MULTI-TARGET TRACKING APPLIED TO EVOLUTIONARY CLUSTERING

Maria Rosario Mestre[⋆] *William J. Fitzgerald*[⋆ †]
[⋆] University of Cambridge, Signal Processing Laboratory, Cambridge, UK
[†] Featurespace, Broers Building, Hauser Forum, Cambridge, UK
[⋆] {mrm46, wjf1000}@cam.ac.uk, [†] William.Fitzgerald@featurespace.co.uk

ABSTRACT

We extend an established and robust method from multi-target tracking showing how it can be used for evolutionary clustering. Our framework models the real-life dynamics of consumer web data: the number of objects grows with time, and not all objects update their state synchronously. Our proposed algorithm tackles this problem by estimating the clusters sequentially using methods of multi-target tracking. We compare this novel technique to clustering algorithms commonly used in the literature and show how our method outperforms the other methods in terms of accuracy, stability and speed of adaptation to group dynamics. Our algorithm successfully detects changepoints in the number of clusters.

Index Terms— Evolutionary clustering, consumer web data, multi-target tracking, changepoint detection

1. INTRODUCTION

The large availability and storage of web data means that pattern extraction algorithms are becoming increasingly attractive. In an evolutionary framework, data objects are observed at multiple time points. This type of data is also referred to as longitudinal, and is very common in e-commerce websites [1, 2]. Our goal is to find a sequence of partitions over time. The term 'evolutionary' refers to the fact that data objects follow dynamic trends and we want to estimate these trends as data objects arrive, i.e. online. By grouping similar objects together, we can identify the most significant group dynamics. This in turn has many useful applications in consumer web data, mobile communication networks, finance or biomedical signal processing. For example, clustering objects can be used to segment a specific market or identify a group of fraudulent transactions from a website. In this paper we use an example from mobile communications.

We assume that we observe a sequence of snapshots or samples of a system with a growing number of data objects. Every time we take a snapshot of the system, we want to cluster all objects seen so far. We assume that objects have different run lengths, which is defined as the period of time since the last state update. This type of setting is very common in real life applications, where a customer of an e-commerce website might not be actively using their profile being in a 'dormant' state. Our purpose is to incorporate the temporal information of the objects so that the more active objects have a higher contribution to the clustering solution. The desirable properties of a clustering algorithm are: accuracy, stability and fast adaptation to new trends.

In order to follow the trends accurately, we need to allow for the number of clusters and data objects to change in time. To the best knowledge of the authors, there has been no work that has attempted to fulfil all these requirements in a unified framework. In the works by [3, 4], it is not stated how to handle a variable number of clusters and data points. In [5, 6], these situations are treated as special cases of the algorithm. Furthermore, we also work on a different data setting. In most work done in evolutionary clustering, the data arrives in sequential batches [3, 4, 5]. In our case we also have sequential batches of data, but not all objects have the same run length within a batch, and this needs to be accounted for.

We propose to use established methods from the multi-target tracking literature for clustering longitudinal data. We combine K-means and recursive filtering of the centres of the clusters, which is the same as tracking multiple objects with a nearest-neighbour data association step [7]. The main difference to any previous work done in multi-target tracking is in the application. In a typical multi-target tracking problem, the targets (in our case, the centroids of the clusters) move over time in an enviroment with clutter, and the goal is to estimate their position. We frame our problem differently in that our targets move but our principal objective is to find the association of objects to different targets. Our focus is in the hard labelling of the objects, which makes this a clustering problem. The work in [7] is very similar to the algorithm presented here but differs in the application. From the point of view of the implementation, the number of objects being tracked is fixed (i.e. clusters), and it does not assign all objects to a cluster.

There are two established variations of the sequential K-means that can be found in the literature and that are applicable to our specific framework [8]. In these algorithms, the cluster centroids are estimated sequentially. We will explain them in more detail in the next section. In addition to these sequential algorithms, we will compare our algorithm to the incremental K-means which performs the clustering by batches of data instead of sequentially [4]. In the rest of the paper, we explain our algorithm and draw a comparison with other clustering algorithms using synthetic data. In order to test how well a clustering algorithm fulfils the desired properties previously mentioned, we develop several performance metrics. Finally, we also show results on a real dataset from mobile communications.

