Dynamic Hypergraph Regularized Non-negative Matrix Factorization

Nasr Ullah Khan, Luke Dickens*
University College London
khannasrullah159@gmail.com, l.dickens@ucl.ac.uk

Abstract

Recent advances in dynamic unigraph methods make predictions about links between nodes at the current time, based on previous observations. The most powerful of these approaches are based on regularising over previous graph laplacians with a greater emphasis placed on more recent observations as opposed to older observations. Concurrently, researchers have identified domains in which hypergraph formulations of data provide more detailed information about relationships between entities when those relationships can be multi-factored. This work presents a natural synthesis of these two strands of work, extending regularisation based on dynamic observations to hypergraphs. We present a modelling framework for dynamic hypergraphs, an algorithm for 1-step ahead prediction of the dynamic adjacency matrix, and experiments demonstrating the improved accuracy of this algorithm compared to more conventional approaches.

1 Introduction

Hypergraphs are a powerful way to represent, and reason about, data. Unlike unigraphs (graphs with pairwise links), hypergraphs capture higher-order relationships between nodes, and these representations can lead to more expressive models that capture higher-order correlations [1]. Numerous pre-existing data-sets can be readily represented as hypergraphs [1], and a number of applications already exploit this capability, including: image retrieval [2], person re-identification [3], gene selection [4, 5], disease prediction [6, 7], sub-type identification [8], functional network analysis [9], recommendation systems [10] and link prediction in social networks [11]. The core feature of hypergraphs that support this improved representational capability is the hyperedge, a generalisation of a unigraph edge with arbitrary arity, which captures far more complicated relationships than pairwise relations. We argue in this paper that above applications (and others) utilise data where time information could be better exploited. For instance, time-stamped MRI data might facilitate pre-emptive disease prediction, while the accuracy of person re-identification may benefit from a greater emphasis on more recent images.

Motivating example. Consider a dataset of supermarket baskets each a collection of products bought together. This data can be effectively captured by a hypergraph, where each product is a node, and each basket is a hyperedge. Most simply, a hyperedge is a set of nodes, and here an intersection of hyperedges indicates products common to the corresponding baskets. If we were to reduce this hypergraph to a unigraph of products, such that two products share an edge if they share a basket, we would lose some of this information. For instance, it may be that two baskets have more than two products in common and the proposed unigraph does not capture this[2]. If time information is given

*Equal contribution.

[2]We can alternatively model any hypergraph as a bipartite graph [12]. For this example both products and baskets would then be nodes of different types, with a basket linking to any product it contains. However, this inflates the size of the graph and it is still unclear how we might model or predict higher-order correlations between products.

N. U. Khan et al., L. Dickens et al., Dynamic Hypergraph Regularized Non-negative Matrix Factorization (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.

for baskets, these data can be readily modelled as a dynamic hypergraph, where we partition shopping baskets by time period (e.g. months), each period captured by a hypergraph, in order to capture or analyse how product choices change over time. We argue that the ability to predict characteristics of the hypergraph at the next time-step would have wide applicability in exploiting such dynamic datasets. It is not clear how unigraph based methods would simulaneously capture higher order relationships and this additional layer of time complexity.

Non-negative matrix factorization. Non-negative matrix factorization (NMF) aims to find two non-negative matrices whose product provides a good approximation to some target matrix. The non-negative constraints lead to an efficient, distributed parts-based representation that can aid in the discovery of causal structure within data [13, 14]. Seminal work on Graph regularised NMF [15] uses a regularization term based on the graph laplacian to enforce a NMF factorization that respects the graph structure, namely that two node embeddings are sufficiently close to each other if the corresponding nodes are connected in the graph. This idea is extended in [16] for 1-step ahead prediction for dynamic (time-sequence of) unigraphs based on NMF methods with a recency weighted regularization over previous graph laplacians. A method for hypergraph regularized NMF prediction is presented in [17] to improve manifold learning used in image clustering.

Contribution. Here we present a method for 1-step ahead prediction for dynamic hypergraphs using NMF, and with regularization based on the time-series of hyper-laplacians (hypergraph version of laplacians). Previous works show that incorporating evolving information into feature extraction can greatly enhance dynamic network analysis [16, 18, 19], and we argue that the same is true for dynamic hypergraph analysis. Following ideas developed for ensemble manifold regularization (EMR) [20], we assume that the graph regularization from various time points is located in the convex hull of the previously given manifold candidates; for us, previous hyper-laplacians. Hence, regularization by the time-dependent hyper-laplacians integrates the intrinsic geometrical structure of the data space into the next step prediction, analogous to the dynamic unigraph method in [16].

2 Approach

Our approach operates on a dynamic hypergraph, a sequence of hypergraphs for times $t = 1, \dots T$:

Definition 1. *A* discrete dynamic hypergraph *is time-indexed set* $\mathcal{G}^* = \{\mathcal{G}_1, ..., \mathcal{G}_T\}$ *with* $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t, \mathbf{W}_t)$ *the* hypergraph *at time* t *and* $\mathcal{V}$ *the static ordered* vertex set *(cardinality* N*). The* hyperedge set *at time* t *is denoted* $\mathcal{E}_t = \{e_{i,t}\}_{i=1}^{M_t}$ *(cardinality* M_t*), where an arbitrary* hyperedge *is an ordered tuple of nodes* $e_{i,t} = (\nu_{1t}, ..., \nu_{n_{it}t})$ *i.e. for* $j < j'$, $\nu_{jt} < \nu_{j't}$ *(*$n_{it} \leq N$*). The* hyperedge *weight matrix at time* t *is a diagonal matrix* $\mathbf{W}_t \in \mathbb{R}_+^{M_t \times M_t}$ *with non-zero entries*[3] $[\mathbf{W}_t]_{ii} > 0$.

We extend the concepts of incidence, node degree, edge degree, adjacency and hyper-laplacian matrices from [17] to our time dependent domain in the natural way to give the following definition.

Definition 2. *At time* t*: the* incidence matrix $\mathbf{H}_t \in \{0,1\}^{N \times M_t}$ *denotes node membership of hyperedges, i.e.* $[\mathbf{H}_t]_{ij} = 1$ *iff* $\nu_i \in e_{j,t}$*; the* node degree matrix $\mathbf{D}_{\nu,t} \in \mathbb{R}_+^{N \times N}$ *is a diagonal matrix of node degrees, i.e.* $[\mathbf{D}_{\nu,t}]_{ii} = \sum_k [\mathbf{W}_t]_{kk} [\mathbf{H}_t]_{ik}$*; the* edge degree matrix $\mathbf{D}_{e,t} \in \mathbb{R}_+^{M_t \times M_t}$ *is a diagonal matrix of edge degrees, i.e.* $[\mathbf{D}_{e,t}]_{jj} = \sum_k [\mathbf{H}_t]_{kj}$*; and the adjacency matrix is given by:*

$$\mathbf{A}_t = \mathbf{H}_t \mathbf{W}_t \mathbf{H}_t^T - \mathbf{D}_{\nu,t} \in \mathbb{R}_+^{\mathcal{V} \times \mathcal{V}}$$

The hyper-laplacian at time t *(or laplacian for brevity) is given by*

$$\mathbf{L}_t = \mathbf{D}_{\nu,t} - \mathbf{A}_t = 2\mathbf{D}_{\nu,t} - \mathbf{H}_t \mathbf{W}_t \mathbf{H}_t^T \in \mathbb{R}_+^{\mathcal{V} \times \mathcal{V}}$$

Given $\mathcal{G}^*$, we aim to predict properties of the hypergraph $\mathcal{G}_{T+1}$. This problem can be cast in a few different ways, this paper presents a method to predict $\mathbf{A}_{T+1}$, the adjacency matrix at time $T + 1$.

