ADM ASSIGNMENT-1
Toeplitz Inverse Covariance-Based Clustering

D.S. Archana   Sanchita Badkas

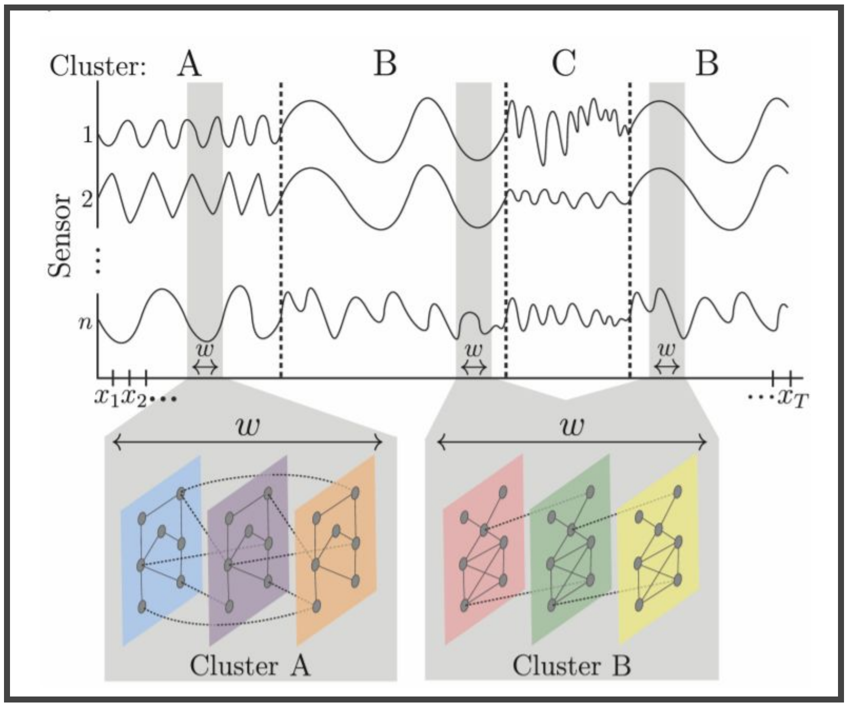
Introduction:

Time series like financial markets, wearable sensors, automobile sensors, etc generate a huge amount of high dimensional data. Due to the ever-evolving nature of these domains, the data patterns are dynamic. Since these domains are governed by multiple factors, the incoming data also belongs to multiple sources. This data can be represented by a timeline of a few key states. For e.g Data generated from wearable sensors can be represented by some key states like, sitting, walking, running. Similarly, in the case of sensors in automobiles, this can be represented by states like accelerating, speeding up, slowing down, etc. This helps us discover patterns in a better way in high dimensional data. Thus, simultaneous segmentation and clustering of the time series is required. The problems faced in order to achieve this can be listed as

- Multiple segments can belong to the same cluster
- Each data point can’t be clustered independently as neighboring points are encouraged to belong to the same cluster
- Clusters aren’t known apriori making it difficult to understand what each cluster refers to.

This calls for a need of an unsupervised clustering algorithm, to discover the states while simultaneously breaking the time series into the sequence of these states.

Toeplitz Inverse Covariance Based Clustering(TICC) is an approach to solve this problem. Thus, in order to cluster data based on its structural similarity, each cluster is defined as a dependency network displaying relationships between the different sensors in a subsequence. The relationships among these sensors are intercluster dependent and intracluster dependent. They are represented using MRF - Markov Random Field. These MRFs provide interpretable insights about the relationships between the sensors to characterize a cluster. By estimating a sparse Gaussian inverse covariance matrix, we gain insights about the clusters. These parameters of clusters are iteratively updated by an algorithm based on the alternating direction method of multipliers.



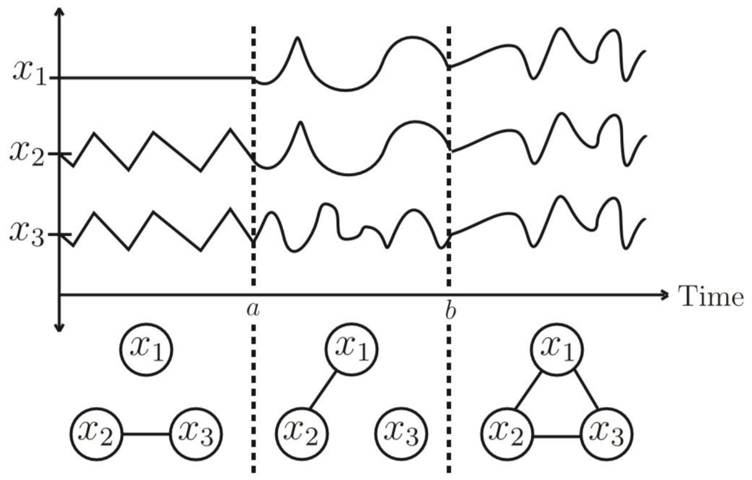
Related Work:

Time-series clustering can be broadly divided into three categories: whole time series clustering, subsequence time series clustering, and time point clustering[2]. This project focuses on the subsequence time series clustering where a set of subsequences are extracted from the time series using a sliding window and these subsequences are clustered. Several methods have been proposed for the subsequence clustering of time series data[4][5][6][7]. However, these methods use distance-based similarity measures that do not yield good results in certain cases[3]. This algorithm uses model-based clustering instead, where the data is assumed to be coming from a series of underlying distributions. This method clusters the subsequences based on the dependency graph of each subsequence. TICC simultaneously segments and clusters time series data using this graph structure which is not the case in other work done in this area.

Background:

<u>Markov Random Field:</u>

Data containing a huge amount of sequences of multivariate time-stamped observation can be modeled as a network of interacting entities, where each entity is a node associated with a time series of data points. These modeled dependency networks are known as Markow random fields(MRFs). In this network, the edge represents a partial correlational or direct effect (while holding other nodes constant.) between the two entities. This correlation evolves over time.



Three sensors with associated time series readings. Based on such data, we infer a time-varying network that reveals 1) the dependencies between the different sensors, and 2)when and how these dependencies change over time.

Forecasting behavior, analyzing trends, detecting anomalies can be easily done using this representation. For example, in an automobile sensor data scenario for stopping at a red light state, we would see a correlation between the brake and accelerator sensor in the same cluster. However, in the case of a state where a car turns, this correlation will be found in the brake sensor at ‘t’ and steering wheel sensor at time ‘t+1’. This would be an intercluster dependency. Thus, with the help of MRFs, we are able to categorize each cluster.

Toeplitz Matrices

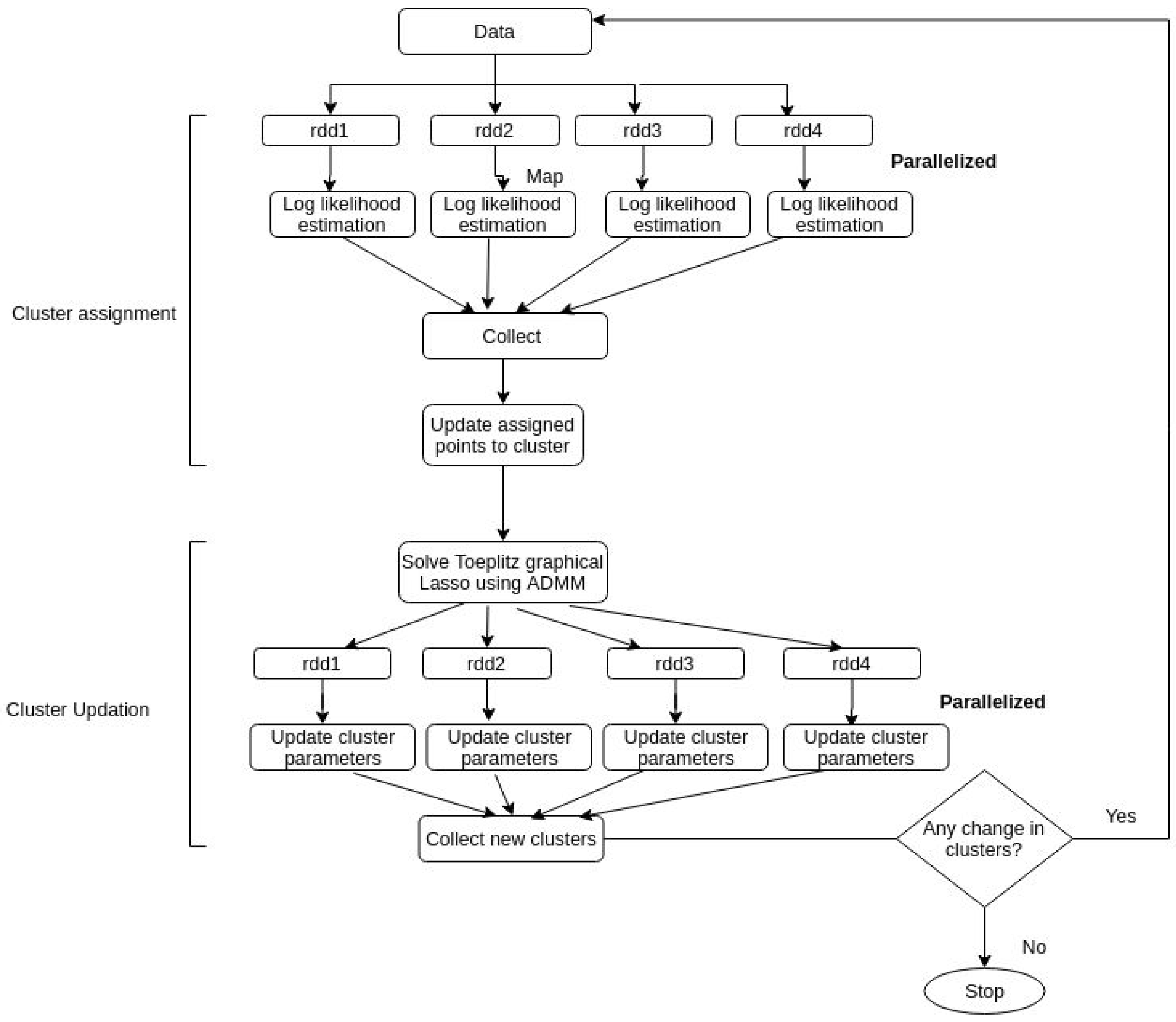
Toeplitz matrix is a matrix in which each diagonal is constant as shown int= the figure. In this matrix Aij =A(i+1)(j+1)=constant.

