Dynamic Spectral Clustering

Amy LaViers, Amir Rahmani, and Magnus Egerstedt

Abstract— **Clustering is a powerful tool for data classification; however, its application has been limited to analysis of static snapshots of data which may be time-evolving. This work presents a clustering algorithm that employs a fixed time interval and a time-aggregated similarity measure to determine classification. The fixed time interval and a weighting parameter are tuned to the system's dynamics; otherwise the algorithm proceeds automatically finding the optimal cluster number and appropriate clusters at each time point in the dataset. The viability and contribution of the method is shown through simulation.**

I. Introduction

The search for structure within a dataset is an established problem with many solutions. Often, this problem amounts to finding a simpler, more natural representation of the data. Notably, grouping similar data into clusters has proved useful in computer vision, genomics, and network analysis. With an appropriate number of clusters, the task of representing the data is simply assignment to one of a (relatively) few distinct clusters. In this sense, clustering may be thought of as data compression: like methods such as principle component analysis (PCA), clustering exploits data structure to find a more succinct representation but does not rely on the same explicit linear relationships. An extensive overview of this community of academic work is given in a review by Schaeffer [1].

These methods are typically used to analyze a single, static dataset; however, many datasets are time-varying and possibly maintain clusters over time which may not be captured by instantaneous analysis alone. Static methods can provide insight to data structure at each time point, but this approach is computationally cumbersome and without a clear procedure for interpretation. Furthermore, to treat such data as a series of static clusters is equivalent to ignoring the additional information provided by time evolving data. For example, when tracking social behaviors biologists are interested in alliances that account for interactions over time; in fact, instantaneous clusters alone are meaningless for these evolving relationships as organisms may interact with alliances of which they are not a member [6].

With this weakness of current clustering methods in mind, we look to incorporate time-varying attributes of data into a clustering method. A particularly natural choice of clustering algorithm is that of Zelnik-Manor and Perona's "Self-Tuning Spectral Clustering" [2]. Unlike a more basic clustering method, such as k-means, where the number of clusters and interaction distances of the nodes in the data set must be assumed a priori, this algorithm automatically (and optimally) tunes these parameters. In time-varying data these are particularly important aspects as clusters may appear and disappear over time.

Thus, we present an extension to Zelnik-Manor and Perona's clustering method [2] that enables its application to data with time-varying attributes. A key feature of our algorithm is that it retains the desirable quality of all spectral clustering techniques in that it partitions data without presuming any direct scale to be characteristic of the data *a priori*. In fact, the final algorithm has only two parameters that can be tuned: a weighting factor on the similarity measure and the length of the time interval over which the dynamic algorithm is applied. Both parameters are robust to the type of changes that will occur as the data evolves in time; this result is theoretically pleasing and has also proven itself useful in practice.

II. Spectral Clustering

We begin with a finite dataset with n entries that form the set $\mathcal{V}$; hence $n = |\mathcal{V}|$. Onto this dataset we may apply a notion of abstract distance, $\delta : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$, that is defined between every pair of points (i, j) in the dataset $\mathcal{V}$ (let $\delta(i, j) = \delta_{ij}$). This distance is an alternate expression for the similarity measure between data points.

The structure that δ imposes over the data can be encoded in a matrix by defining a complete, weighted graph over the data points, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, where $\mathcal{E}$ is the edge set of $\mathcal{G}$ and the edge weights, w_{ij}, are a monotonically decreasing function of the distance measure between data points, δ_{ij}. For a δ that corresponds to Euclidean distance, points which are closest to each other have the biggest weights, are most similar in terms of location, and, thus, will be more likely to be in the same cluster.

