cost of merge based on what would change if we delayed the merge until the two merged subtrees became more distant from one another (due to intermediate merges).

Thus, consider a possible merge of subtrees $S_1$ and $S_2$. Performing the merge incurs a penalty for nodes that are still too close, and a benefit for nodes that are already too far apart. The benefit and penalty are expressed in terms of the change in cost if either $S_1$ or $S_2$ participates in another merge, and hence the elements of $S_1$ and $S_2$ increase their average distance by 1. This penalty may be

written by taking the partial derivative of the squared cost with respect to the distance of an element to the root. At any point in the execution of the algorithm at time $t$, let $\text{root}(i)$ be the root of the current subtree containing $i$. For $i \in S_1$ and $j \in S_2$, let $d_T^m(i,j)$ be the *merge distance* of $i$ and $j$ at time $t$, i.e., $d_T^m(i,j)$ is the distance between $i$ and $j$ at time $t$ if $S_1$ and $S_2$ are merged together. Then, $d_T^m(i,j) = d_T(i,\text{root}(i)) + d_T(j,\text{root}(j)) + 2$ The benefit of merging now is given by:

$$\text{sim}_M(m) - \left( \text{cp} \cdot \mathop{\text{E}}_{\substack{i \in \text{leaf}(m_\ell) \\ j \in \text{leaf}(m_r)}} (d_{T'}(i,j) - d_T^m(i,j)) \right). \quad (6)$$

We refer to this heuristic as *Linear-Internal*. Notice that, as desired, the benefit is positive when the distance in $T'$ is large, and negative otherwise. Similarly, the magnitude of the penalty depends on the derivative of the squared error (Equation 3).

As another heuristic, we may also observe that our decision about merging $S_1$ with $S_2$ may also depend on objects that do not belong to either subtree. Assume that elements of $S_1$ are already too far apart from some subtree $S_3$. Then merging $S_1$ with $S_2$ may introduce additional costs downstream that are not apparent without looking outside the potential merge set. In order to address this problem, we modify (6) to penalize a merge if it increases the distance gap (i.e., the distance at time $t$ versus the distance at time $t-1$) between elements that participate in the merge and elements that do not. Similarly, we give a benefit to a merge if it decreases the distance gap between elements in the merge and elements not in the merge. The joint formulation is then as follows:

$$\begin{aligned} \text{sim}_M(m) \quad &- \quad \text{cp} \cdot \mathop{\text{E}}_{\substack{i \in \text{leaf}(m_\ell) \\ j \in \text{leaf}(m_r)}} (d_{T'}(i,j) - d_T^m(i,j)) \\ &+ \quad \text{cp} \cdot \mathop{\text{E}}_{\substack{i \in \text{leaf}(m) \\ j \notin \text{leaf}(m)}} (d_{T'}(i,j) - d_T^m(i,j)). \quad (7) \end{aligned}$$

This heuristic considers the *internal* cost of merging elements $i \in S_1$ and $j \in S_2$, and the *external* cost of merging elements $i \in S_1 \cup S_2$ and $j \notin S_1 \cup S_2$; therefore, we refer to it as *Linear-Both*. For completeness, we also consider the external cost alone:

$$\text{sim}_M(m) + \text{cp} \cdot \mathop{\text{E}}_{\substack{i \in \text{leaf}(m) \\ j \notin \text{leaf}(m)}} (d_{T'}(i,j) - d_T^m(i,j)). \quad (8)$$

We refer to this final heuristic as *Linear-External*.

## 4.2 $k$-means clustering

Let the objects to be clustered be normalized to unit vectors in the Euclidean space, i.e., the objects at time $t$ are given by $U_t = \{x_{1,t}, \ldots\}$ where each $x_{i,t} \in \Re^\ell$ and the distance matrix $M_t(i,j) = \text{dist}(i,j,t) = ||x_{i,t} - x_{j,t}||$. (See, for instance, [7].)

We begin with a description of the traditional $k$-means algorithm. Let $t$ be a fixed timestep and let $U = U_{\leq t}, x_i = x_{i,t}, M = M_t$. The algorithm begins with a set of $k$ cluster centroids, $c_1, \ldots, c_k$, with $c_i \in \Re^\ell$; these centroids can be initialized either randomly, or by using the results of the previous clustering $C_{t-1}$ (which is exactly "incremental $k$-means"). Let $\text{closest}(j)$ be the set of all points that are closest to centroid $c_j$, i.e.,

$$\text{closest}(j) = \{x \in U \mid j = \arg\min_{j'=1,\ldots,k} ||c_{j'} - x||\}.$$

The algorithm proceeds during several passes, during each of which it updates each centroid based on the data elements currently assigned to that centroid:

$$c_j \leftarrow \mathop{\text{E}}_{x \in \text{closest}(j)} |\,(x),$$

after which $c_j$ is normalized to have unit length. The algorithm terminates after sufficiently many passes and the clustering $C_t = C$ is given by the set $\{c_1, \ldots, c_k\}$ of $k$ centroids.