2. STATE-SPACE MODEL

We observe snapshots $Y_1, Y_2, \dots, Y_T$ of a system with growing number of objects, i.e. $Y_t = \{\mathbf{y}_k\} = \{\mathbf{y}_1, \dots, \mathbf{y}_{N_t}\}$. We use $t = \{1, \dots, T\}$ to index snapshot sampling times, and $k = \{1, \dots, N_t\}$ to index objects in each snapshot. The number of clusters C_t can vary between time steps. We assume the number of clusters to be c for now. At every time step, all objects observed so far are sampled using their last state. Then, the objects are ordered by decreasing run length, so that $\mathbf{y}_k$ is more recent than $\mathbf{y}_{k-1}$. Even though run lengths do not align with time steps, by a slight abuse of language we will refer to each object $\mathbf{y}_k$ as having been received at different times. Note that our dataset is accumulative and carries forward the inactive objects. This choice has been done so that information contained in historical data is still used, as in most evolutionary clustering algorithms. We cluster the data objects in each snapshot sequentially.

The sequential clustering problem can be formulated as a state-space problem,

$$\mathbf{y}_k = H_k \mathbf{x}_k + \boldsymbol{\epsilon}_k \tag{1}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \boldsymbol{\theta}_k \tag{2}$$

where $\mathbf{y}_k$ is the $p \times 1$ observation vector, $\mathbf{x}_k$ is the $pc \times 1$ unobserved state vector, H_k is the $p \times pc$ observation model matrix and $\boldsymbol{\epsilon}_k$, $\boldsymbol{\theta}_k$ are zero-mean gaussian noise vectors with covariance matrices R_k and Q_k respectively. In our framework, the state vector is built by stacking all centroids such that $\mathbf{x}_k = \mathrm{vec}(\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,c})$, where $\mathbf{x}_{k,c}$ is the c-th centroid at time k with dimensions $p \times 1$. We construct the observation model with the constraint that the observation $\mathbf{y}_k$ can only belong to a single cluster. The observation model matrix H_k is thus $H_k = \mathbf{h}_k \otimes I_{pp}$ where I_{pp} is the $p \times p$ identity matrix and $\mathbf{h}_k$ is a $1 \times c$ vector with 1 at position $i(\mathbf{y}_k)$ and zero elsewhere. The integer $i(\mathbf{y}_k)$ is the index of the cluster to which $\mathbf{y}_k$ belongs. The symbol $\otimes$ is the Kronecker product.

We use the well-known Kalman filter to get an estimate $\hat{\mathbf{x}}_k$ of the state vector [9]. Our model is nonlinear as the observation model depends on the observation, the Kalman filter solution is therefore an approximation. The covariance matrix Q_k can be defined as $Q_k = \mathrm{cov}(\mathbf{x}_k) = \sigma_1^2 I_{pp} \otimes I_{cc}$. For simplicity, we assume that centroids and coordinates are independent from each other.

For the covariance matrix R_k, we tried two different models where observations are independent from each other. In the first model all observations have the same variance $R_k = \sigma_2^2 I_{pp}$. In the second model, we used a varying estimate for R_k such that $R_k = \sigma_2^2 I_{pp} ||\mathbf{y}_k - \hat{\mathbf{x}}_{k-1,i(\mathbf{y}_k)}||$. The intuition behind this is that if an observation is far from its assigned centroid, the confidence in the estimate is low. We found the second model yielded better results, and thus the results reported here are for this covariance matrix. We refer to the final state estimate in a certain snapshot $\mathbf{x}_{N_t}$ as $\mathbf{x}_t$ for convenience. We name our algorithm 'KC' in reference to 'Kalman clustering'. It has been implemented as a function that takes parameters KC($Y_t, \mathbf{x}_{t-1}, C_t$), where $\mathbf{x}_{t-1}$ is the state estimate at time $t-1$. For $t = 1$, we use an initial estimate $\mathbf{x}_0$ which is the mean over Y_1.

As mentioned previously, we would like to compare our approach to two better-known sequential K-means algorithms (3) and (4) found in [8]. For each time k, we update the centroids

$$\hat{\mathbf{x}}_{k,i(\mathbf{y}_k)} = \hat{\mathbf{x}}_{k-1,i(\mathbf{y}_k)} + \frac{(\mathbf{y}_k - H_k \hat{\mathbf{x}}_{k-1})}{\sum_{k'=1}^{k'=k} \mathbb{1}_{i(\mathbf{y}_k)}(i(\mathbf{y}_{k'}))} \tag{3}$$

$$\hat{\mathbf{x}}_{k,i(\mathbf{y}_k)} = \hat{\mathbf{x}}_{k-1,i(\mathbf{y}_k)} + \gamma(\mathbf{y}_k - H_k \hat{\mathbf{x}}_{k-1}) \tag{4}$$

where γ is a constant between 0 and 1 and $\mathbb{1}(x)$ is the indicator function such that $\mathbb{1}_a(x) = 1$ if $x = a$ and 0 otherwise. We will refer to clustering using update (3) as 'sequential' and clustering using update (4) as 'exponential'.