We define the normalized weighted adjacency matrix as

$$L_N = \Delta^{-\frac{1}{2}} A \Delta^{-\frac{1}{2}} = I - \Delta^{-\frac{1}{2}} L \Delta^{-\frac{1}{2}}, \tag{1}$$

where the graph Laplacian is $L = \Delta - A$; $\Delta_{n\times n}$ is the diagonal degree matrix which is defined as

$$\Delta_{ij} = \begin{cases} deg(v_i) & i = j \\ 0 & i \neq j \end{cases}, \tag{2}$$

for our completely connected graph this is simply the identity matrix scaled by the number of nodes in $\mathcal{V}$, $n \cdot \mathbb{I}_{nxn}$; and $A_{n\times n}$ is the weighted adjacency matrix which encodes the distances between nodes and is defined as

$$A_{ij} = \begin{cases} w_{ij} & (v_i, v_j) \in E \\ 0 & o.w. \end{cases}. \tag{3}$$

Authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332; Emails: {*alaviers,arahmani,magnus*}*@gatech.edu*.

ISBN 978-963-311-370-7

Thus, L_N is a direct variant of the the graph Laplacian; we will use the spectral properties of this matrix to perform the clustering.

This matrix L_N contains the complete connectivity information of $\mathcal{G}$ and the similarity metric defined by δ. Since L is symmetric and positive semi-definite, we know that L_N is also symmetric and has eigenvalues belonging to the interval $(-\infty, 1]$. Furthermore, for the completely connected graph, the largest eigenvalue of L_N equals 1 and its associated eigenvector is a vector of ones (perhaps scaled by a constant), $\mathbf{1} \in \mathbb{R}^n$.

For further intuition about this matrix, consider an ideally clustered graph with c clusters where there is an edge-weight of zero between nodes which do not belong to the same cluster. As outlined in [7], L_N is now in block diagonal form with c blocks as in Eq. 4.

$$L_N = \begin{bmatrix} L_{N_1} & & & \\ & L_{N_2} & & \\ & & \ddots & \\ & & & L_{N_c} \end{bmatrix} \tag{4}$$

In this disconnected case, L_N has c eigenvalues equal to one. Correspondingly, the c eigenvectors associated with these characteristic eigenvalues define the clustering: if row i of eigenvector j contains a nonzero entry, then node i is in cluster j for $i = \{1, ..., n\}$ and $j = \{1, ..., c\}$. Every other entry of the eigenvector will be a zero and none of the cluster assignments are multiply defined. This structure is shown by:

$$V_{n\times c} = \begin{bmatrix} \mathbf{1}_1 & 0 & & 0 \\ 0 & \mathbf{1}_2 & & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & & \mathbf{1}_c \end{bmatrix}, \tag{5}$$

where the c eigenvectors corresponding to eigenvalues of 1 have been concatenated into a matrix, V and $\mathbf{1}_i$ is a vector of ones with dimension equal to the number of nodes in the ith connected component . We can also think of this ideal case as one where the eigenvectors have no inter-dependence when expressed in terms of the n dimensional space defined by the data points (and nodes of our constructed graph); that is, each basis vector is used once and only once to describe the eigenvectors.

Next, consider our graph $\mathcal{G}$, where $\mathcal{V} = \{1, ..., n\}$ are the nodes of the graph defined by n data points; $\mathcal{E} = \{1, ..., m\}$ is the edge set of a complete graph, K_n (where $m = \frac{n(n-1)}{2}$); and w_{ij} is a weight on edge $(v_i, v_j) \in \mathcal{E}$ that defines a monotonically decreasing function of distance (or monotonically increasing function of similarity) $\delta : \mathcal{E} \to \mathbb{R}$ between the two nodes.

There is one connected component in this graph; hence one eigenvalue of L_N is equal to 1. A first look at the eigenvectors in the form of Eq. 5 shows a single vector of ones describing one uninformative cluster; however, the data may still have underlying clusters. These clusters are determined by groups of data points which are close to each other (where closeness is defined by δ since some edge weights will be small relative to the others).