Approximating Dynamic Hypergraphs. Here we define our dynamic hypergraph regularized NMF (DyHGrNMF) method, which combines aspects of dynamic graph regularized NMF [16] with hypergraph regularized NMF [17], to approximate $\mathbf{A}_{T+1} \approx \mathbf{BF}$, using positive low-rank real matrices $\mathbf{B}, \mathbf{F}^T \in \mathbb{R}_+^{N \times K}$.

[3] We denote the (i, j)th element of matrix $\mathbf{M}$ by $[\mathbf{M}]_{ij}$

Definition 3. *Given a dynamic hypergraph $\mathcal{G}^*$, regularization strength $\lambda > 0$ and feature dimension $K \leq N$, DyHGrNMF seeks to minimize the following objective:*

$$\mathcal{O}^{DyHGrNMF}(\mathbf{B}, \mathbf{F}, \theta) = \|\mathbf{A}_T - \mathbf{BF}\|_F^2 + \lambda \left(\sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T) \right) \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This gives the following optimisation problem:

$$\min_{\{\mathbf{B},\mathbf{F},\theta\}} \{\mathcal{O}^{DyHGrNMF}(\mathbf{B}, \mathbf{F}, \theta)\} \quad \textit{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0; \theta \geq 0$$

The first term in Equation (1) ensures that $\mathbf{BF}$ is a good approximation for $\mathbf{A}_T$, the adjacency matrix for the current time-step, while the second term regularizes this in terms of the hyper-laplacians of all prior hypergraphs $\mathcal{G}_t$, $t \leq T-1$ geometrically weighted according to their recency. As argued in [16], this has the effect of encouraging two nodes $\nu_i, \nu_j \in \mathcal{V}$ to be embedded[4] close to one another, i.e. $\sum_k ([\mathbf{F}]_{ik} - [\mathbf{F}]_{jk})^2$ is small, if they share hyperedges in previous time-steps and with a greater emphasis on more recent graphs.

Update rules. To make the approximation, we begin by an arbitrary initialisation of the basis matrix, $\mathbf{B} \geq 0$ and feature matrix $\mathbf{F} \geq 0$, then iteratively update elements of these with the following update rules, for all $i = 1, \ldots, N$, $j = 1, \ldots, N$ and $k = 1, \ldots, K$ as follows

$$[\mathbf{B}]_{ik} \leftarrow \frac{[\mathbf{A}_T\mathbf{F}^T]_{ik}}{[\mathbf{BFF}^T]_{ik}} \cdot [\mathbf{B}]_{ik} \tag{2}$$

$$[\mathbf{F}]_{kj} \leftarrow \frac{[\mathbf{B}^T\mathbf{A}_T]_{kj}}{\left[\mathbf{B}^T\mathbf{BF} + \lambda \sum_{t=1}^{T-1} \theta^{T-t}\mathbf{FL}_t\right]_{kj}} \cdot [\mathbf{F}]_{kj} \tag{3}$$

A detailed derivation of these update rules can be found in the Appendix. We also define an approach to learn a symmetric approximation named Symmetric DyHGrNMF (SDyHGrNMF) the objective function and update rules for which can also be found in the Appendix.

3 Data-sets and Evaluations

In order to evaluate the 1 step ahead adjacency matrix approximations defined in Section 2, we apply our algorithms to the following data-sets:

1. *Co-authorship*: 20 years of time-stamped co-authorship data from [21].
2. *Email*: 20 years of data from email-Enron dataset (sets of email addresses on emails) [22]
3. *Stocks*: Novel data-set derived from adjusted closing prices of top 9 blue-chip stocks from 16 December 2020 to 16 December 2022 [23], processed to give a dynamic hypergraph. Processing used the time sequence of correlation matrices, $\{\mathbf{\Sigma}_t\}_{t=1}^T$, from adjusted closing prices for the 9 stocks over T time periods, to construct $\{\mathbf{H}_t\}_{t=1}^T$, where $[\mathbf{H}_t]_{ij} = 1$ iff $[\Sigma_t]_{ij} \geq c$ for some $c \in (0, 1)$.

Further details regarding the datasets are in the Appendix. For each dataset, there are $T_{\max}$ time-dependent hypergraphs $\{\mathcal{G}_1, ..., \mathcal{G}_{T_{\max}}\}$, our experiments consider one prediction at time $T_{\max}$ and predict the adjacency matrix the next step ahead, with approximation $\widehat{\mathbf{A}}_{T+1} = \mathbf{BF}$. The evaluation score is the RMSE, i.e. $\|\mathbf{A}_{T+1} - \widehat{\mathbf{A}}_{T+1}\|_F^{0.5}$. This is consistent with that used for dynamic unilink prediction in [16]. Note that the choice to predict the adjacency matrix is motivated by a number of applications in of hypergraph learning that rely on good quality prediction of the adjacency matrix for the target hypergraph, e.g. see [1]. In particular, link prediction and graph clustering can both benefit from a node's features (rows of $\mathbf{F}$) being 'close' to the mean feature vector of hyperedges it belongs to, and this 'closeness' is what the regularization term in Equation (1) controls.

[4] We treat the ith column of $\mathbf{F}$ as the embedding for ν_i.

Baselines. We compare our DyHGrNMF (asymmetric) and SDyHGrNMF (symmetric) algorithms with a selection of baselines. The HGrNMF-T and SHGrNMF-T algorithms are respectively asymmetric and symmetric variants of the GrNMF algorithm in [16], where the regularizer is taken to be $\mathbf{L}_{T-1}$, the laplacian at time $T-1$. Thus, to predict $\mathbf{A}_{T+1}$, DyHGrNMF and SDyHGrNMF respectively seek to minimise the following objective functions:

$$\mathcal{O}^{HGrNMF-T}(\mathbf{B},\mathbf{F},\theta) = \|\mathbf{A}_T - \mathbf{B}\mathbf{F}\|_F^2 + \lambda Tr(\mathbf{F}\mathbf{L}_{T-1}\mathbf{F}^T)$$
$$\mathcal{O}^{SHGrNMF-T}(\mathbf{F},\theta) = \|\mathbf{A}_T - \mathbf{F}^T\mathbf{F}\|_F^2 + \lambda Tr(\mathbf{F}\mathbf{L}_{T-1}\mathbf{F}^T)$$

HGrNMF-T does not incorporate the sequence of laplacians in an auto-regressive manner, only taking a single laplacian from time $T-1$ into account. Thus comparison with DyHGrNMF evaluates the relative benefits of incorporating information from previous timesteps (before $T-1$). A similar comparison can be made between SHGrNMF-T and SDyGrNMF, but for symmetric approximations.

We also include two simpler baselines. The first is a matrix version of *martingale* prediction, which predicts the adjacency matrix at time $T+1$ from the previous adjacency matrix, i.e. $\widehat{\mathbf{A}}_{T+1} = \mathbf{A}_T$ [24]. The second is the *Collapsed Tensor* method from [16], where the next adjacency matrix is predicted as the mean adjacency matrix from previous time-steps, i.e. $\widehat{\mathbf{A}}_{T+1} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{A}_t$.

Table 1: RMSE over time (Score) between actual ($\mathbf{A}_{T+1}$) and predicted ($\widehat{\mathbf{A}}_{T+1}$) adjacency matrices for a variety of datasets and models. Lower scores better. Best scores shown in bold.

