EVOLUTIONARY AFFINITY PROPAGATION

Natalia M. Arzeno and Haris Vikalo

Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX, USA

ABSTRACT

Evolutionary affinity propagation, an evolutionary clustering algorithm that groups data points by exchanging messages on a factor graph, is proposed. The algorithm promotes temporal smoothness of the clustering solutions at distinct temporal snapshots by linking variable nodes of the graph across time, and is capable of detecting cluster births and deaths. Unlike most existing evolutionary clustering methods that require additional processing in order to approximate the number of clusters, evolutionary affinity propagation determines the number of clusters automatically. A comparison with existing methods on simulated and experimental data demonstrates accuracy and robustness of the proposed framework.

***Index Terms*—** affinity propagation, time series data, evolutionary clustering

1. INTRODUCTION

In recent years, traditional clustering algorithms such as k-means, spectral clustering, and agglomerative clustering have been adapted to the evolutionary clustering setting [1–3]. The goal of evolutionary clustering is to discover structure in data collected over multiple points in time while taking into account the correlated nature of the data generation process. To this end, evolutionary clustering algorithms modify the objective of traditional clustering problems to promote sustained cluster membership across multiple time points, yielding results that are typically more informative and accurate than those achieved by static methods treating temporal snapshots of data independently [1, 4, 5]. Evolutionary clustering methods have been used in a range of different applications including prediction of links between blogs [1], identifying communities of spammers [6], tracking parameters of multi-antenna communication systems [7] and oceanography [8].

Evolutionary k-means and evolutionary agglomerative hierarchical clustering algorithms, both proposed in [1], consider the clustering objective that consists of a term reflecting result of clustering at the current time step, and a historical cost term dependent upon clustering configuration at the previous time step [1]. Evolutionary spectral clustering [2, 3] also adopts an objective that combines snapshot and temporal cost terms, formulating the latter in different ways depending on whether one wants to emphasize cluster quality or promote continuity of cluster membership. Xu et al. proposed AFFECT [5], an evolutionary clustering framework where the similarity matrix at each time step is formed as the sum of a deterministic matrix and a Gaussian noise matrix. AFFECT enables adaptation of classic k-means, agglomerative, and spectral clustering algorithms to evolutionary setting [5], and allows optimizable adaptive weight of the temporal cost term in the objective function. Recently, Kim et al. proposed a Temporal Multinomial Mixture for temporal clustering of categorical data streams [9]. Ahmed and Xing [4] proposed a clustering algorithm that relies on a temporal Dirichlet process mixture model, where the cluster parameters can evolve in Markovian fashion and the posterior optimal cluster evolution is inferred by a Gibbs sampling algorithm. Xu et al. proposed an evolutionary clustering method that combines a hierarchical Dirichlet process with a hierarchical transition matrix from an infinite hierarchical hidden Markov model [10].

A major challenge for most evolutionary as well as traditional clustering methods is the requirement to infer the number of clusters, which is often done heuristically. Evolutionary clustering methods that can automatically decide the number of clusters typically rely on Dirichlet process models [4, 10]. In an ideal case, evolutionary clustering methods should permit varying numbers of clusters, i.e., they should allow clusters to be born, evolve, or die at each time step. Moreover, they should be capable of handling data points that appear or disappear over time. There exist algorithms that can satisfy some [3, 5] or most of these requirements [4, 11] but practically feasible solutions remain elusive.

We propose evolutionary affinity propagation (EAP), an evolutionary clustering algorithm that builds upon ideas of static affinity propagation to cluster data acquired at multiple time points by passing messages on a factor graph linking temporal data snapshots. EAP automatically determines the number of clusters, detects cluster births and deaths, and accurately tracks clusters across time. Moreover, the algorithm can handle non-metric similarities and can be efficiently implemented for large, sparse datasets. Note that the only other evolutionary clustering methods that automatically detect the number of clusters use Dirichlet process models [4, 10] and focus on inferring the parameters of the cluster distributions, while EAP focuses on data and their cluster assignments.