Since the eigenvalues and eigenvectors vary continuously with modifications to the matrix entries, the number of relatively large eigenvalues of L_N (close to one) gives a good, initial guess of the number of clusters, c, in the dataset. Again, only one eigenvalue of L_N equals 1 (since there is one connected component in a complete graph), but the other eigenvalues corresponding to highly clustered regions of the graph, which have now drifted, remain close to 1.

Likewise, the eigenvectors that before gave a definite clustering of the graph's clusters are now skewed from the ideal case. The effect of this skew can be reversed or reduced by rotating the new eigenvectors until they are close to the ideal case again. As before, we would like to express these eigenvectors with the smallest possible inter-dependence in the n-dimensional space of the dataset.

Let the first c eigenvectors of L_N compose $V \in \mathbb{R}^{n\times c}$. Rotating V such that each row of the new, rotated matrix Z has only one non-zero entry provides the cluster assignment for each node. Defining this rotation by a matrix $R(\Theta)_{c\times c}$, we want to find a vector $\Theta \in \mathbb{R}^{c(c-1)/2}$ where each entry is an angle in $[-\frac{\pi}{2}, \frac{\pi}{2})$ such that it minimizes a cost function,

$$\min_{\Theta} \mathcal{J} = \Sigma_{i=1}^{n} \Sigma_{j=1}^{c} (\frac{Z_{ij}}{M_i})^2, \tag{6}$$

subject to the constraint

$$Z_{n\times c} = V_{n\times c} R(\Theta)_{c\times c}. \tag{7}$$

To achieve clear clustering assignments one term on each row of Z should be large relative to the other entries of that row; in the ideal case, this corresponds to only one nonzero entry per row (as in Eq. 5). Thus, we set $M_i = \max_j |Z_{ij}|$ in order for the optimization to achieve the desired structure in Z. This optimization can be solved using the gradient descent approach proposed by Goyal [3] with the update rule:

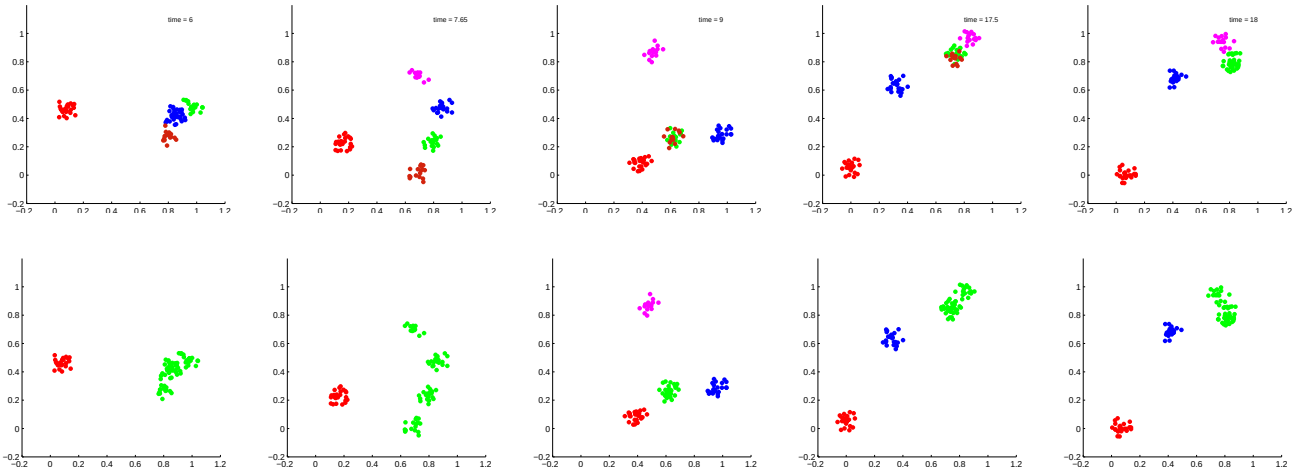
$$\Theta_{k+1} = \Theta_k - \alpha \nabla \mathcal{J}|_{\Theta=\Theta_k}. \tag{8}$$

After this rotation node i belongs to the cluster given by $\arg\max_j |Z_{ij}|$.