For the 'KC', 'sequential', 'exponential' and 'incremental' K-means algorithms, we compute the index of the cluster assigned to each object by minimising the Euclidian distance, like a nearest-neighbour association, $i(\mathbf{y}_k) = \mathrm{argmin}_c\, ||\mathbf{y}_k - \hat{\mathbf{x}}_{k-1,c}||$. The only difference with the 'incremental' K-means is that it assigns all objects and updates all centroids in a batch instead of sequentially.

3. THE ALGORITHM

The estimation of the number of clusters is done sequentially between snapshots. This part of the algorithm is common to all clustering solutions mentioned above. We assume that the number of clusters C_t does not change too abruptly, and we therefore assume that $(C_t - C_{t-1}) \in \{-1, 0, 1\}$ for every time step. Our approach is very similar to a split-merge rule, in which we recompute the clusterings for $(C_{t-1} + 1)$ and $(C_{t-1} - 1)$ clusters, assign costs to each option, and keep the clustering that yields the lowest cost. In order to make it less computationally intensive, we only look for local variations of the clustering, where a specific cluster might split or two clusters might merge. In order to do this, we first pick the clusters that seem the best candidates to split or merge. The best candidate to split is the cluster c_s that has the highest variance. Similarly, we pick the two clusters c_{m1} and c_{m2} that are the best candidates to merge when the distance between their centroids is the minimum value. The 'split' and 'merge' solutions $\mathbf{x}_s$ and $\mathbf{x}_m$ are computed by running the Kalman filter on the objects assigned to c_s with 2 clusters and c_{m1}, c_{m2} with 1 cluster respectively, leaving the other clusters unchanged.

The centroids for the local split-merge clusters are initialised by taking observations at random. We refer to these initial estimates as $\mathbf{x}_{0,s}$ and $\mathbf{x}_{0,m}$.

We then compute the cost for each clustering by using the Aikaike Information Criterion (AIC), which is commonly used to determine the number of clusters [10]. For a given clustering $\mathcal{C}_t$, the associated cost would be:

$$f_{aic}(\mathcal{C}_t) = \frac{1}{N_t} \sum_{c=1}^{C_t} \sum_{\substack{\mathbf{y}_k \\ i(\mathbf{y}_k)=c}} \|\mathbf{y}_k - \mathbf{x}_{t,c}\|^2 + \lambda C_t, \quad (5)$$

where λ is used as a penalisation for a high number of clusters and the first term is the goodness of fit of the clustering. We also need to divide by the number of objects to be able to account for the growing number of objects in time. We show the overview of the 'KC' algorithm in Algorithm 1. The only difference with the 'sequential', 'exponential' and 'incremental' versions is that they use a different function instead of KC.

Algorithm 1 Our Kalman clustering

```
 1: initial estimate for $\mathbf{x}_0$
 2: $C_0 \leftarrow 1$
 3: for all snapshots $Y_t$ where $t \in \{1, \ldots, T\}$ do
 4:     current solution: $\mathbf{x}_t \leftarrow \mathrm{KC}(Y_t, \mathbf{x}_{t-1}, C_{t-1})$
 5:     find $c_s$, $c_{m1}$ and $c_{m2}$
 6:     $Y_s \leftarrow \{\mathbf{y}_k : i(\mathbf{y}_k) = c_s\}$
 7:     $\{\mathbf{x}_{s1}, \mathbf{x}_{s2}\} \leftarrow \mathrm{KC}(Y_s, \mathbf{x}_{0,s}, 2)$
 8:     split solution: $\mathbf{x}_s \leftarrow \{\mathbf{x}_t \backslash \mathbf{x}_{t,c_s}, \mathbf{x}_{s1}, \mathbf{x}_{s2}\}$
 9:     $Y_m \leftarrow \{\mathbf{y}_k, \mathbf{y}_{k'} : i(\mathbf{y}_k) = c_{m1}, i(\mathbf{y}_{k'}) = c_{m2}\}$
10:     $\mathbf{x}_m \leftarrow \mathrm{KC}(Y_m, \mathbf{x}_{0,m}, 1)$
11:     merge solution: $\mathbf{x}_m \leftarrow \{\mathbf{x}_t \backslash \{\mathbf{x}_{t,c_{m1}}, \mathbf{x}_{t,c_{m2}}\}, \mathbf{x}_m\}$
12:     $\{\mathbf{x}_t, C_t\} \leftarrow \mathrm{argmin}_{\mathcal{C} \in \{\mathbf{x}_t, \mathbf{x}_s, \mathbf{x}_m\}} f_{aic}(\mathcal{C})$
13: end for
```

