Deep Augmented Multiview Clustering

Ragunathan Mariappan, Jiang Liu, Vaibhav Rajan
Department of Information Systems and Analytics,
School of Computing, National University of Singapore

1 Introduction

In multiview learning, *views* refer to measurements for the same subjects, that differ in source, datatype or modality. Each matrix, representing a view, has a relationship between two *entity types*, along each matrix dimension, and entity types may be involved in multiple views. See figure 1 for an example. *Collective Matrix Factorization (CMF)* is a general technique to learn shared representations from arbitrary collections of heterogeneous data sources [6]. CMF collectively factorizes the input set of matrices to learn a low-rank latent representation for each entity type from *all* the views in which the entity type is present. It can be used to simultaneously complete one or more matrices in the collection of matrices. Since CMF models arbitrary collections of matrices, this setting is also referred to as *augmented multiview learning* [2].

Given M matrices (indexed by m), $X^{(m)} = [x_{ij}^{(m)}]$, that describe relationships between E entities ($e_1, \dots, e_E$), each with dimension d_e , our model Deep Collective Matrix Factorization (dCMF) [5] jointly obtains latent representations of each entity $U^{(e_i)}$ and low-rank factorizations of each matrix $X^{(m)} \approx U^{(r_m)} \cdot U^{(c_m)^T}$, such that $U^{(e_i)} = f_{\theta}^{e_i}([X]_{e_i})$ where $f^{(\cdot)}$ is an entity-specific non-linear transformation, obtained through a neural network based encoder with weights θ and $[X]_{e_i}$ denotes all matrices in the collection that contains a relationship of entity e_i . The entities corresponding to the rows and columns of the m^{th} matrix are denoted by r_m and c_m respectively. There are two steps in dCMF model construction:

1. Input Transformation: For each entity e_i , we create a new matrix $C^{(i)}$, that we call *concatenated matrix*, by concatenating all the matrices containing entity e_i .
2. Network Construction: We then use E (dependent) autoencoders to obtain the latent factors $U^{(i)}$ from the concatenated matrices $C^{(i)}$. For each entity e_i our network has an autoencoder whose input is $C^{(i)}$, and the decoding is represented by $C^{(i)'}.$ The bottleneck or encoding of each autoencoder, after training, forms the latent factor $U^{(i)}$. The latent factors are learnt by training all the autoencoders together by solving:

$$\underset{\{U^{(e)} \in \mathbb{R}^{d_e \times K}\}_E}{\operatorname{argmin}} \sum_{m=1}^M l_R(X^{(m)}, X^{(m)'}) + \sum_{e=1}^E l_E(C^{(e)}, C^{(e)'}) \quad (1)$$

where l_E is the reconstruction loss between the autoencoder's input $C^{(i)}$ and the decoding $C^{(i)'}$; l_R is the matrix reconstruction loss, where the reconstructed matrix $X^{(m)'} = U^{(r_m)} \cdot U^{(c_m)^T}$ of the view $X^{(m)}$ is obtained by multiplying the associated row and column entity representations $U^{(r_m)}$ and $U^{(c_m)}$.

Collective training of all autoencoders induces dependencies between the autoencoder networks that may result in simultaneous under-fitting in some networks and over-fitting in other networks. We address these optimization challenges through multi-task Bayesian optimization and an acquisition function that is adapted for collective learning of hyperparameters (details in [5]).

1.1 Multi-Way Clustering with dCMF representations

In this preliminary study, we extend the dCMF model to obtain, for an arbitrary collection of input matrices, the latent representations of each entity $U^{(e_i)}$ as well as the following:

- disjoint clusters within each entity type, and
- associations between clusters across entity types.

Let K_{e_i} be the number of clusters for entity type e_i . To simultaneously learn clusters and associations between them, we assume another factorization for each matrix:

$$X^{(m)} \approx I^{(r_m)} \cdot A^{(m)} \cdot I^{(c_m)^T}$$

where $I^{(r_m)}, I^{(c_m)}$ are cluster indicator matrices for the row and column entities respectively, and $A^{(m)}$ is the cluster association matrix, with A_{ij} representing the strength of association between cluster i of the row entity type and cluster j of the column entity type.

The values in $I^{(e_i)}$ are between 0 and 1, i.e., $I^{(e_i)} \in (0, 1)^{d_e \times K_{e_i}}$ and $\sum_{j=1}^{K_{e_i}} I_{ij}^{(e)} = 1$, which can be interpreted as a soft cluster assignment score. Further, collective learning of association matrices for all the input matrices enables us to learn multiple *Cluster Chains* that associate clusters across entities in all the input matrices as shown below.