Which number of clusters produces the rotation closest to the ideal case? To answer this, we consider the *quality* of each clustering. Quality can be defined as

$$q(c, n) = 1 - \left(\frac{\mathcal{J}}{n} - 1\right). \tag{9}$$

Note that n is the minimum of $\mathcal{J}$ for the ideally clustered case, and in that case q is equal to 1. Thus, clustering quality is correspondingly farther from 1 for poorer cluster assignments. Since the number of nodes $n = |\mathcal{V}|$ is constant, $\arg\max_c q(c, n)$ provides the optimum cluster number. This results in a number of clusters chosen independent of human choice or prejudice.



(a) t=6 (b) t=7.65 (c) t=9 (d) t=17.5 (e) t=18

Fig. 1. A comparison of previous, static methods and our choice of dynamic weights: spectral clustering using aggregated distances (top) using instantaneous distances (bottom).

III. Dynamic Data Clustering

Previous works have used repeated patterns in clusters assigned at periodic time points as a way to identify resilient clusters [5]. Clustering at each time step and basing the dynamic case on the frequency of each static cluster places undue importance on these intermediate clusterings. In particular, since the relationships of nodes inside the intermediate clusterings is lost in this method, it is not clear what the interpretation of these intermediate clusters is. That is, information about nodes which are close to each other inside a larger cluster is lost with this type of analysis.

Instead, we look for an algorithm that captures the information contained within these larger clusters at each time point. Our choice of weight - an aggregate, pairwise distance between nodes - tracks the relative positions of each node as it moves between other clusters and among other nodes and retains that information throughout a desired time interval. In other words, our algorithm saves the information about inter-agent relationships over some time rather than considering only the movement of agents between clusters.

We define the distance between two dynamically evolving points in our dataset, $d : \mathcal{V}(t) \times \mathcal{V}(t) \to \mathbb{R}$, as a function of their instantaneous distance over a window of time leading to current time (let $d(i,j) = d_{ij}$). In general:

$$d_{ij}(k) = f(\delta_{ij}(0), \cdots, \delta_{ij}(k)). \tag{10}$$

Specifically, we choose this function to simply sum the distances over time, i.e.

$$d_{ij}(k) = \sum_{m=k-l}^{k} \delta_{ij}(m), \tag{11}$$

over a sliding time period $[k-l, k] \in [t_0, t_f]$. We redefine our graph to contain the aggregated information of inter-agent distance over a period of time as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \hat{w}(k))$. Here we define new weights of the form

$$\hat{w}_{ij}(k) = -\exp\left(\frac{d_{ij}^2(k)}{\sigma_i(k)\sigma_j(k)}\right), \tag{12}$$

where $\sigma_i(k)$ is a local scaling factor for the inter-agent distances and is chosen to be the aggregated distance to the 7th nearest neighbor to node i. This choice of weight is inherited from [2] and will be discussed further below. Now, the data points are described in terms not of their absolute positions (or similarities), but in terms of net movement over the sliding window. However, the algorithm still tries to reduce the inter-dependence of the new, aggregate eigenvectors in the same n-dimensional space.

This choice of weight self-tunes the scale of the algorithm to the scale inherent in the data. Using the 7th nearest neighbor biases the method to assign clusterings where the number of nodes in each cluster is on the order of 7; while it has proven useful for a wide range of data sets of various sizes of clusters [2], if the number of nodes in a cluster should be more on the order of 7 million in an ill defined data set (where the difference in the weights inside and out of a given cluster is small), this parameter may need to be adjusted though it is expected to remain constant for one time varying data set. This implements Zelnik-Manor and Perona's second main contribution to spectral clustering.