2. BACKGROUND ON AFFINITY PROPAGATION

Affinity propagation (AP) takes similarities between data points and, by passing messages on a graph, finds the best exemplar for each point and thus a cluster assignment for the data [12]. The goal of AP is to maximize the total similarity between points, or data instances, and their exemplars (i.e., cluster representatives) subject to some constraints. The factor graph used by AP, illustrated in Fig. 1, consists of variable nodes c_{ij} that take on binary values and indicate if point j is an exemplar of point i and factor nodes $I_i(c_{i1}, \ldots, c_{iN})$ and $E_j(c_{1j}, \ldots, c_{Nj})$ that enforce single-cluster membership and self-selection (if j is an exemplar for i then it must also be an exemplar for itself), respectively [13]. To elect exemplars and form clusters, AP requires exchange of only two messages, responsibility and availability, between data points. The responsibility ρ_{ij} indicates suitability of point j to be an exemplar for point i while the availability α_{ij} contains evidence why point i should choose j as an exemplar,

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max[\rho_{kj}, 0] & \text{if } i = j, \\ \min\left[0, \rho_{jj} + \sum_{k \neq \{i,j\}} \max[\rho_{kj}, 0]\right] & \text{if } i \neq j, \end{cases}$$

$$\rho_{ij} = s_{ij} - \max_{k \neq j}(\alpha_{ik} + s_{ik}).$$

The number of clusters is automatically inferred by AP and can be tuned via self-similarity, or preference, of the data points if prior information is available. AP does not require that similarities be metric, and it can be efficiently implemented for large, sparse datasets.

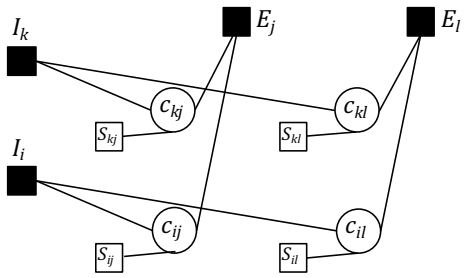

Fig. 1. Affinity propagation factor graph, where $i, j, k, l \in \{1, \ldots, N\}$ for N data points.

In recent years, a number of AP extensions have been proposed including semi-supervised clustering with strict [14, 15] or soft [16] pairwise constraints, relaxation of the self-selection constraint [15, 17], hierarchical AP [18], AP with identification of subclasses [19], and fast AP with adaptive message updates [20]. AP has also been used to cluster temporal data in [21], where modified availability messages are used to impose preference of assigning points at time $t+1$ to the same exemplar as at time t. However, this scheme does not impose backward temporal smoothness and would require additional post-processing steps to attempt tracking clusters.

3. EAP ALGORITHM

The proposed algorithm clusters points by exchanging messages on the factor graph shown in Fig. 2. Unlike the conventional AP where the points at each time step are clustered independently from those observed at other time steps, EAP relies on additional factor nodes to establish connection between the variable nodes at consecutive time steps thus promoting temporal smoothness of the cluster assignments. These nodes, denoted as D_{ij}^t in Fig. 2, penalize clustering configurations where data points change exemplars in consecutive time steps.

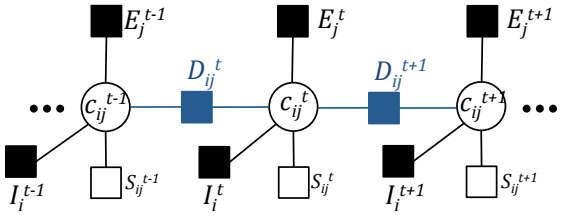

Fig. 2. Factor graph of evolutionary affinity propagation

To promote solutions where an exemplar is consistently associated with a cluster, EAP introduces consensus nodes (see Section 3.1) and employs factors that encourage points to select a consensus node, rather than a data point, as their exemplar. At any time step, EAP's responsibility messages depend upon messages from previous and subsequent time steps. The final composition of clusters stems from considering all points as potential exemplars at all time steps while promoting exemplar stability and temporal smoothness. Note that imposing exemplar stability enables tracking time evolution of clusters and eliminates need for a computationally intensive cluster matching step such as the one used in [5].

Specific messages exchanged between the nodes of the factor graph in Fig. 2 are given next. As in the factor graph for AP in Fig. 1, variable c_{ij}^t takes on value 1 if j is the exemplar for i and is 0 otherwise. The factor node I_i^t ensures that each data point is assigned to only one cluster, E_j^t enforces the constraint that if j is an exemplar for any $i \neq j$ then j must also be an exemplar for itself, and S_{ij}^t passes the similarity between a point and its exemplar (i.e., communicates s_{ij}^t). To encourage temporal smoothness, we penalize changes in clusters and reward assignments to nodes in the consensus node set C^t (creation and evolution of consensus nodes is discussed in Section 3.1). This is accomplished by setting

$$D_{ij}^t(c_{ij}^{t-1}, c_{ij}^t) = \begin{cases} -\gamma & \text{if } c_{ij}^{t-1} \neq c_{ij}^t \\ 0 & \text{if } c_{ij}^{t-1} = c_{ij}^t = 1 \text{ and } j \in C^t \\ -\omega & \text{otherwise.} \end{cases} \quad (1)$$

Note that, unlike in the traditional AP formulation, the values of nodes in the EAP graph are time-dependent.