Illustration. Figure 1 (left) shows an example of 3 input matrices containing dyadic relationships between 4 entities: patients, drugs, known side effects and treated diseases. Matrices $X^{(pr)}$, $X^{(pt)}$ could be extracted from an EMR database and matrix $X^{(sr)}$ could be from a public repository of drugs and their side effects, e.g., SIDER [3]. Figure 1 (center) shows multi-way clusters in each matrix obtained by clustering each entity. The shaded blocks correspond to one cluster chain found by associating the 4th side effect cluster – the 2nd drug cluster – the 1st patient cluster – the 3rd diseases treated cluster. Similarly other cluster chains exist (not shown) that allow us to associate clusters across entity types for an arbitrary collection of input matrices.

Figure 1: Left: 4 entities p : patients, r : drugs, s : side effect diseases, t : treated diseases and 3 relations between the entities, matrices $X^{(pr)}, X^{(sr)}, X^{(pt)}$, Center: An example entity cluster chain. Here the entities p, r, s and t are clustered into 4, 3, 4 and 4 groups respectively. This example captures the cluster chain $CC = \{I_1^p : [I_2^r, I_3^s], I_2^t : I_4^s\}$, Right: dCMF clustering model construction.

1.2 Model Construction

We do not directly tri-factorize the input matrices as done previously in [4, 7]. Instead we use latent representations learnt from dCMF [5] to obtain the cluster indicator and cluster association matrices. This is done by extending the dCMF architecture in two ways:

- **Additional Network Layers.** The autoencoder network for each entity e_i is augmented with a neural network layer for generating $I^{(e_i)}$ as depicted in Fig 1. A softmax activation ensures that the values in $I^{(e_i)}$ are between 0 and 1, and $\sum_{j=1}^{K_{e_i}} I_{ij}^{(e)} = 1$. The representation $I^{(e_i)}$ thus generated is passed to another linear layer with no bias to generate $I^{(e)} \cdot W^{(e)}$.

For a matrix $X^{(m)}$ with row and column entity representations, $U^{(r_m)}, U^{(c_m)}$, we perform another reconstruction through $X^{(m)} \approx (I^{(r_m)} \cdot W^{(r_m)}) \cdot (I^{(c_m)} \cdot W^{(c_m)})^T$. The required tri-factorization can then be obtained by constructing the cluster association matrix as $A^{(m)} = W^{(r_m)} \cdot W^{(c_m)T}$.

- **Loss function.** We add another term (l_C) to the loss function of dCMF as described below. All the latent factors are learnt by collectively training all the autoencoders by solving:

$$\underset{\{U^{(e)} \in \mathbb{R}^{d_e \times K}, I^{(e)} \in (0,1)^{d_e \times K_c}\}_E}{\operatorname{argmin}} \sum_{m=1}^M l_R(X^{(m)}, X^{(m)'}) + \sum_{m=1}^M l_C(X^{(m)}, X_c^{(m)'}) + \sum_{e=1}^E l_E(C^{(e)}, C^{(e)'}) \quad (2)$$

where, in addition to l_E and l_R (from equation 1) we have the matrix reconstruction loss, l_C , where the reconstructed matrix $X_c^{(m)'} = ((I^{(r_m)} \cdot W^{(r_m)}) \cdot (I^{(c_m)} \cdot W^{(c_m)})^T)$ of the view $X^{(m)}$ is obtained from the associated row and column entity cluster indicators $I^{(r_m)}$ and $I^{(c_m)}$.

Note that the cluster association matrix $A^{(m)} = W^{(r_m)} \cdot W^{(c_m)T}$ capturing the association between the clusters are obtained by the layer weights. The autoencoders are trained using SGD and Bayesian optimization for hyperparameter tuning as described in [5].

1.3 Experimental Results

We assume an augmented multiview setup consisting of three matrices (as shown in Fig 1 (left)).

Data. We used the following publicly available data sources:

1. *SIDER*: We construct the matrix $X^{(s,r)}$ using SIDER [3], where the entity types are s : known side-effects and r : drugs. There are 1322 unique drugs with more than 774K listed side effects.
2. *MIMIC-III*: EMR data of patients were obtained from MIMIC-III [1], and used to construct matrices $X^{(p,r)}$ and $X^{(p,t)}$, with entity types p : patients and t : diseases-treated. MIMIC-III contains around 651K diagnoses and 4M drug prescriptions data of more than 40K patients.

There were 1322 unique common drugs and 1042 unique common diseases across MIMIC-III and SIDER. Hence we could construct a dataset of the setting in figure 1 with $p = 39363$, $r = 1322$, $s = 1042$, $t = 1042$. The sparsity level of the matrices are $X^{(p,r)}$: 98.8%, $X^{(s,r)}$: 83.3%, $X^{(p,t)}$: 99.5%.