4. RESULTS

4.1. Synthetic data

We generated synthetic data with different dynamics to be tested. The objects originated from varying number of clusters modelled as a mixture of t-distributions. We used t-distributions to emulate most real web datasets. Compared to [4], we had more structural changes. We created a timeline with 3 event times. The first and second events at t_1 and t_2 mark the period of activity of cluster 2, after which it becomes inactive. There is a changepoint whenever the number of clusters changes. Event t_2 is either the beginning of a split or a merge, and thus a changepoint. The timeline in Figure 1 shows an example of a split scenario. We generated 25 scenarios with random splits and 25 scenarios with random merges. Time t_3, as well as the positions of the 'splitting' and 'merging' clusters were random. We worked on $T = 1000$ with an incremental rate of 10 new objects per time step, so that at time T we have 10,000 objects. We used two different 'ground truth' labellings: one that 'forgets' about inactive clusters and another one that does not. We tried both for each algorithm and reported the best results. This is only needed to compute the adjusted Rand index mentioned later, the other performance metrics do not rely on the true labelling.

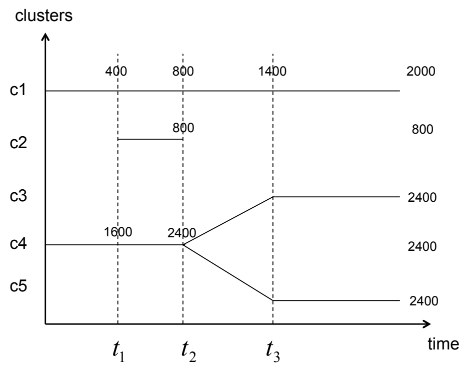

Fig. 1: Timeline of the synthetic data showing a split of c_4 into c_3 and c_5. The accumulated number of objects in each cluster is shown.

We compared the 'sequential' (3), 'exponential' (4) and 'incremental' K-means [4] to our solution in terms of accuracy, stability and speed of adaptation. Similarly to [4], we used the adjusted Rand index which is a well-known measure for clustering accuracy [11]. For stability, we based our metric on the oscillations n_o in the number of clusters, $s = 1 - \frac{n_o}{1000}$. This metric gives the proportion of time when the number of clusters was stable. Finally, we used a rule-based automated check to determine whether the changepoint at t_2 was successfully identified and to measure the reaction time, i.e. how long it took to adapt to the changepoint. We do not describe the series of rules used because of space constraints.

We set $\lambda = 20$ across all methods for comparison purposes. The parameter γ in the 'exponential' algorithm affects how fast the clustering responds to different dynamics. However there is a compromise between stability, accuracy and the speed of adaptation to changes. A low γ yields a higher reaction time but fewer oscillations in the number of clusters over time. If the oscillations are too high, this makes the clustering result difficult to interpret. We found an oscillation threshold of 15% of the time to be acceptable. Therefore, we decided to set $\gamma = 0.67$, which yielded a very high adjusted Rand index and more than 85% stability. The covariance parameters σ_1 and σ_2 were selected so as to optimise the adjusted Rand index. The best performance was consistently obtained for $[\sigma_1 = 1, \sigma_2 = 3]$. An illustration of a scenario with a split can be seen in Figure 2. The clustering was able to successfully identify the split between two groups of objects, since they were assigned to different clusters.

We summarise the results in Table 1. Our algorithm 'KC' outperformed the others in terms of proportion of change-

Algorithm	Parameters	Proportion changepoints detected	Average adjusted Rand index	Stability	Average reaction time
Sequential	$\lambda = 20$	68%	0.74	>**0.99**	399
Incremental	$\lambda = 20$	80%	**0.75**	>**0.99**	401
Exponential	$\gamma = 0.67, \lambda = 20$	80%	0.71	0.88	**361**
KC	$\sigma_1 = 1, \sigma_2 = 3, \lambda = 20$	**86%**	**0.75**	0.96	379

Table 1: Performance of the different algorithms on synthetic data over 50 runs. The best values have been highlighted.