A message m_{ij} is defined as $m_{ij} = m(c_{ij} = 1) - m(c_{ij} = 0)$. Starting from the max-sum update rules [22] and performing substitutions and simplifications of update equations (omitted for brevity), we find that there are only four messages then need to be computed and exchanged as EAP clusters points at time t: α_{ij}^t (between E_j^t and c_{ij}^t), ρ_{ij}^t (between c_{ij}^t and E_j^t), ϕ_{ij}^t (between D_{ij}^{t+1} and c_{ij}^t), and δ_{ij}^t (between D_{ij}^t and c_{ij}^t). The availability α_{ij}^t remains unchanged from static AP (see Section 2) whereas the responsibility ρ_{ij}^t is dependent upon new messages $\delta_{ij}^t, \phi_{ij}^t$,

$$\alpha_{ij}^t = \begin{cases} \sum_{k\neq j} \max[\rho_{kj}^t, 0] & \text{if } i = j \\ \min\left[0, \rho_{jj} + \sum_{k\neq\{i,j\}} \max[\rho_{kj}, 0]\right] & \text{if } i \neq j. \end{cases}$$

$$\rho_{ij}^t = s_{ij}^t + \eta_{ij}^t + \phi_{ij}^t + \delta_{ij}^t.$$

For every t, δ_{ij}^t depends on the previous time step whereas ϕ_{ij}^t depends on the following time step, with $\delta_{ij}^t = 0$ for all i, j in the first time step and $\phi_{ij}^t = 0$ for all i, j in the last time step. Using the max-sum message update rules, we find

$$\delta_{ij}^t = \begin{cases} -\gamma + \omega & \text{if } d1 = 1, d2 = 1 \\ \omega \mathbb{1}(j \in C^t) + \rho_{ij}^{t-1} + \alpha_{ij}^{t-1} - \phi_{ij}^{t-1} & \text{if } d1 = 1, d2 = 0 \\ -\rho_{ij}^{t-1} - \alpha_{ij}^{t-1} + \phi_{ij}^{t-1} & \text{if } d1 = 0, d2 = 1 \\ \gamma - \omega \mathbb{1}(j \notin C^t) & \text{if } d1 = 0, d2 = 0, \end{cases}$$

where we define $d1 = \mathbb{1}\big(\gamma - \omega \geq \rho_{ij}^{t-1} + \alpha_{ij}^{t-1} - \phi_{ij}^{t-1}\big)$, $d2 = \mathbb{1}\big(-\gamma + \omega \mathbb{1}(j \notin C^t) \geq \rho_{ij}^{t-1} + \alpha_{ij}^{t-1} - \phi_{ij}^{t-1}\big)$, and $\mathbb{1}$ denotes an indicator function. Using the same max-sum message update rules leads to

$$\phi_{ij}^{t-1} = \begin{cases} -\gamma + \omega & \text{if } p1 = 1, p2 = 1 \\ \omega \mathbb{1}(j \in C^t) + \rho_{ij}^t + \alpha_{ij}^t - \delta_{ij}^t & \text{if } p1 = 1, p2 = 0 \\ -\rho_{ij}^t - \alpha_{ij}^t + \delta_{ij}^t & \text{if } p1 = 0, p2 = 1 \\ \gamma - \omega \mathbb{1}(j \notin C^t) & \text{if } p1 = 0, p2 = 0, \end{cases}$$

where $p1 = \mathbb{1}\big(\gamma - \omega \geq \rho_{ij}^t + \alpha_{ij}^t - \delta_{ij}^t\big)$ and $p2 = \mathbb{1}\big(-\gamma + \omega \mathbb{1}(j \notin C^t) \geq \rho_{ij}^t + \alpha_{ij}^t - \delta_{ij}^t\big)$. Let us define the set of exemplars $E = \{j : \alpha_{jj}^t + \rho_{jj}^t + \delta_{jj}^t + \phi_{jj}^t > 0\}$. Then the exemplar j for point i is identified as

$$\arg\max_{j \in E} \ \alpha_{ij}^t + \rho_{ij}^t + \delta_{ij}^t + \phi_{ij}^t. \tag{2}$$

Complexity of EAP. Since the number of consensus nodes is much smaller than the number of data points N, the computational complexity of an EAP iteration (which involves exchanging messages $\alpha, \rho, \delta, \phi$ between the nodes in each of T time steps) is $O(N^2T)$. The complexity of creating and updating consensus nodes (see Section 3.1) does not exceed $O(N^2T)$. Note that running an iteration of the classic (static) AP over T time steps is of the same complexity, $O(N^2T)$. Also note that when N is large and the similarity matrix is sparse, EAP messages need not be passed between all pairs of points and thus the complexity can be reduced.