Data-set	Model	Score
Co-authorship	DyHGrNMF (ours)	**3.635**
	HGrNMF-T	3.909
	SDyHGrNMF (ours)	18.89
	SHGrNMF-T	18.89
	Martingale	4.272
	Collapsed Tensor	109.9
Email	DyHGrNMF (ours)	**1.144**
	HGrNMF-T	1.150
	SDyHGrNMF (ours)	12.93
	SHGrNMF-T	16.18
	Martingale	1.150
	Collapsed Tensor	5.629
Stocks ($c = 0.3$)	DyHGrNMF (ours)	**0.4092**
	HGrNMF-T	0.5612
	SDyHGrNMF (ours)	2.504
	SHGrNMF-T	2.504
	Collapsed Tensor	20.38
Stocks ($c = 0.4$)	DyHGrNMF (ours)	**2.366**
	HGrNMF-T	3.566
	SDyHGrNMF (ours)	10.94
	SHGrNMF-T	10.94
	Collapsed Tensor	78.57
Stocks ($c = 0.5$)	DyHGrNMF (ours)	**1.423**
	HGrNMF-T	1.553
	SDyHGrNMF (ours)	4.522
	SHGrNMF-T	4.522
	Collapsed Tensor	25.87

Analysis. Table 1 shows RMSE over time for the proposed novel models and baselines. All hyperparameters were optimised by grid-search over feasible values (see appendix for details). For each NMF based method, hyperparameters include: feature dimension K – the number of rows of $\mathbf{F}$, regularisation strength λ, and number of outer iterations of algorithm (max 3). Dynamic hypergraph methods have an additional hyperparameter θ, the geometric weighting parameter. DyHGrNMF outperforms all other models, and is the only model to consistently outperform simple baselines. Symmetric NMF models perform substantially worse than others, including simple baselines.

4 Discussion

This paper presents a new framework for modelling dynamic hypergraphs, and presents 2 new models for approximating adjacency matrices from past observations. Our algorithms, namely DyHGrNMF and SDyHGrNMF, represent asymmetric and symmetric variants of hypergraph regularized non-negative matrix factorization for such dynamic hypergraphs, and models combine the dynamic characteristics of hypergraph models with the ability to exploit higher order relationships in NMF. Evaluations show that our asymmetric model consistently predicts adjacency matrices more accurately than a competitive baseline, HGrNMF-T, that does not integrate the entire history of observations. This shows that the dynamic hypergraph structure as a whole captures useful information about future observations and we argue that this could be effectively exploited in novel algorithms. For future work, we aim to identify good applications to utilise these approximations, and explore how node embeddings can be effectively exploited. Two promising methodological directions include 1) methods to predict missing nodes from partial hyper-edges (extending link prediction beyond binary relationships) and 2) methods to predict incidence matrices – higher order grouping of nodes beyond pairwise association.

References

[1] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2548–2566, 2020. 1, 3

[2] Yuchi Huang, Qingshan Liu, and Dimitris Metaxas.] video object segmentation by hypergraph cut. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1738–1745. IEEE, 2009. 1

[3] Wei Zhao, Shulong Tan, Ziyu Guan, Boxuan Zhang, Maoguo Gong, Zhengwen Cao, and Quan Wang. Learning to map social network users by unified manifold alignment on hypergraph. *IEEE transactions on neural networks and learning systems*, 29(12):5834–5846, 2018. 1

[4] Ze Tian, TaeHyun Hwang, and Rui Kuang. A hypergraph-based learning algorithm for classifying gene expression and arraycgh data with prior knowledge. *Bioinformatics*, 25(21):2831–2838, 2009. 1

[5] Xiao Zheng, Wenyang Zhu, Chang Tang, and Minhui Wang. Gene selection for microarray data classification via adaptive hypergraph embedded dictionary learning. *Gene*, 706:188–200, 2019. 1

[6] Yue Gao, Chong-Yaw Wee, Minjeong Kim, Panteleimon Giannakopoulos, Marie-Louise Montandon, Sven Haller, and Dinggang Shen. Mci identification by joint learning on multiple mri data. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part II 18*, pages 78–85. Springer, 2015. 1

[7] Wei Shao, Yao Peng, Chen Zu, Mingliang Wang, Daoqiang Zhang, Alzheimer's Disease Neuroimaging Initiative, et al. Hypergraph based multi-task feature selection for multimodal classification of alzheimer's disease. *Computerized Medical Imaging and Graphics*, 80:101663, 2020. 1

[8] Emad Ramadan, Sudhir Perincheri, and David Tuck. A hyper-graph approach for analyzing transcriptional networks in breast cancer. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 556–562, 2010. 1

[9] Li Xiao, Junqi Wang, Peyman H Kassani, Yipu Zhang, Yuntong Bai, Julia M Stephen, Tony W Wilson, Vince D Calhoun, and Yu-Ping Wang. Multi-hypergraph learning-based brain functional connectivity analysis in fmri data. *IEEE transactions on medical imaging*, 39(5):1746–1758, 2019. 1

[10] Quan Fang, Jitao Sang, Changsheng Xu, and Yong Rui. Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning. *IEEE Transactions on Multimedia*, 16(3):796–812, 2014. 1

[11] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *The world wide web conference*, pages 2147–2157, 2019. 1

[12] Qionghai Dai and Yue Gao. Mathematical foundations of hypergraph. In *Hypergraph Computation*, pages 19–40. Springer, 2023. 1

[13] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 2

[14] David A Ross and Richard S Zemel. Learning parts-based representations of data. *Journal of Machine Learning Research*, 7(11), 2006. 2

[15] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010. 2

[16] Xiaoke Ma, Penggang Sun, and Yu Wang. Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks. *Physica A: Statistical mechanics and its applications*, 496:121–136, 2018. 2, 3, 4

[17] Kun Zeng, Jun Yu, Cuihua Li, Jane You, and Taisong Jin. Image clustering by hyper-graph regularized non-negative matrix factorization. *Neurocomputing*, 138:209–217, 2014. 2

[18] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012. 2

[19] Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88:1–30, 2015. 2

[20] Bo Geng, Dacheng Tao, Chao Xu, Linjun Yang, and Xian-Sheng Hua. Ensemble manifold regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1227–1233, 2012. 2

[21] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018. 3, 12

[22] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1800683115. 3, 12

[23] Yahoo finance. https://uk.finance.yahoo.com/. 3, 12

[24] Emile Richard, Stéphane Gaïffas, and Nicolas Vayatis. Link prediction in graphs with autoregressive features. *The Journal of Machine Learning Research*, 15(1):565–593, 2014. 4

[25] Da Kuang, Chris Ding, and Haesun Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 106–117. SIAM, 2012. 8

[26] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008. 7

[27] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008. 7, 11

[28] Cornell arb coauth-dblp. https://www.cs.cornell.edu/~arb/data/coauth-DBLP/, . 12

[29] Cornell arb email-enron. https://www.cs.cornell.edu/~arb/data/email-Enron/, . 12

A Appendix

Note 1. *We sometimes define the basis and feature matrices at time T as $\mathbf{B}$ and $\mathbf{F}$ i.e. we remove the time T subscript for brevity and reduced clutter of notation.*

Definition 4. *Sequence of degree matrices for dynamic hyperedges in a dynamic hypergraph: The (time-ordered) sequence of degree matrices for dynamic hyperedges in a dynamic hypergraph can be represented as $\{\mathbf{D}_{e,t}\}_{t=1}^{T}$ where $\mathbf{D}_{e,t} \in \mathbb{R}_{+}^{M_t \times M_t}$ with entries $[\mathbf{D}_{e,t}]_{ij}$.*

The scalar $[\mathbf{D}_{e,t}]_{ij}$, corresponding to the i-th row and the j-th column entry of $\mathbf{D}_{e,t}$, can be written as:

$$[\mathbf{D}_{e,t}]_{ij} = D_{\mathbf{e},t}(e_{i,t}, e_{j,t}) = \begin{cases} \delta_{\mathbf{e},t}(e_{i,t}) & if i = j \\ 0 & if i \neq j \end{cases}, \forall e_{i,t} \in \mathcal{E}_t$$

where $\delta_{\mathbf{e},t}(e_{i,t})$ is the degree for some arbitrary hyperedge $e_{i,t} \in \mathcal{E}_t$, that can be written in terms of column sums of the incidence matrix in the following way:

$\delta_{\mathbf{e},t}(e_{i,t}) = \sum_{k=1}^{|\mathcal{V}|} [\mathbf{H}_t]_{kj} = \sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{\nu_k \in e_{j,t}\}$.