We define the snapshot quality of a $k$-means clustering to be

$$\text{sq}(C,M) = \sum_{x \in U} (1 - \min_{c \in C} ||c - x||).$$

(Since all points are on the unit sphere, distances are bounded above by 1.)

We define the history cost, i.e., the distance between two clusterings, to be

$$\text{hc}(C,C') = \min_{f:[k]\to[k]} ||c_i - c'_{f(i)}||,$$

where $f$ is a function that maps centroids of $C$ to centroids of $C'$. That is, the distance between two clusterings is computed by matching each centroid in $C$ to a centroid in $C'$ in the best possible way, and then adding the distances for these matches.

As stated earlier, we use a greedy approximation algorithm to choose the next cluster in the sequence. However, in the case of $k$-means, the greedy algorithm becomes particularly easy. At time $t$, for a current centroid $c_j^t$, let $c_{f(j)}^{t-1} \in C_{t-1}$ be the closest centroid in $C_{t-1}$. Let $n_j^t = |\text{closest}(j)|$ be the number of points belonging to cluster $j$ at time $t$; let $n_{f(j)}^{t-1}$ be the corresponding number for $c_{f(j)}^{t-1}$. Let $\gamma = n_j^t / \left(n_j^t + n_{f(j)}^{t-1}\right)$. Then, update $c_j^t$ as

$$\begin{aligned} c_j^t \leftarrow \quad & (1-\gamma) \cdot \text{cp} \quad c_{f(j)}^{t-1} \\ & +\gamma \cdot (1 - \text{cp}) \quad \mathop{\text{E}}_{x \in \text{closest}(j)} (x). \end{aligned}$$

In words, the new centroid $c_j^t$ lies in between the centroid suggested by non-evolutionary $k$-means and its closest match from the previous timestep, weighted by the cp and the relative sizes of these two clusters. Again, this is normalized to unit length, and we continue with the usual $k$-means iterations.

# 5. EXPERIMENTS

In this Section, we perform an extensive study of our algorithms under different parameter settings. We show how distance from history can be reduced significantly while still maintaining very high snapshot quality. For our experiments, we use the collection of timestamped photo–tag pairs from `flickr.com` indicating that at a given time, a certain tag was placed on a photo. A bipartite tag-photo graph is formed for each week, and two tags are considered to be similar if they co-occur on the same photo at the same timestep, as described before in Section 3. Our goal is to apply evolutionary clustering algorithms to this space of tags.

*$k$-means clustering over time.* For this experiment, we selected the most commonly occurring 5000 tags that in the Flickr data and proceeded to study their clustering. We ran $k$-means with $k = 10$ centroids over time $t = 0 \ldots 67$, for

several values of cp. Recall that cp = 0 is exactly the same as applying $k$-means independently to each snapshot, but with the clusters found in the previous step as the starting seed; it is "incremental $k$-means," in other words.

Figure 1 shows the results. We observe the following: Both the snapshot quality and the distance from history decrease as cp increases. In fact, incremental $k$-means (cp = 0) gives the best snapshot quality and worst distance from history. This is to be expected since clustering each snapshot independently should give the best quality performance, but at the cost of high distance from history. Also, even low values of cp lower the distance from history significantly. For example, even when cp is as low as 0.125, $k$-means incorporates history very well, which results in a significant drop in distance from history.

AGGLOMERATIVE CLUSTERING OVER TIME. We empirically find that Linear-Both and Linear-Internal significantly outperform both Linear-External and Squared, so in Figure 2, we plot only the performance of Linear-Both and Linear-Internal over the top 2000 tags. The plots for Linear-Both are smoother than those for Linear-Internal, for all values of the change parameter cp. This demonstrates that the extra processing for Linear-Both improves the cluster tracking ability of the algorithm. Also note that the distance from history plot shows very high values for a few timesteps. We suspect this is due to increased activity during that timeframe; that was when Flickr "took off." Note that this peak also appears during $k$-means clustering (Figure 1(b)), reinforcing the idea that this is an artifact of the data.

EFFECT OF CP ON SNAPSHOT QUALITY. Figure 3(a,b) shows the dependence of snapshot quality on cp. The snapshot quality values at time $t$ are normalized by the corresponding value for cp = 0 to remove the effects of any artifacts in the data itself. We observe that the snapshot quality is inversely related to cp. I.e., higher the cp, more the weight assigned to the distance from history, and thus worse the performance on snapshot quality.