In our dynamic extension, we have introduced a second parameter, the length of the sliding time window l, which, while invariant to many types of data or small changes in a given dataset, may need to be tweaked for individual dataset. This window represents the algorithm's ability to use the past temporal information to influence clusterings. Thus, it should be sized according to the rate of change of the data and to any notion of memory the system has. That is, the window should be short enough so that quickly changing data is able to form and reform clusters - the lack of any window may

result in equal weights across all edges as after some point each node has undergone a similar net overall change. Even for a very slowly evolving system, the window should remain small enough to accurately model the fact that nodes "forget" past interactions after some time.

IV. Simulation Results

To compare the performance of our dynamic clustering to that of the static one, we generated a 100 point dataset with groups of points moving in $\mathbb{R}^2$. The distance metric was defined as the Euclidean distance between the position of each two points,

$$\delta_{ij}(k) = \|x_i(k) - x_j(k)\|, \tag{13}$$

summed over a time window of 10 seconds. Figure 1 depicts the results of the proposed clustering algorithm for the dynamic case on the top row versus that of the static case on the bottom row.

The figure illustrates our algorithm's ability to keep track of the cluster associations over time. Consider the three clusters which converge at time t=6: dynamic clustering shows a blue, burgundy, and green cluster while static clustering shows them as one green cluster. In the case where clusters converge or diverge and remain close or far for some time, the dynamic algorithm identifies this as a cluster merge or split, respectively. These phenomena are visible at times t=18 when the green and burgundy clusters have merged and t=7.65 with the emergence of the pink cluster.

V. Conclusions

Clustering is an established tool, with many variations, that has proven itself useful in a diversity of applications where classification of data into distinct groups is desired. Many applications involve data points which are time evolving and, thus, may become more suited to a different classification as time passes. Furthermore, the dataset may evolve such that the original discrete classifications are no longer apt for representing the data and new classifications may become necessary.

These applications present new questions and challenges for current clustering algorithms. Previous dynamic applications have considered the intermediate static clusterings of time-evolving data and created a dynamic characterization from these static clusters. The inherently dynamic algorithm presented here redefines the method for encoding dataset structure into a formal graph. Specifically, a new approach for weighting the graph edges allows a spectral clustering algorithm to capture any time-varying aspect of the data.

This algorithm retains some computational overkill. Namely, it is a reasonable assumption that clusterings do not change much from time point to time point. Hence, saving the optimal rotation and cluster number from the previous time step would reduce some computation and allow for real time data processing. Furthermore, difference equations can provide an update rule for the evolution of L_N which would result in a dynamic model of this clustering system. Finally, the algorithm still has yet to prove itself in many real-world applications such as animal interactions and video segmentation. Such extensions to this work would enhance the power of this new, dynamic clustering method.

Acknowledgment

This work is partially supported by the Office of Naval Research through the MURI-HUNT.

References

[1] S. E. Schaeffer. "Graph clustering," *Computer Science Review I*, pp. 27-64, 2007.

[2] L. Zelnik-Manor and P. Perona, "Self-Tuning Spectral Clustering," *Proceedings of Advances in Neural Information Processing Systems*, 2005.

[3] V. K. Goyal and M. Vetterli. "Block transform adaptation by stochastic gradient descent," *IEEE Digital Signal Processing Workshop*, Bryce Canyon, UT, Aug. 1998.

[4] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems*, 14, 2002.

[5] M. Lahiri and T. Y. Berger-Wolf. "Mining periodic behavior in dynamic social networks," *IEEE Computer Society*, pp. 373-382, 2008.

[6] S. Sundaresan, I. Fischhoff, J. Dushoff & D. Rubenstein. "Network metrics reveal differences in social organization between two fission-fusion species, Grevys zebra and onager," *Oecologia*, 151(1), 2007.

[7] U. Von Luxburg. "A Tutorial on Spectral Clustering," *Statistics and Computing*, 17(4), 2007.