Definition 5. *Objective function and constrained optimisation formulation for SDyHGrNMF (1-step): The loss function to minimise for SDyHGrNMF (1-step) is:*

$\mathcal{O}^{SDyHGrNMF}(\mathbf{F}, \theta) = \|\mathbf{A}_T\|_F^2 + \|\mathbf{F}^T\mathbf{F}\|_F^2 - 2Tr\{\mathbf{A}_T(\mathbf{F}^T\mathbf{F})^T\} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)$
where λ controls the relative importance of the regulariser. Thus, we get the following optimisation problem:

$$\min_{\{\mathbf{F},\theta,\lambda\}} \{\mathcal{O}^{SDyHGrNMF}(\mathbf{F}, \theta)\} \quad s.t. \quad \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

Update rules. To make the approximation for SDyHGrNMF (1-step), we begin by an arbitrary initialisation of the feature matrix $\mathbf{F} \geq 0$, then iteratively update elements of these with the following update rules, for all $i = 1, \ldots, N$, $j = 1, \ldots, N$ and $k = 1, \ldots, K$ as follows

$$[\mathbf{F}]_{kj} \leftarrow \frac{2[\mathbf{F}\mathbf{A}_T]_{kj}}{2[\mathbf{F}\mathbf{F}^T\mathbf{F}]_{kj} + \lambda \sum_{t=1}^{T-1} \theta^{T-t}[\mathbf{F}\mathbf{L}_t]_{kj}} \cdot [\mathbf{F}]_{kj} \tag{4}$$

Using the above update rule to find $\mathbf{F}$, the hyper-links at time $T+1$ (1-step) are predicted as: $\hat{\mathbf{A}_{T+1}} = \mathbf{F}^T\mathbf{F}$.

Algorithm 1: DyHGrNMF (1-step)

function dyhgrnmf_1_step(Inputs) ;

Input : seq_adj_matrices: Sequence of adjacency matrices $\{\mathbf{A}_t\}_{t=1}^{T}$

Input : seq_laplacian_matrices: Sequence of laplacian matrices $\{\mathbf{L}_t\}_{t=1}^{T-1}$

Input : dimensions: The number of components in the initial step of non-hypergraph regularised NMF

Input : _lambda: Explicit regularisation parameter λ

Input : theta: Implicit regularisation parameter θ

Input : K: Number of iterations of the algorithm to perform

Output : Updated basis, feature matrices at time $T-1$ using which we predict $\mathbf{A}_T = \mathbf{F}_{T-1}^T\mathbf{F}_{T-1}$

$\mathbf{B}_{T-1} \leftarrow sklearn.decomposition.NMF(dimensions).fit\_transform(\mathrm{A}_{T-1})$

$\mathbf{F}_{T-1} \leftarrow sklearn.decomposition.NMF(dimensions).components\_$

for _ *in range(K)* **do**

  for *i in range(#{*$B_{ij,T-1}\}_{i=1}^{N}$*)* **do**

    for *j in range(#{*$B_{ij,T-1}\}_{j=1}^{K}$*)* **do**

      $B_{ij,T-1} \leftarrow (\frac{(A_{T-1}F_{T-1}^T)_{ij}}{(B_{T-1}F_{T-1}F_{T-1}^T)_{ij}})(B_{ij,T-1})\}$**end**

    end

    for *i in range(#{*$F_{ij,T-1}\}_{i=1}^{K}$*)* **do**

      for *j in range(#{*$F_{ij,T-1}\}_{j=1}^{N}$*)* **do**

        $(F_{ij,T-1}) \leftarrow (\frac{(B_{T-1}^T A_{T-1})_{ij}}{(B_{T-1}^T B_{T-1}F_{T-1})_{ij} + (\_lambda)\sum_{t=1}^{T-1}(theta)^{T-t}(F_{T-1}L_t)_{ij}})(F_{ij,T-1})$**end**

      end

    end

Derivation of update rules for DyHGrNMF (1-step). In the constrained optimisation problem for DyHGrNMF, the objective function is to be minimised, subject to a certain set of inequality constraints.

So let's attempt to apply the Karush-Kuhn-Tucker (KKT) approach to constrained optimisation in preliminaries, to $\mathcal{O}^{DyHGrNMF}$. First, note that the direct constrained optimisation procedure cannot be applied to $\mathcal{O}^{DyHGrNMF}$, since the Lagrangian for this problem does not allow for direct, closed-form solutions of $\mathbf{B}$ and $\mathbf{F}$. By fixing θ, λ, an iterative 2-step strategy as done in GNMF, GrNMF and HNMF is used. At each iteration, either $\mathbf{B}$ or $\mathbf{F}$ is optimised while the other remains fixed. Iterations are repeated until the algorithm converges or the maximum number of iterations is reached. This is called the Alternating Non-negative Least Squares (ALS) method [26].

As part of the derivation, let's first prove a Lemma that expresses $\mathcal{O}^{DyHGrNMF}$ just in terms of traces of matrices, so that the KKT optimisation procedure can be applied to the optimisation procedure in DyHGrNMF (in particular, first order conditions of the Lagrangian can be more easily evaluated, by using properties of derivatives of traces of matrices [27]).

Lemma 1. The optimisation problem for DyHGrNMF (1-step) can be reformulated as:

$$\min_{\{\mathbf{B},\mathbf{F}\}} \{\mathcal{O}^{DyHGrNMF}(\mathbf{B},\mathbf{F})\} \quad \text{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0$$

$$\mathcal{O}^{DyHGrNMF}(\mathbf{B},\mathbf{F}) = Tr(\mathbf{B}\mathbf{F}\mathbf{F}^T\mathbf{B}^T) - 2Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)$$

Algorithm 2: SDyHGrNMF (1-step)

function sdyhgrnmf_1_step(Inputs) ;

Input : seq_adj_matrices: Sequence of adjacency matrices $\{\mathbf{A}_t\}_{t=1}^{T}$

Input : seq_laplacian_matrices: Sequence of laplacian matrices $\{\mathbf{L}_t\}_{t=1}^{T-1}$

Input : dimensions: The number of components in the initial step of non-hypergraph regularised SNMF

Input : _lambda: Explicit regularisation parameter λ

Input : theta: Implicit regularisation parameter θ

Input : K: Number of iterations of the algorithm to perform

Output : Updated feature matrix at time $T-1$ using which we predict $\widehat{\mathbf{A}}_T = \mathbf{F}_{T-1}^T \mathbf{F}_{T-1}$

$\mathbf{F}_{T-1} \leftarrow SymmNMF(\mathbf{A}_{T-1}, dimensions)$

The SymmNMF function is an implementation of non-hypergraph regularised symmetric NMF from [25].

for _ *in range(K)* **do**

  for *i in range(#{*$F_{ij,T-1}\}_{i=1}^{K}$*)* **do**

    for *j in range(#{*$F_{ij,T-1}\}_{j=1}^{N}$*)* **do**

      $(F_{ij,T-1}) \leftarrow (\frac{2(F_{T-1}A_{T-1})_{ij}}{2(F_{T-1}F_{T-1}^T F_{T-1})_{ij} + \_lambda \sum_{t=1}^{T-1} \theta^{T-t}(F_{T-1}L_t)_{ij}})(F_{T-1})_{ij}$ **end**

    end

  end

Proof of Lemma 1. By the optimisation problem for DyHGrNMF (1-step ahead):

$$\min_{\{\mathbf{B},\mathbf{F},\theta,\lambda\}} \{\mathcal{O}^{DyHGrNMF}(\mathbf{B},\mathbf{F},\theta)\} \quad \text{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

=

$$\min_{\{\mathbf{B},\mathbf{F},\theta,\lambda\}} \{\|\mathbf{A}_T\|_F^2 + \|\mathbf{BF}\|_F^2 - 2Tr\{\mathbf{A}_T(\mathbf{BF})^T\} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