3.1. Creating and tracking consensus nodes

EAP starts with a burn-in period where messages are passed among the data points in the forward-backward fashion until at least 2 exemplars are identified for each time step. Following the burn-in period, a consensus node i' is created for each data point i identified as an exemplar. The feature values of the consensus node are defined as the mean of the features of all the points assigned to the exemplar i and its message values are initialized based on the identified data point exemplars, where the availabilities require special handling (the maximum value of α_{lk}^t is 0 when $l \neq k$). The consensus node creation is formalized as Algorithm 1 for the forward pass of a single iteration, where e_i^t denotes the exemplar assigned to point i at time t and x_k^t is the feature vector of point k at t.

Algorithm 1 Cluster birth: Creation of consensus nodes

$V^t \leftarrow$ set of data points at time t
$C^t \leftarrow$ set of consensus nodes at time t
$E^t \leftarrow$ set of exemplars at time t
for $i \in V^t \cap E^t$ **do**:
create consensus node i' at time t: $x_{i'}^t \leftarrow \sum_{j:e_j^t=i} x_i^t$
initialize message values of i' to those of i: $m_{i'j}^t \leftarrow m_{ij}^t$, $m_{ji'}^t \leftarrow m_{ji}^t$, for $j \in V^t \cup C^t, m \in \{\alpha, \rho, \delta, \phi\}$
$m_{i'i'}^t \leftarrow m_{ii}^t$, for $m \in \alpha, \rho, \delta, \phi$
update $\alpha_{i'i}^t$ and $\alpha_{ii'}^t$:
$y \leftarrow \arg\max_{j \in V^t \setminus i} \alpha_{ij}^t + \rho_{ij}^t + \delta_{ij}^t + \phi_{ij}^t$, $\alpha_{i'i}^t \leftarrow \alpha_{iy}^t$, $\alpha_{ii'}^t \leftarrow 0$
update exemplars to replace i with i'
end for
initialize consensus nodes at next time step
for $k \in C^t \setminus C^{t+1}$ **do**:
$x_k^{t+1} \leftarrow \sum_{j:e_j^t=k} x_j^{t+1}$
$l \leftarrow \arg\max_{j \in E^{t+1}} \sum_{i \in V^t} \mathbb{1}(e_i^t = k)\mathbb{1}(e_i^{t+1} = j)$
set messages for k at $t+1$ using messages for l at $t+1$ following initialization of i' messages above
end for

The exemplars in EAP are identified in the forward pass, after the messages are updated. When the set of consensus nodes $C^t \neq \emptyset$, consensus nodes are favored as exemplars if $\alpha_{ik}^t + \rho_{ik}^t + \delta_{ik}^t + \phi_{ik}^t > 0$ for any $k \in E \cap C^t$, where E is the set of exemplars. To track clusters, an additional update is performed when a consensus node takes on a data point as an exemplar. In the exemplar assignment, if a consensus node k does not identify itself as an exemplar but rather has data point i as an exemplar, the consensus node takes on the message values of i and the data points assigned to i as an exemplar are re-assigned to the consensus node k.

If an existing consensus node is not selected as an exemplar, the cluster corresponding to the consensus node is considered to have died. Once a cluster has died, it may be "revived" only in the case of frequent change of exemplars before the message values converge.

4. EXPERIMENTAL RESULTS

We test the performance of EAP on both synthetic and real data and compare it to those of the AFFECT's evolutionary spectral clustering [5] as well as to the classic AP (AP is applied to each temporal snapshot of data independently). For both EAP and AP, the similarity between points is defined as the negative squared Euclidean distance. The same is done for AFFECT when applied to real dataset but for synthetic data AFFECT uses similarity between x_i and x_j defined as $\exp(-\|x_i - x_j\|_2^2/2\sigma^2)$, with the default value $2\sigma^2 = 5$. The latter is due to our observation that AFFECT has significantly better performance on synthetic data when using Gaussian kernel similarities rather than negative squared Euclidean distance. AFFECT was implemented using the Matlab toolbox provided by its authors. Note that the AFFECT framework was previously shown to outperform evolutionary k-means, evolutionary spectral clustering, and the method in [23]. Accuracy is evaluated by means of Rand index [24], defined as the percentage of pairs of points that are correctly classified as being either in the same cluster or in different clusters.