$$\Theta_i = \begin{bmatrix} A^{(0)} & (A^{(1)})^T & (A^{(2)})^T & \cdots & \cdots & (A^{(w-1)})^T \\ A^{(1)} & A^{(0)} & (A^{(1)})^T & \ddots & & \vdots \\ A^{(2)} & A^{(1)} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & (A^{(1)})^T & (A^{(2)})^T \\ \vdots & & \ddots & A^{(1)} & A^{(0)} & (A^{(1)})^T \\ A^{(w-1)} & \cdots & \cdots & A^{(2)} & A^{(1)} & A^{(0)} \end{bmatrix},$$

This is also called block Toeplitz. The cluster parameters (**θ**) in the algorithm as represented as Toeplitz matrices. $A^{(0)}$ here represents the intra-time partial correlations ie, $A^{(0)}_{ij}$ is the relationship between values of sensor i and j. Off diagonal blocks like $A^{(1)}_{ij}$ refers to how sensor 'i' at time t is correlated to sensor j at time t+1 and so on. The matrix is nw*nw dimension where n is the number of sensors and w is the window size. This matrix is constructed on the assumption of time invariance over this window. This Toeplitz structure has been used to model the time invariance and estimates the cluster parameters just for a window and does not depend on the start or endpoint of a subsequence.

ADMM(Alternating direction method of multipliers)
ADMM is a convex optimization approach for large scale optimization tasks. The optimization problem is split into two subproblems and alternates in optimizing the subproblems through the iterations to obtain a global optimum. This algorithm scales well as it is distributed by splitting the problem and alternating amongst them. The paper uses ADMM to update the inverse covariances or the Toeplitz matrices of the clusters. ADMM converges quickly and hence is a fast method for updating cluster parameters of each cluster at each iteration.