However, while the snapshot quality decreases linearly and is well-behaved as a function of cp for $k$-means, the situation is different for agglomerative clustering. The snapshot quality takes a hit as soon as history is incorporated even a little bit, but the degradation after that is gentler. This suggests that $k$-means can accommodate more of history without compromising the snapshot quality.

EFFECT OF CP ON DISTANCE FROM HISTORY. Figure 3(c,d) shows the dependence of distance from history on the change parameter cp. The y-axis values are normalized by the corresponding value for cp = 0 at that timestep to remove any data artifacts. We see that the distance from history is inversely related with cp. I.e., as the value of cp is increased, our algorithms weigh the distance higher, and reducing the distance from history becomes relatively more important than increasing snapshot quality. Thus, higher cp leads to lower distance from history.

While $k$-means gets closer to history for small values of cp, the situation is more dramatic with agglomerative clustering. Even values of cp as small as 0.05 reduce the distance from history in a dramatic fashion. This suggests that the agglomerative clustering algorithm is easily influenced by history.

# 6. CONCLUSIONS

We considered the problem of clustering data over time and proposed an evolutionary clustering framework. This framework requires that the clustering at any point in time should be of high quality while ensuring that the clustering does not change dramatically from one timestep to the next. We presented two instantiations of this framework: $k$-means and agglomerative hierarchical clustering. Our experiments on Flickr tags showed that these algorithms have the desired properties — obtaining a solution that balances both the current and historical behavior of data.

It will be interesting to study this framework for a larger family of clustering algorithms. It will also be interesting to investigate tree-based clustering algorithms that construct non-binary and weighted trees.

# 7. REFERENCES


[1] C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the International Conference on Very Large Data Bases*, pages 852–863, 2003.

[2] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. Proc. of the DARPA Broadcast News Transcription and Understanding Workshop, 1998.

[3] P. Auer and M. Warmuth. Tracking the best disjunction. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 312–321, 1995.

[4] D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, 2004.

[5] C. Chatfield. *The Analysis of Time Series*. Chapman and Hall, 1984.

[6] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the International Conference on the World-Wide Web*, pages 2–11, 2005.

[7] I. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001.

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[9] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[10] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.

[11] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[12] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *Proceedings of the International Conference on Extending Database Technology*, pages 106–122, 2004.

[13] M. Meila. Comparing clusterings by the variation of information. In *Proceedings of the ACM Conference on Computational Learning Theory*, pages 173–187, 2003.

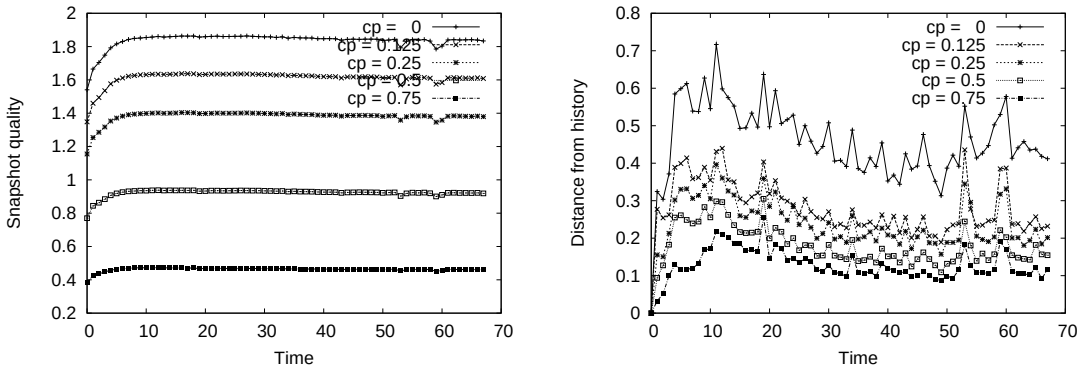
[14] P. Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, volume 9, page 648, 1997.

[15] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 131–142, 2004.