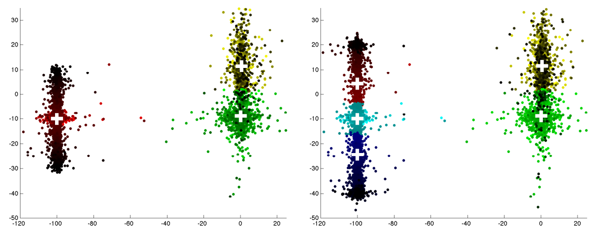

Fig. 2: KC $[\sigma_1 = 1, \sigma_2 = 3]$ before and after detection of the split at $t = 600$ (left) and $t = 800$ (right) respectively.

points successfully detected. There is a general trade-off between stability of the solution and speed of adaptation, as has already been pointed out in [4]. Both the incremental and sequential K-means had the highest reaction times, but also the highest stability. At the other end of the spectrum, the exponential clustering achieved the best reaction times, but the worst stability. Our solution struck the best compromise between stability and speed of adaptation having the second best average reaction time.

4.2. Real data

The issue with real datasets is that the ground truth is not known, but certain heuristics can be used to determine the effectiveness of the clustering. If there is a landmark or changepoint in the dataset, then a clustering algorithm should be able to detect it. We use the same dataset as [4], however we extracted different features. The dataset comes from [12], it is a study of mobile communications of one hundred students and academic staff at MIT between the 20th September 2004 and the 20th March 2005. We used location features of 85 mobiles during that time period that we sampled 1000 times. Each mobile kept a log of the cell tower ids it was receiving signal from and these ids were used as approximate locations. A new log was made whenever a mobile made a transition between neighbouring cell towers. We also used the run lengths of each object, as people did not update their location synchronously. We show here the number of clusters over the time period as estimated by KC with $[\sigma_1 = 1, \sigma_2 = 3, \lambda = 20]$. No optimisation was done to choose the parameters making overfitting unlikely. The number of clusters shows high variability over the time period. The clustering algorithm successfully identifies significant changepoints. The visible peaks and landmarks in the time series all represent weekends and holidays where the mobility of people increased significantly. Interestingly, the most significant changepoints are Thanksgiving (25th November) and the 31st December shown by stars.

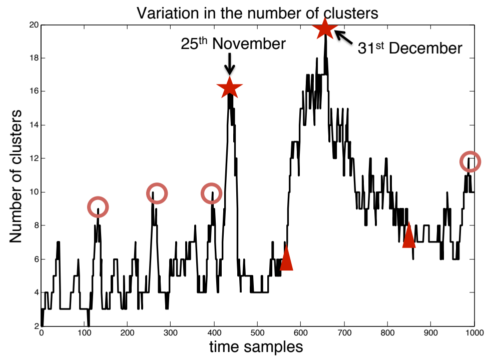

Fig. 3: All the circles represent landmarks, the stars are the most significant ones, the triangles are the start of winter vacation and spring term.

5. CONCLUSIONS

In this paper, we propose to apply a well-known multi-target tracking algorithm to a new evolutionary clustering problem with automatic detection of the number of clusters. This is the first time that this has been done to the best knowledge of the authors. We also made a systematic comparison with other well-known clustering algorithms in the 3 main desirable properties: accuracy, stability and speed of adaptation. We showed that our algorithm found the best compromise between stability and reaction time. We could extend the proposed solution using more sophisticated covariance structures in the Kalman filtering. Additionally, our algorithm could be formulated in a fully probabilistic framework. We leave this for future work.

6. ACKNOWLEDGEMENTS

We would like to thank Featurespace (who own all rights) for their collaboration and financial support for this work. We would like to thank Amelia Waddington for her insightful comments.

7. REFERENCES

[1] Y. Yang and B. Padmanabhan, "Segmenting customer transactions using a pattern-based clustering approach," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 411–418.

[2] Y. Yang and B. Padmanabhan, "Ghic: a hierarchical pattern-based clustering algorithm for grouping web transactions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 9, pp. 1300–1304, 2005.

[3] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 554–560.

[4] K.S. Xu, M. Kliger, and AO Hero, "Evolutionary spectral clustering with adaptive forgetting factor," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2174–2177.

[5] Y. Chi, X. Song, D. Zhou, K. Hino, and B.L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 153–162.

[6] K.S. Xu, M. Kliger, and A.O. Hero III, "Adaptive evolutionary clustering," *Arxiv preprint arXiv:1104.1990*, 2011.

[7] R.G. Sea, "An efficient suboptimal decision procedure for associating sensor data with stored tracks in real-time surveillance systems," in *Decision and Control, 1971 IEEE Conference on*. IEEE, 1971, vol. 10, pp. 33–37.

[8] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification and Scene Analysis 2nd ed.*, 1995.

[9] R.E. Kalman et al., "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[10] C.D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, vol. 1, Cambridge University Press Cambridge, 2008.

[11] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[12] N. Eagle, A.S. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15274–15278, 2009.