System Diagram

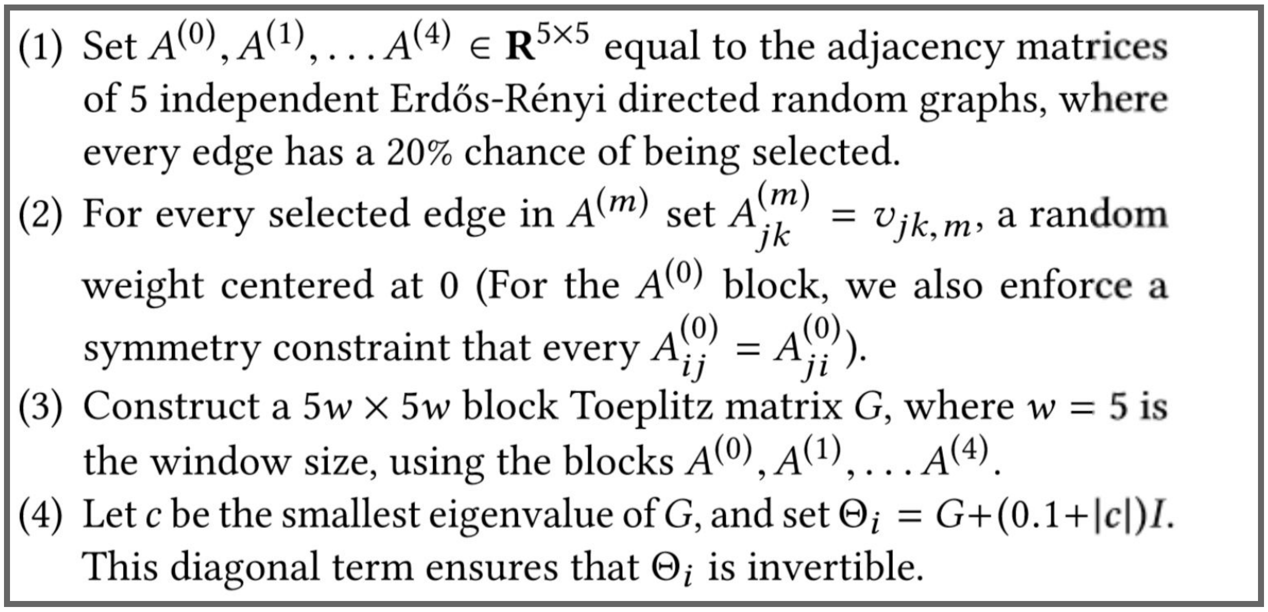

Data
rdd1
rdd2
rdd3
rdd4
Parallelized
Map
Log likelihood estimation
Log likelihood estimation
Log likelihood estimation
Log likelihood estimation
Cluster assignment
Collect
Update assigned points to cluster
Solve Toeplitz graphical Lasso using ADMM
rdd1
rdd2
rdd3
rdd4
Parallelized
Cluster Updation
Update cluster parameters
Update cluster parameters
Update cluster parameters
Update cluster parameters
Collect new clusters
Any change in clusters?
Yes
No
Stop

Modifications

The algorithm uses a variation of expectation maximization to alternate between assigning points to the clusters and then updating cluster parameters. We propose to parallelize this process to speed up the entire system. We have used Apache Spark with a 4 node cluster for parallelizing the application. Since the entire algorithm is not data-parallel, it is not possible to completely parallelize the algorithm using Spark. Hence we have parallelized two components of the algorithm: The log-likelihood estimation and the cluster parameter updating process. Also since the algorithm uses window size, we have to partition the data before passing it to different nodes in the cluster. The log-likelihood estimation for each subsequence is used to assign that subsequence to one of the clusters.

This a combinatorial optimization problem where the assignment is done in such a way that the log-likelihood and temporal consistency are jointly maximized. The calculation of the log-likelihood of each subsequence can be parallelized as they are not dependent on each other. We have parallelized the calculation of these likelihood estimates using RDD in Spark. The points are then assigned to clusters based on the minimum cost path obtained for the T-length subsequence using the Viterbi algorithm. This step cannot be parallelized as assigning points to a cluster is dependent on all likelihood values to calculate the shortest path.
The second step is the maximization step of the EM algorithm. This step updates the inverse covariances of points assigned to each cluster. This inverse covariance matrix is the cluster parameter. This process involves two steps: Solving Toeplitz graphical lasso using ADMM and updating the cluster parameters. The Toeplitz graphical lasso is solved at iteration for each cluster. Since the algorithm may run hundreds of iterations before converging, solving Toeplitz graphical lasso must be solved hundreds of times. To improve the efficiency of this, the algorithm uses ADMM which is a parallel algorithm to solve the optimization problem. These values are then used to update the cluster parameters which are Toeplitz matrices ($\boldsymbol{\theta}$). This step can be parallelized as cluster parameters are not dependent on each other and can be calculated separately. Hence we have parallelized the updating of cluster parameters step again using RDD in Spark and equally pass clusters to different nodes in our Spark cluster for updates.