=

$$\min_{\{\mathbf{B},\mathbf{F},\theta,\lambda\}} \{\|\mathbf{A}_T\|_F^2 + \|\mathbf{BF}\|_F^2 - 2Tr\{\mathbf{A}_T(\mathbf{BF})^T\} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0$$

(since θ, λ are assumed to be fixed in the iterative 2-step optimisation procedure)

=

$$\min_{\{\mathbf{B},\mathbf{F},\theta,\lambda\}} \{Tr(\mathbf{A}_T\mathbf{A}_T^T) + Tr(\mathbf{BFF}^T\mathbf{B}^T) - 2Tr\{\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T\} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0$$

(by definition of $\|.\|_F^2$ and $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$)

=

$$\min_{\{\mathbf{B},\mathbf{F},\theta,\lambda\}} \{Tr(\mathbf{BFF}^T\mathbf{B}^T) - 2Tr\{\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T\} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{B} \geq 0; \mathbf{F} \geq 0$$

since $Tr(\mathbf{A}_T\mathbf{A}_T^T)$ is not a function of $\mathbf{B}$ or $\mathbf{F}$, that are the arguments with respect to which the objective function is minimised. (Q.E.D.)

Back to derivation of update rules for DyHGrNMF (1-step). The KKT optimisation procedure can now be applied to the optimisation problem for DyHGrNMF (1-step). The Lagrangian for this problem is:

$\mathcal{L}(\mathbf{B},\mathbf{F},\mathbf{\Psi}_T,\mathbf{\Phi}_T) = Tr(\mathbf{BFF}^T\mathbf{B}^T) - 2Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T) + \lambda\sum_{t=1}^{T-1}\theta^{T-t}Tr(\mathbf{FL}_t\mathbf{F}^T) + Tr(\mathbf{\Phi}_T\mathbf{B}) + Tr(\mathbf{\Psi}_T\mathbf{F})$. Since the trace is only defined for square matrices, hence $\mathbf{\Phi}_T = [\mathbf{\Phi}_T]_{ij} \in \mathbb{R}^{K\times N}$ and $\mathbf{\Psi}_T = [\mathbf{\Psi}_T]_{ij} \in \mathbb{R}^{N\times K}$ are the Lagrange multipliers for the constraints $[\mathbf{B}]_{ij}, [\mathbf{F}]_{ij}$ respectively (at time T).

The partial derivatives of $\mathcal{L}$ with respect to $\mathbf{B}$ and $\mathbf{F}$ are:

$\frac{\partial\mathcal{L}}{\partial\mathbf{B}} = \frac{\partial}{\partial\mathbf{B}}\{Tr(\mathbf{BFF}^T\mathbf{B}^T)\} - 2\frac{\partial}{\partial\mathbf{B}}\{Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T)\} + \frac{\partial}{\partial\mathbf{B}}\{Tr(\mathbf{\Phi}_T\mathbf{B})\}$ (the terms independent of $\mathbf{B}$ are trivially dropped)

Note that: $\frac{\partial}{\partial\mathbf{B}}\{Tr(\mathbf{BFF}^T\mathbf{B}^T)\} = \mathbf{B}(\mathbf{FF}^T)^T + \mathbf{B}(\mathbf{FF}^T) = 2\mathbf{BFF}^T$ since $\frac{\partial}{\partial\mathbf{X}}(Tr(\mathbf{XBX}^T)) = \mathbf{XB}^T + \mathbf{XB}$ for some matrices $\mathbf{B},\mathbf{X}$ and $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

$\frac{\partial}{\partial\mathbf{B}}\{Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T)\} = \mathbf{A}_T\mathbf{F}^T$ since $\frac{\partial}{\partial\mathbf{X}}(Tr(\mathbf{BX}^T)) = \mathbf{B}$

and $\frac{\partial}{\partial\mathbf{B}}\{Tr(\mathbf{\Phi}_T\mathbf{B})\} = \mathbf{\Phi_T}$. Hence,

$\frac{\partial\mathcal{L}}{\partial\mathbf{B}} = 2\mathbf{BFF}^T - 2\mathbf{A}_T\mathbf{F}^T + \mathbf{\Phi_T}$. Now,

$\frac{\partial\mathcal{L}}{\partial\mathbf{F}} = \frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{BFF}^T\mathbf{B}^T)\} - 2\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T)\} + \lambda\sum_{t=1}^{T-1}\theta^{T-t}\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{FL}_t\mathbf{F}^T)\} + \frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{\Psi}_T\mathbf{F})\}$ (the terms independent of $\mathbf{F}$ are trivially dropped)

Note that: $\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{BFF}^T\mathbf{B}^T)\} = \mathbf{B}^T\mathbf{BF} + \mathbf{B}^T\mathbf{BF} = 2\mathbf{B}^T\mathbf{BF}$ (Since $\frac{\partial}{\partial\mathbf{X}}\{Tr(\mathbf{AXBX}^T\mathbf{C})\} = \mathbf{A}^T\mathbf{C}^T\mathbf{XB}^T + \mathbf{CAXB}$. Set $\mathbf{B} = \mathbf{I}_N$ where $\mathbf{I}_N$ is the $(N\times N)$-identity matrix which gives: $\frac{\partial}{\partial\mathbf{X}}\{Tr(\mathbf{AXX}^T\mathbf{C})\} = \mathbf{A}^T\mathbf{C}^T\mathbf{X} + \mathbf{CAX}$)

$\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{B}^T)\} = \frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{A}_T(\mathbf{BF})^T)\} = \frac{\partial}{\partial\mathbf{F}}\{Tr((\mathbf{BF})^T\mathbf{A}_T)\}$ since $Tr(\mathbf{AB}) = Tr(\mathbf{BA})$

$= \frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{F}^T\mathbf{B}^T\mathbf{A}_T)\} = \mathbf{B}^T\mathbf{A}_T$ since $\frac{\partial}{\partial\mathbf{X}}(Tr(\mathbf{X}^T\mathbf{B})) = \mathbf{B}$

$\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{FL}_t\mathbf{F}^T)\} = \mathbf{FL}_t^T + \mathbf{FL}_t$ since $\frac{\partial}{\partial\mathbf{X}}(Tr(\mathbf{XBX}^T)) = \mathbf{XB}^T + \mathbf{XB}$

Recall that the un-normalised dynamic hyper-laplacian is $\mathbf{L}_t = \mathbf{D}_{\nu,t} - \mathbf{A}_t$, so $\mathbf{L}_t^T = (\mathbf{D}_{\nu,t} - \mathbf{A}_t)^T = \mathbf{D}_{\nu,t}^T - \mathbf{A}_t^T = \mathbf{D}_{\nu,t} - \mathbf{A}_t$ since the degree matrix is diagonal hence symmetric and the adjacency matrix is symmetric. Hence the hyper-laplacian is symmetric and so we can write:

$\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{FL}_t\mathbf{F}^T)\} = 2\mathbf{FL}_t$

and $\frac{\partial}{\partial\mathbf{F}}\{Tr(\mathbf{\Psi}_T\mathbf{F})\} = \mathbf{\Psi}_T$

Therefore the partial derivative of $\mathcal{L}$ with respect to feature matrix $\mathbf{F}$ is:

$\frac{\partial\mathcal{L}}{\partial\mathbf{F}} = 2\mathbf{B}^T\mathbf{BF} - 2\mathbf{B}^T\mathbf{A}_T + \lambda\sum_{t=1}^{T-1}\theta^{T-t}2\mathbf{FL}_t + \mathbf{\Psi}_T$