First, we consider four 2-dimensional Gaussian mixture models used to generate synthetic datasets in [5], each with 200 data points. At every time step t, points in each of the components are drawn from the corresponding Gaussian distributions. The first dataset is generated from two well-separated Gaussians over 40 time steps. At each step, the first dimension of the mean of each component is altered by a random walk. The second dataset is generated from two colliding Gaussians with the initial means $[-3, -3]$ and $[3, 3]$ and identity covariance. For $t = 2, \ldots, 9$, the mean of the first component is increased by $[0.4, 0.4]$ and kept constant thereafter. The third dataset is generated in a similar way as the second one, with the difference that at $t = 10$ and $t = 11$ points in the second component switch to the first component with probability 0.25. From $t = 12$ to $t = 25$, the data points maintain the membership they had at $t = 11$. Finally, the fourth dataset is generated the same way as the second one for the first 9 time steps. For $t = 10$ and $t = 11$, data points in the second cluster switch membership with a probability of 0.25 to a new third Gaussian component with mean $[-3, -3]$ and identity covariance.

EAP achieved near-perfect clustering and correctly tracked clusters for all 4 datasets; it outperformed AFFECT with spectral clustering for the datasets having points changing clusters and points forming a third separate cluster, providing significantly higher accuracy during the steps that follow cluster membership change. When analyzing the fourth dataset, AFFECT does not detect emergence of the new cluster until $t = 18$ even though it was formed at $t = 10$; EAP detects the new cluster by $t = 12$. The average Rand index for the three methods is shown in Table 1.

Next, we test EAP on a real dataset consisting of daily closing stock prices from January to June 2000 for 3424 stocks obtained from the CRSP/Compustat Merged Database [25].[1] Feature vectors were constructed using piecewise normalized derivatives [26], previously successfully used for evolutionary clustering in [5]. Stocks were clustered with EAP, AP, the AFFECT framework with spectral clustering, and static spectral clustering. To avoid bias towards solutions with higher number of clusters, modified Rand index (modRand) [14, 16] is used in addition to Rand index for performance characterization and comparison. In the modified Rand index (modRand $\in [0, 1]$) pairs of points correctly identified as being in different clusters can account for no more than half of the total score. Average clustering results for the 6 months using industry sectors as the "true" labels [5] are presented in Table 2. EAP achieves higher modified Rand index than AFFECT, static AP, and static spectral clustering. Performing analysis of stock data via evolutionary clustering provides insight into the dynamics of stocks and helps discover groups of stocks behaving similarly during a market regime switch, which is of importance in portfolio diversification tasks. Further details are omitted for brevity.

Dataset	EAP	AP	AFFECT
separated Gaussians	1	1	1
colliding Gaussians	1	0.943	1
cluster change	0.998	0.879	0.964
third cluster	0.999	0.971	0.963

Table 1. Accuracy in terms of Rand index for synthetic data

Algorithm	Rand	modRand	no. of clusters
EAP	0.858	0.562	50-67
AFFECT	0.799	0.515	10
AP	0.861	0.530	107-115
spectral	0.797	0.509	10

Table 2. Rand and modRand indices when clustering stocks

5. CONCLUSION

We developed evolutionary affinity propagation (EAP), an evolutionary clustering algorithm which groups points by passing messages on a factor graph. The EAP graph introduces factors connecting variable nodes across time, promoting clustering solutions characterized by temporal smoothness. The algorithm can identify cluster births and deaths as well as track clusters across time. In the experiments on both synthetic and real datasets, EAP outperforms an evolutionary spectral clustering algorithm as well as the individual time step clustering by AP, yielding more accurate and interpretable solutions than competing methods. EAP's features make it a desirable choice in applications interested in discovery of structure in data acquired at multiple time steps.

[1] This period was chosen to include the dot-com bubble burst in 03/2000.

6. REFERENCES

[1] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006, pp. 554–560, ACM.

[2] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2007, pp. 153–162, ACM.

[3] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng, "On evolutionary spectral clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 4, pp. 17:1–17:30, Dec. 2009.