Datasets:
The TICC algorithm and it's parallelized version is tested on synthetic multivariate data. All the K clusters have a mean of 0 so that the clustering result is based entirely on the structure of the data. For each cluster, a random ground truth Toeplitz inverse covariance is generated as follows,

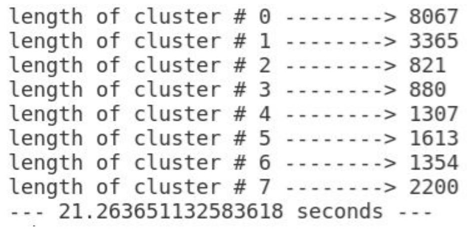


(1) Set $A^{(0)}, A^{(1)}, \dots A^{(4)} \in \mathbf{R}^{5\times5}$ equal to the adjacency matrices of 5 independent Erdős-Rényi directed random graphs, where every edge has a 20% chance of being selected.
(2) For every selected edge in $A^{(m)}$ set $A^{(m)}_{jk} = v_{jk,m}$, a random weight centered at 0 (For the $A^{(0)}$ block, we also enforce a symmetry constraint that every $A^{(0)}_{ij} = A^{(0)}_{ji}$).
(3) Construct a $5w \times 5w$ block Toeplitz matrix G, where $w = 5$ is the window size, using the blocks $A^{(0)}, A^{(1)}, \dots A^{(4)}$.
(4) Let c be the smallest eigenvalue of G, and set $\Theta_i = G + (0.1 + |c|)I$. This diagonal term ensures that Θ_i is invertible.

The dataset has 100K observations in R^5 , where K represents the number of clusters.

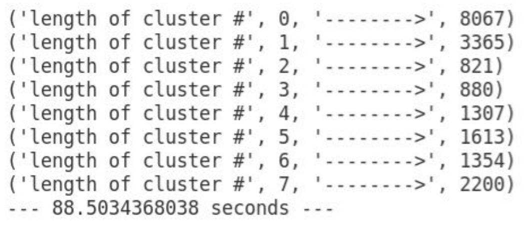
Evaluation measures:
Performance is measured by clustering the data points and comparing them with the actual values. The time required for the algorithms to converge is also measured.

Results:
For the parallelized version of TICC algorithm, we achieved the same accuracy as the original algorithm. However, since the parallelized algorithm was executed on a single machine, the time taken was increased. This was due to the overhead of RDD creation in Spark. The comparison of cluster assignments and execution time of both algorithms can be seen below.



```
length of cluster # 0 --------> 8067
length of cluster # 1 --------> 3365
length of cluster # 2 --------> 821
length of cluster # 3 --------> 880
length of cluster # 4 --------> 1307
length of cluster # 5 --------> 1613
length of cluster # 6 --------> 1354
length of cluster # 7 --------> 2200
--- 21.263651132583618 seconds ---
```

Original code



```
('length of cluster #', 0, '-------->', 8067)
('length of cluster #', 1, '-------->', 3365)
('length of cluster #', 2, '-------->', 821)
('length of cluster #', 3, '-------->', 880)
('length of cluster #', 4, '-------->', 1307)
('length of cluster #', 5, '-------->', 1613)
('length of cluster #', 6, '-------->', 1354)
('length of cluster #', 7, '-------->', 2200)
--- 88.5034368038 seconds ---
```

Distributed code

References

[1] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’17). Association for Computing Machinery, New York, NY, USA, 215–223. DOI:https://doi.org/10.1145/3097983.3098060

[2] S. Zolhavarieh, S. Aghabozorgi, and Y. W. Teh. A review of subsequence time series clustering. The Scientific World Journal, 2014.
[3] E. Keogh, J. Lin and W. Truppel, "Clustering of time series subsequences is meaningless: implications for previous and future research," Third IEEE International Conference on Data Mining, Melbourne, FL, USA, 2003, pp. 115-122, doi: 10.1109/ICDM.2003.1250910.

[4] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD ’12). Association for Computing Machinery, New York, NY, USA, 262–270.

[5] S. Rodpongpun, V. Niennattrakul, and C. A. Ratanamahatana, “Selective subsequence time series clustering,” Knowledge-Based Systems, vol. 35, pp. 361–368, 2012.

[6] Wang, J., Zhang, Y., Zhou, L. et al. CONTOUR: an efficient algorithm for discovering discriminating subsequences. Data Min Knowl Disc 18, 1–29 (2009).
[7] Denton, A.M., Besemann, C.A. & Dorr, D.H. Pattern-based time-series subsequence clustering using radial distribution functions. Knowl Inf Syst 18, 1–27 (2009).