By the Karush-Kuhn-Tucker (KKT) conditions in the KKT optimisation procedure, we have: $[\mathbf{\Phi_T}]_{ij}\cdot[\mathbf{B}]_{ij} = 0$ and $[\mathbf{\Psi_T}]_{ij}\cdot[\mathbf{F}]_{ij} = 0$. Therefore, we can write the 2 partial derivatives evaluated above, in scalar form (the ij-th element of the matrix expression) as (where $i = 1,...,N, j = 1,...,K$ in (1); $i = 1,...,K, j = 1,...,N$ in (2)):

$[\frac{\partial\mathcal{L}}{\partial\mathbf{B}}]_{ij} = 2[\mathbf{BFF}^T]_{ij} - 2[\mathbf{A}_T\mathbf{F}^T]_{ij} + [\mathbf{\Phi}_T]_{ij}$ (1)

$[\frac{\partial\mathcal{L}}{\partial\mathbf{F}}]_{ij} = 2[\mathbf{B}^T\mathbf{BF}]_{ij} - 2[\mathbf{B}^T\mathbf{A}_T]_{ij} + \lambda\sum_{t=1}^{T-1}2\theta^{T-t}[\mathbf{FL}_t]_{ij} + [\mathbf{\Psi}_T]_{ij}$ (2)

where $X_{ijt} = [\mathbf{X}_t]_{ij}$ for some matrix $\mathbf{X}$. As part of the KKT optimisation procedure, the critical points of the Lagrangian are found by setting $\frac{\partial\mathcal{L}}{\partial\mathbf{B}} = \mathbf{0}$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{F}} = \mathbf{0}$. Let's multiply (1) by $[\mathbf{B}]_{ij}$ and (2) by $[\mathbf{F}]_{ij}$, which gives:

$2[\mathbf{BFF}^T]_{ij}\cdot[\mathbf{B}]_{ij} - 2[\mathbf{A}_T\mathbf{F}^T]_{ij}\cdot[\mathbf{B}]_{ij} + [\mathbf{\Phi}_T]_{ij}\cdot[\mathbf{B}]_{ij} = 0$ (1)

$2[\mathbf{B}^T\mathbf{BF}]_{ij}\cdot[\mathbf{F}]_{ij} - 2[\mathbf{B}^T\mathbf{A}_T]_{ij}\cdot[\mathbf{F}]_{ij} + \lambda\sum_{t=1}^{T-1}2\theta^{T-t}[\mathbf{FL}_t]_{ij}\cdot[\mathbf{F}]_{ij} + [\mathbf{\Psi}_T]_{ij}\cdot[\mathbf{F}]_{ij} = 0$ (2)

Setting $[\mathbf{\Phi}_T]_{ij} \cdot [\mathbf{B}]_{ij} = [\mathbf{\Psi}_T]_{ij} \cdot [\mathbf{F}]_{ij} = 0$ by the KKT-conditions, and dividing both sides of (1) and (2) by 2:

$[\mathbf{B}\mathbf{F}\mathbf{F}^T]_{ij} \cdot [\mathbf{B}]_{ij} - [\mathbf{A}_T\mathbf{F}^T]_{ij} \cdot [\mathbf{B}]_{ij} = 0 \;\; (1)$

$[\mathbf{B}^T\mathbf{B}\mathbf{F}]_{ij} \cdot [\mathbf{F}]_{ij} - [\mathbf{B}^T\mathbf{A}_T]_{ij} \cdot [\mathbf{F}]_{ij} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} \cdot [\mathbf{F}]_{ij} = 0 \;\; (2)$

$\Rightarrow [\mathbf{B}\mathbf{F}\mathbf{F}^T]_{ij} \cdot [\mathbf{B}]_{ij} = [\mathbf{A}_T\mathbf{F}^T]_{ij} \cdot [\mathbf{B}]_{ij} \;\; (1)$

$\Rightarrow [\mathbf{B}^T\mathbf{B}\mathbf{F}]_{ij} \cdot [\mathbf{F}]_{ij} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} \cdot [\mathbf{F}]_{ij} = [\mathbf{B}^T\mathbf{A}_T]_{ij} \cdot [\mathbf{F}]_{ij} \;\; (2)$

It can be seen from the first order conditions that it is difficult to obtain an analytic, closed form solution for B_{ijT} and F_{ijT}. Inspired from notation and derivation for GNMF, GrNMF and HNMF, it is easy to see from $(1), (2)$ that the update rules can be written as:

$[\mathbf{B}]_{ij} \leftarrow \frac{[\mathbf{A}_T\mathbf{F}^T]_{ij}}{[\mathbf{B}\mathbf{F}\mathbf{F}^T]_{ij}} \cdot [\mathbf{B}]_{ij} \;\; (1)$

$[\mathbf{F}]_{ij} \leftarrow \frac{[\mathbf{B}^T\mathbf{A}_T]_{ij}}{\left[\mathbf{B}^T\mathbf{B}\mathbf{F} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}\mathbf{L}_t\right]_{ij}} \cdot [\mathbf{F}]_{ij} \;\; (2)$

(Q.E.D.)

Derivation of update rules for SDyHGrNMF (1-step). As part of this derivation, let's prove the following Lemma:

Lemma 2. The (1-step ahead) objective function $\mathcal{O}^{SDyHGrNMF}$ in the relevant optimisation problem can be expressed as:

$\mathcal{O}^{SDyHGrNMF}(\mathbf{F}, \theta) = Tr(\mathbf{F}^T\mathbf{F}\mathbf{F}^T\mathbf{F}) - 2Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{F}) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)$ s.t. $\mathbf{F} \geq 0$.

Proof of Lemma 2. Using the definition for the objective function and constrained optimisation problem formulation for SDyHGrNMF (1-step ahead), the optimisation problem for SDyHGrNMF (1-step ahead) can be written as:

$$\min_{\{\mathbf{F},\theta,\lambda\}} \{\mathcal{O}^{SDyHGrNMF}(\mathbf{F}, \theta)\} \quad \text{s.t.} \quad \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

=

$$\min_{\{\mathbf{F},\theta,\lambda\}} \{\|\mathbf{A}_T\|_F^2 + \|\mathbf{F}^T\mathbf{F}\|_F^2 - 2Tr(\mathbf{A}_T(\mathbf{F}^T\mathbf{F})^T) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

=

$$\min_{\{\mathbf{F},\theta,\lambda\}} \{Tr(\mathbf{A}_T\mathbf{A}_T^T) + Tr((\mathbf{F}^T\mathbf{F})(\mathbf{F}^T\mathbf{F})^T) - 2Tr(\mathbf{A}_T(\mathbf{F}^T\mathbf{F})^T) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

(since $\|\mathbf{A}\|_F^2 = Tr(\mathbf{A}\mathbf{A}^T)$)

=

$$\min_{\{\mathbf{F},\theta,\lambda\}} \{Tr(\mathbf{A}_T\mathbf{A}_T^T) + Tr(\mathbf{F}^T\mathbf{F}\mathbf{F}^T\mathbf{F}) - 2Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{F}) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$
$$\text{s.t.} \quad \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

(since $(\mathbf{A}\mathbf{B})^T = \mathbf{B}^T\mathbf{A}^T$)

=

$$\min_{\{\mathbf{F},\theta,\lambda\}} \{Tr(\mathbf{A}_T\mathbf{A}_T^T) + Tr(\mathbf{F}^T\mathbf{F}\mathbf{F}^T\mathbf{F}) - 2Tr(\mathbf{A}_T\mathbf{F}^T\mathbf{F}) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F}\mathbf{L}_t\mathbf{F}^T)\}$$

$$\text{s.t.} \quad \mathbf{F} \geq 0; \theta \geq 0, \lambda > 0$$

(since $Tr(\mathbf{A}_T \mathbf{A}_T^T)$ is not a function of $\mathbf{F}, \theta$ or λ.

(Q.E.D.)