[16] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

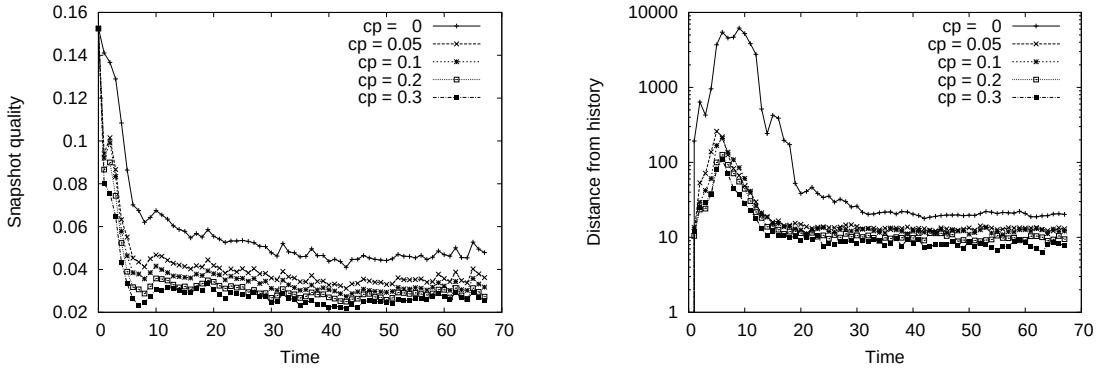
[17] Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and on-line event detection. In *Proceedings of the 21st ACM International Conference on Research and Development in Information Retrieval*, pages 28–36, 1998.

[18] J. Zhang, Z. Ghahramani, and Y. Yang. A probabilistic model for online document clustering with applications to novelty detection. In *Proceedings of Advances in Neural Information Processing Systems*, 2005.

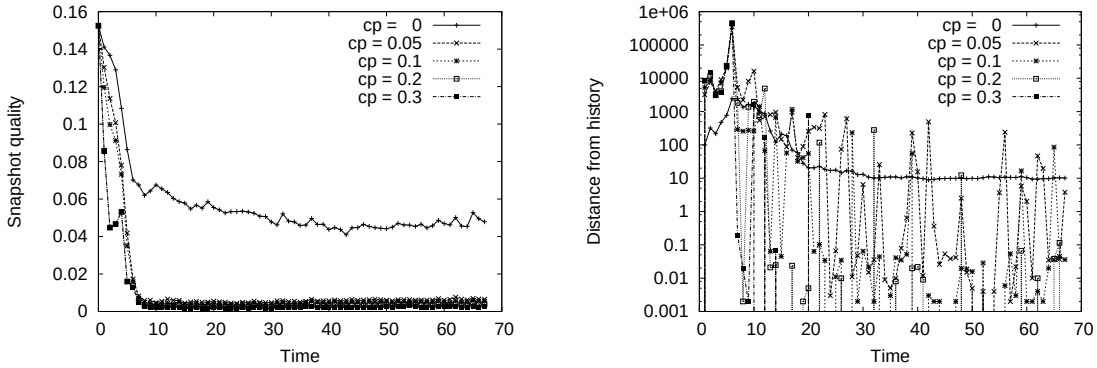



(a) Snapshot quality over time (b) Distance from history, over time

**Figure 1:** ***k-means clusters over time:*** **As the change parameter cp increases, both the snapshot quality and the distance from history decrease. The case of cp $= 0$ is "incremental $k$-means."**

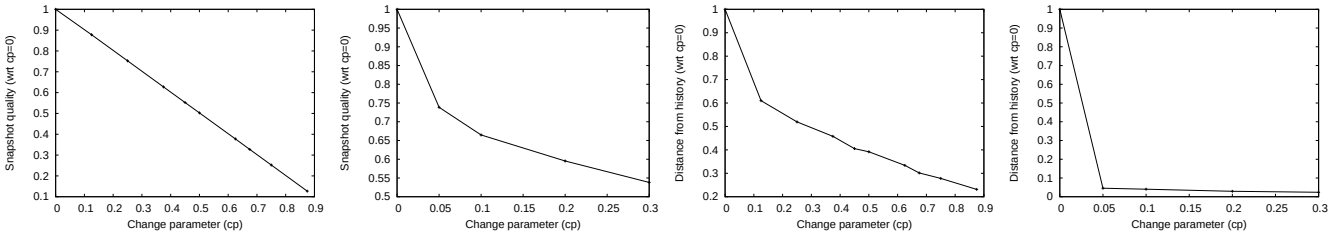



(a) Linear-Both snapshot quality (log-linear) (b) Linear-Both distance from history (log-linear)




(a) Linear-Internal snapshot quality (log-linear) (b) Linear-Internal distance from history (log-linear)

**Figure 2:** ***Performance of agglomerative clustering over time:*** **The plots for Linear-Both are far smoother than those of Linear-Internal.**




(a) $k$-means snapshot quality vs. cp (b) Agglomerative snapshot quality vs. cp (c) $k$-means distance from history vs. cp (d) Agglomerative distance from history vs. cp

**Figure 3:** ***Snapshot quality and distance from history, versus the change parameter cp.***