[4] A. Ahmed and E. Xing, "Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering," in *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 219–230. Society for Industrial and Applied Mathematics, Apr. 2008.

[5] Kevin S. Xu, Mark Kliger, and Alfred O. Hero III, "Adaptive evolutionary clustering," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 304–336, Jan. 2013.

[6] Kevin S. Xu, Mark Kliger, and Alfred O. Hero III, "Tracking communities of spammers by evolutionary clustering," in *International Conference on Machine Learning Workshop on Social Analytics: Learning from Human Interactions*, 2010.

[7] N. Czink, Ruiyuan Tian, S. Wyne, F. Tufvesson, J.-P. Nuutinen, J. Ylitalo, E. Bonek, and A.F. Molisch, "Tracking time-variant cluster parameters in MIMO channel measurements," in *Second International Conference on Communications and Networking in China, 2007.*, Aug. 2007, pp. 1147–1151.

[8] Stephan Günnemann, Hardy Kremer, Charlotte Laufkötter, and Thomas Seidl, "Tracing evolving subspace clusters in temporal climate data," *Data Mining and Knowledge Discovery*, vol. 24, no. 2, pp. 387–410, Sept. 2011.

[9] Young-Min Kim, Julien Velcin, Stéphane Bonnevay, and Marian-Andrei Rizoiu, "Temporal multinomial mixture for instance-oriented evolutionary clustering," in *Advances in Information Retrieval*, Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr, Eds., number 9022 in Lecture Notes in Computer Science, pp. 593–604. Springer International Publishing, Mar. 2015.

[10] Tianbing Xu, Zhongfei Zhang, P.S. Yu, and B. Long, "Evolutionary clustering by hierarchical dirichlet process with hidden markov state," in *Eighth IEEE International Conference on Data Mining, 2008. ICDM '08*, Dec. 2008, pp. 658–667.

[11] F. Folino and C. Pizzuti, "An evolutionary multiobjective approach for community discovery in dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1838–1852, Aug. 2014.

[12] Brendan J. Frey and Delbert Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

[13] Inmar E Givoni and Brendan J Frey, "A binary variable model for affinity propagation," *Neural computation*, vol. 21, no. 6, pp. 1589–1600, June 2009.

[14] Inmar E. Givoni and Brendan J. Frey, "Semi-supervised affinity propagation with instance-level constraints," in *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, 2009, pp. 161–168.

[15] Michele Leone, Sumedha, and Martin Weigt, "Unsupervised and semi-supervised clustering by message passing: soft-constraint affinity propagation," *The European Physical Journal B*, vol. 66, no. 1, pp. 125–135, Oct. 2008.

[16] Natalia M. Arzeno and Haris Vikalo, "Semi-supervised affinity propagation with soft instance-level constraints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 5, pp. 1041–1052, May 2015.

[17] Michele Leone, Sumedha, and Martin Weigt, "Clustering by soft-constraint affinity propagation: applications to gene-expression data," *Bioinformatics*, vol. 23, no. 20, pp. 2708–2715, Oct. 2007.

[18] Inmar Givoni, Clement Chung, and Brendan J. Frey, "Hierarchical affinity propagation," arXiv e-print 1202.3722, Feb. 2012.

[19] Chang-Dong Wang, Jian-Huang Lai, Ching Y Suen, and Jun-Yong Zhu, "Multi-exemplar affinity propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2013.

[20] Yasuhiro Fujiwara, Makoto Nakatsuji, Hiroaki Shiokawa, Yasutoshi Ida, and Machiko Toyoda, "Adaptive message update for fast affinity propagation," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2015, pp. 309–318, ACM.

[21] Vicenç Quera, Francesc S. Beltran, Inmar E. Givoni, and Ruth Dolado, "Determining shoal membership using affinity propagation," *Behavioural Brain Research*, vol. 241, pp. 38–49, Mar. 2013.

[22] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, Aug. 2006.

[23] J. Rosswog and K. Ghose, "Detecting and tracking spatio-temporal clusters with adaptive history filtering," in *IEEE International Conference on Data Mining Workshops*, Dec. 2008, pp. 448–457.

[24] William M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, Dec. 1971.

[25] The University of Chicago Booth School of Business, "Center for research in security prices (CRSP)," 2015.

[26] Martin Gavrilov, Dragomir Anguelov, Piotr Indyk, and Rajeev Motwani, "Mining the stock market: Which measure is best?," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2000, pp. 487–496, ACM.