Back to derivation of update rules for SDyHGrNMF (1-step). Using Lemma 2, we can write up the Lagrangian for this optimisation problem as:

$\mathcal{L}(\mathbf{F}, \mathbf{\Phi}_T) = Tr(\mathbf{A}_T \mathbf{A}_T^T) + Tr(\mathbf{F}^T \mathbf{F} \mathbf{F}^T \mathbf{F}) - 2Tr(\mathbf{A}_T \mathbf{F}^T \mathbf{F}) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} Tr(\mathbf{F} \mathbf{L}_t \mathbf{F}^T) + Tr(\mathbf{\Phi}_T \mathbf{F})$

It is easy to see that this problem now only has 1 first order condition. Let's now find the partial derivative of $\mathcal{L}$ with respect to $\mathbf{F}$:

$\frac{\partial \mathcal{L}}{\partial \mathbf{F}} = \frac{\partial}{\partial \mathbf{F}} \{Tr(\mathbf{F}^T \mathbf{F} \mathbf{F}^T \mathbf{F})\} - 2 \frac{\partial}{\partial \mathbf{F}} \{\mathbf{A}_T \mathbf{F}^T \mathbf{F}\} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \frac{\partial}{\partial \mathbf{F}} \{\mathbf{F} \mathbf{L}_t \mathbf{F}^T\} + \frac{\partial}{\partial \mathbf{F}} \{Tr(\mathbf{\Phi}_T \mathbf{F})\}$

Let's individually evaluate each of the partial derivatives in the expression above. Note that:

$\frac{\partial}{\partial \mathbf{F}} \{Tr(\mathbf{F}^T \mathbf{F} \mathbf{F}^T \mathbf{F})\} = 4\mathbf{F}\mathbf{F}^T\mathbf{F}$. We get this by the following result for the derivative of trace of a higher order matrix [27]:

$\frac{\partial}{\partial \mathbf{X}} \{Tr(\mathbf{B}^T \mathbf{X}^T \mathbf{C} \mathbf{X} \mathbf{X}^T \mathbf{C} \mathbf{X} \mathbf{B})\} = \mathbf{C}\mathbf{X}\mathbf{X}^T\mathbf{C}\mathbf{X}\mathbf{B}\mathbf{B}^T + \mathbf{C}^T\mathbf{X}\mathbf{B}\mathbf{B}^T\mathbf{X}^T\mathbf{C}^T\mathbf{X} + \mathbf{C}\mathbf{X}\mathbf{B}\mathbf{B}^T\mathbf{X}^T\mathbf{C}\mathbf{X} + \mathbf{C}^T\mathbf{X}\mathbf{X}^T\mathbf{C}^T\mathbf{X}\mathbf{B}\mathbf{B}^T$ for arbitrary matrices $\mathbf{B}, \mathbf{C}, \mathbf{X}$ with suitable dimensions such that $\mathbf{B}^T\mathbf{X}^T\mathbf{C}\mathbf{X}\mathbf{X}^T\mathbf{C}\mathbf{X}\mathbf{B}$ is a square matrix.

Applying this identity to the partial derivative at hand, set $\mathbf{B} = \mathbf{I}_N$ and $\mathbf{C} = \mathbf{I}_K$, which gives:

$\frac{\partial}{\partial \mathbf{X}} \{Tr(\mathbf{X}^T \mathbf{X} \mathbf{X}^T \mathbf{X})\} = \mathbf{X}\mathbf{X}^T\mathbf{X} + \mathbf{X}\mathbf{X}^T\mathbf{X} + \mathbf{X}\mathbf{X}^T\mathbf{X} + \mathbf{X}\mathbf{X}^T\mathbf{X} = 4\mathbf{X}\mathbf{X}^T\mathbf{X}$

Going to the next derivatives in $\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$:

$\frac{\partial}{\partial \mathbf{F}} \{\mathbf{A}_T \mathbf{F}^T \mathbf{F}\} = \mathbf{F}\mathbf{A}_T^T + \mathbf{F}\mathbf{A}_T = \mathbf{F}(\mathbf{A}_T^T + \mathbf{A}_T)$ (since $\frac{\partial}{\partial \mathbf{X}}(Tr(\mathbf{B}\mathbf{X}^T\mathbf{X})) = \mathbf{X}\mathbf{B}^T + \mathbf{X}\mathbf{B}$)

$\frac{\partial}{\partial \mathbf{F}} \{Tr(\mathbf{F} \mathbf{L}_t \mathbf{F}^T)\} = \mathbf{F}\mathbf{L}_t^T + \mathbf{F}\mathbf{L}_t = \mathbf{F}(\mathbf{L}_t^T + \mathbf{L}_t)$ (since $\frac{\partial}{\partial \mathbf{X}}(Tr(\mathbf{X}\mathbf{B}\mathbf{X}^T)) = \mathbf{X}\mathbf{B}^T + \mathbf{X}\mathbf{B}$)

$\frac{\partial}{\partial \mathbf{F}} \{Tr(\mathbf{\Phi}_T \mathbf{F})\} = \mathbf{\Phi}_T$

Hence, $\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$ can be written as:

$\frac{\partial \mathcal{L}}{\partial \mathbf{F}} = 4\mathbf{F}\mathbf{F}^T\mathbf{F} - 2\mathbf{F}(\mathbf{A}_T^T + \mathbf{A}_T) + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}(\mathbf{L}_t^T + \mathbf{L}_t) + \mathbf{\Phi}_T$

$= 4\mathbf{F}\mathbf{F}^T\mathbf{F} - 4\mathbf{F}\mathbf{A}_T + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}((\mathbf{D}_{\nu,t} - \mathbf{A}_t)^T + (\mathbf{D}_{\nu,t} - \mathbf{A}_t)) + \mathbf{\Phi}_T$ (by definition of the un-normalised dynamic hyper-laplacian and symmetry of the adjacency matrix)

$= 4\mathbf{F}\mathbf{F}^T\mathbf{F} - 4\mathbf{F}\mathbf{A}_T + 2\lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}(\mathbf{D}_{\nu,t} - \mathbf{A}_t) + \mathbf{\Phi}_T$

$= 4\mathbf{F}\mathbf{F}^T\mathbf{F} - 4\mathbf{F}\mathbf{A}_T + 2\lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}\mathbf{L}_t + \mathbf{\Phi}_T$ (by definition of $\mathbf{L}_t$)

By the KKT-conditions, we have $[\mathbf{\Phi}_T]_{ij} \cdot [\mathbf{F}]_{ij} = [\mathbf{\Phi}_T]_{ij} \cdot [\mathbf{F}]_{ij} = 0 \forall i, j$. Hence, $\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$ can be written in scalar notation as (where $i = 1, ..., K, j = 1, ..., N$):

$(\frac{\partial \mathcal{L}}{\partial \mathbf{F}})_{ij} = 4[\mathbf{F}\mathbf{F}^T\mathbf{F}]_{ij} - 4[\mathbf{F}\mathbf{A}_T]_{ij} + 2\lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} + [\mathbf{\Phi}_T]_{ij}$

As done for the derivation of update rules for DyHGrNMF (1-step), to find the critical point/s of the Lagrangian for solving the constrained optimisation problem, we have:

$\frac{\partial \mathcal{L}}{\partial \mathbf{F}} = \mathbf{0}_{K,N} \Rightarrow (\frac{\partial \mathcal{L}}{\partial \mathbf{F}})_{ij} = 0 \, \forall \, i = 1, ..., K$ and $\forall \, j = 1, ..., N$, since it is easy to see that the R.H.S. of $\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$ is a $(K \times N)$ matrix expression.

$\Rightarrow (\frac{\partial \mathcal{L}}{\partial \mathbf{F}})_{ij} = 4[\mathbf{F}\mathbf{F}^T\mathbf{F}]_{ij} - 4[\mathbf{F}\mathbf{A}_T]_{ij} + 2\lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} + [\mathbf{\Phi}_T]_{ij} = 0.$

Multiplying both sides by $[\mathbf{F}]_{ij}$:

$\Rightarrow 4[\mathbf{F}\mathbf{F}^T\mathbf{F}]_{ij} \cdot [\mathbf{F}]_{ij} - 4[\mathbf{F}\mathbf{A}_T]_{ij} \cdot [\mathbf{F}]_{ij} + 2\lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} \cdot [\mathbf{F}]_{ij} + [\mathbf{\Phi}_T]_{ij} \cdot [\mathbf{F}]_{ij} = 0.$

Applying the KKT-conditions and dividing both sides by 2:

$\Rightarrow 2[\mathbf{F}\mathbf{F}^T\mathbf{F}]_{ij} \cdot [\mathbf{F}]_{ij} - 2[\mathbf{F}\mathbf{A}_T]_{ij} \cdot [\mathbf{F}]_{ij} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} \cdot [\mathbf{F}]_{ij} = 0$

$\Rightarrow 2[\mathbf{F}\mathbf{F}^T\mathbf{F}]_{ij} \cdot [\mathbf{F}]_{ij} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} [\mathbf{F}\mathbf{L}_t]_{ij} \cdot [\mathbf{F}]_{ij} = 2[\mathbf{F}\mathbf{A}_T]_{ij} \cdot [\mathbf{F}]_{ij}$. Taking $[\mathbf{F}]_{ij}$ common from the right on both sides of the equation:

$\Rightarrow 2[\mathbf{F}\mathbf{F}^T\mathbf{F} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}\mathbf{L}_t]_{ij} \cdot [\mathbf{F}]_{ij} = 2[\mathbf{F}\mathbf{A}_T]_{ij} \cdot [\mathbf{F}]_{ij}$.

It can be seen from the first order conditions that it is difficult to obtain an analytic, closed form solution for $[\mathbf{F}]_{ij}$. Inspired from notation and derivation for GNMF, GrNMF, HNMF and DyHGrNMF stated previously, and using the equation for the first order condition above (dividing both sides of it by $2[\mathbf{F}\mathbf{F}^T\mathbf{F} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}\mathbf{L}_t]_{ij}$), the update rule for $[\mathbf{F}]_{ij}$ can be written as:

$[\mathbf{F}]_{ij} \leftarrow \frac{2[\mathbf{F}\mathbf{A}_T]_{ij}}{[2\mathbf{F}\mathbf{F}^T\mathbf{F} + \lambda \sum_{t=1}^{T-1} \theta^{T-t} \mathbf{F}\mathbf{L}_t]_{ij}} \cdot [\mathbf{F}]_{ij}$ for $i = 1, ...K, j = 1, ..., N$.

(Q.E.D.)

B Description of the datasets

Co-authorship. The dataset, used in [21], was downloaded from [28]. As stated in [28], this is a temporal higher-order network dataset, which here means a sequence of timestamped simplices where each simplex is a set of nodes. In this dataset, nodes are authors and a simplex is a publication recorded on DBLP. Timestamps are the year of publication. This dataset has been restricted to simplices that consist of at most 25 nodes. The number of nodes in this dataset are 1,924,991, the number of timestamped simplices are 3,700,067 and the number of unique simplices are 2,599,087. As part of the dynamic hyper-graph construction for this dataset, we only take data for 20 years.

Email. The dataset, used in [22], was downloaded from [29]. As stated in [29], this is a temporal higher-order network dataset, which here means a sequence of timestamped simplices where each simplex is a set of nodes. In email communication, messages can be sent to multiple recipients. In this dataset, nodes are email addresses at Enron and a simplex is comprised of the sender and all recipients of the email. Timestamps are in millisecond resolution and only email addresses from a core set of employees are included in this dataset. The dataset has been restricted to simplices that consist of at most 25 nodes, and this is the version of the dataset used. The number of nodes are 143, the number of timestamped simplices are 10,883 and the number of unique simplices are 1,542. As part of the dynamic hyper-graph construction for this dataset, we only take data for 20 years.

Stocks. The data was downloaded from [23]. The top 9 blue chip stocks from 16 December 2020 to 16 December 2022 were Apple (stock ticker: AAPL), Berkshire Hathaway (stock ticker: BRK-B), Coca-Cola Co (stock ticker: KO), Johnson and Johnson (stock ticker: JNJ), American Express Company (stock ticker: AXP), AbbVie Inc. (stock ticker: ABBV), Nike Inc (stock ticker: NKE), Lockheed Martin Corp (stock ticker: LMT), Honeywell International Inc (stock ticker: HON). There are 50 time periods, and the number of nodes is the number of blue-chip stocks taken into account.

Methodology of dynamic hyper-graph construction for Co-authorship and Email datasets. For each year of data, construct a dictionary of list of lists where each element in the nested list consists of a timestamped (not necessarily unique) simplex. Each key in this dictionary corresponds to a unique year and each value corresponds to a list of lists. We define this dictionary as *_constructed_timeslices*. Using this dictionary, we define a new variable: *num_time_slices* = $\lceil (prop\_timeslice) * (num\_years\_data) \rceil$ where *prop_timeslice* is set to $\frac{1}{100}$ for both datasets and *num_years_data* is the number of years of data taken into account (i.e 20 years as stated previously). Next, we define a new variable: *avg_time_slice* = $\frac{num\_years\_data}{num\_time\_slices}$. Next, we define a new list: *time_slice_boundaries* where the first element is the first year of data to take into account i.e. *start_time*. We update elements in this list in the following way:

for *i* in range(*num_time_slices* + 1), **do**

time_slice_boundaries.append(round(*start_time* + *i*avg_time_slice*))

end for

Finally we make the dynamic hyper-graph, defined as *dyhy_lol* in the following way:

dyhy_lol = []

num_timeslices = len(*time_slice_boundaries*) - 1

for i in range(*num_timeslices*), **do**

dyhy_lol_subset = []

for key in *_constructed_timeslices*.keys(), **do**

if key in range(*time_slice_boundaries[i]*, *time_slice_boundaries[i+1]*), **do**

dyhy_lol_subset.append(*_constructed_timeslices[key]*)

end if

end for

dyhy_lol.append(*dyhy_lol_subset*)

end for

Methodology of dynamic hyper-graph construction for Stocks dataset. Let's write down some pseudo-code used to generate the dyanmic hyper-graph for this dataset. We have defined *T*, which is equivalent to the T in our mathematical notation, to be 50 for this dataset. Define *seq_inc_matrices* as the time-dependent sequence of incidence matrices to construct. Define *stock* to be the data-frame of data loaded for some stock in *stock_list* = [AAPL, BRK-B, KO, JNJ, AXP, ABBV, NKE, LMT, HON]. Here, we assume some Pythonic and Pandas (package in Python) indexing:

seq_inc_matrices = []

dyhy_lol = []

for *t* in $[1, 2, ..50]$, **do**

Step 1: Define the adjusted closing prices across different time periods for each stock

stock_adj_close = *stock*.AdjClose[0:t + 454] **for** *stock* in *stock_list*

Step 2: Concatenate the adjusted closing prices across different time periods to obtain time-variant (450×9) matrices for stock data (the columns dictate the stocks)

Step 3: Compute the returns and correlation matrices for each stock matrix

Step 4: For each correlation matrix corresponding to the constructed stock matrix at time *t*, construct the incidence matrix *inc_matrix_timepoint* (corresponding to $\mathbf{H}_t$) at time *t* where $[\mathbf{H}_t]_{ij} = 1$ iff $[\mathbf{\Sigma}_t]_{ij} \geq c$ for some $c \in (0, 1)$.

seq_inc_matrices.append(inc_matrix_timepoint)

end for

Hyper-parameter optimisation. The feasible values across which we have done grid search to find values giving the smallest score, are: $dimensions = [4, |\mathcal{V}|//2, |\mathcal{V}| - 3]; \lambda, \theta = log\_space(0, 0.2)$ where $log\_space(0, 0.2)$ is an even distribution of values on a log scale from 0 to 0.2.