Baselines. We compare our results against the baselines Data Fusion by Matrix Factorization (DFMF) [7] and Collective Factorization on Related Matrices (CFRM) [4]. DFMF is a tri-factorization based representation learning method – we use the learnt representations to cluster using K-Means. CFRM is a multi-way spectral clustering algorithm that can be used directly.

Evaluation. We create a *test set* of drugs (r_u) that exhibit Adverse Drug Reactions (s_u) that are hidden during training. They are associated with the corresponding affected patients (p_u) and the diseases for which they are treated (t_u). We create 4 test sets by randomly selecting 5% of the drugs (twice) to form the test set of drugs and hide, in each case, different percentages (1% and 4%) of their known side effects. We set the number of clusters to be 4 for each of the entities. Starting from the patient clusters, we form chains by finding the associated clusters in the other entities. The methods are evaluated based on the signal found in the cluster chains for these hidden ADRs. This is measured by the *percentage of hidden entities found* in the test set.

Results. Tables 1 and 2 show the percentage of hidden entities found in cluster chains from DCMF, DFMF and CFRM. For each cluster chain (in a column) the percentage with respect to each of the entities (along a row) is shown. Results for the other two test sets are similar and not shown.

The maximum signal for the hidden entities in our test set is found by one of the chains from DCMF in all our experiments. This suggests that collective learning of representations and cluster structures across the matrices is enabling signal propagation from the diseases found in the EMR to predict side effects that

	DCMF				DFMF				CFRM			
	CC 1	CC 2	CC 3	CC 4	CC 1	CC 2	CC 3	CC 4	CC 1	CC 2	CC 3	CC 4
p_%found	16.66	38.71	22.42	22.22	12.74	59.98	17.42	9.86	26.29	16.39	51.66	5.66
r_%found	30.30	*60.61	0.00	0.00	12.12	24.24	3.03	24.24	6.06	3.03	6.06	3.03
s_%found	0.64	0.64	0.64	0.64	3.85	0.64	5.13	0.64	7.05	0.00	7.05	0.00
t_%found	3.78	3.78	3.57	10.49	2.73	7.45	67.05	15.63	7.45	3.25	1.68	7.45

Table 1: 5% drugs randomly selected, 1% of their side-effects hidden; Bold: best results over all clusters in each clustering method, *: best result over all clusters in all methods, CC: Cluster Chain.

	DCMF				DFMF				CFRM			
	CC 1	CC 2	CC 3	CC 4	CC 1	CC 2	CC 3	CC 4	CC 1	CC 2	CC 3	CC 4
p_%found	17.81	46.99	15.40	19.80	27.39	30.55	34.00	8.07	33.40	26.39	36.35	3.85
r_%found	3.03	9.09	3.03	*63.64	12.12	4.55	4.55	0.00	1.52	4.55	6.06	1.52
s_%found	1.88	3.20	1.88	1.88	3.38	6.20	3.38	3.38	8.08	8.08	8.08	1.32
t_%found	26.25	11.46	4.58	26.25	3.02	6.88	16.46	1.46	4.38	10.10	2.08	2.50

Table 2: 5% drugs randomly selected, 4% of their side-effects hidden; Bold: best results over all clusters in each clustering method, *: best result over all clusters in all methods, CC: Cluster Chain.

are hidden. Additional experiments are ongoing towards a comprehensive evaluation. In the future, we plan to develop methods to identify the cluster chain of interest by comparing the signal within each chain with known (or predicted) associations from various biomedical ontologies. The representation learning model can be extended by modeling temporal correlations in clinical time series through the use of RNN-based variational autoencoders. Finally, the model can be developed to automatically obtain multiple clusterings in a systematic manner.

References

- [1] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.
- [2] Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. Group-sparse embeddings in collective matrix factorization. In *International Conference on Learning Representations (ICLR-14)*, 2014.
- [3] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2015.
- [4] Bo Long, Zhongfei Mark Zhang, Xiaoyun Wu, and Philip S Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning*, pages 585–592. ACM, 2006.
- [5] Ragunathan Mariappan and Vaibhav Rajan. Deep collective matrix factorization for augmented multi-view learning. *Machine Learning*, pages 1–26, 2018.
- [6] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 650–658, 2008.
- [7] Marinka Žitnik and Blaž Zupan. Data fusion by matrix factorization. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):41–53, 2014.

Appendix A Deep Learning based Multiview Representations

In this section we provide a short description of Deep Collective Matrix Factorization (dCMF) for Augmented Multiview Learning, a technique developed in the PI's group at NUS. A complete description can be found in [6]. dCMF is the first deep-learning based method for unsupervised learning of multiple shared representations, that can model non-linear interactions, from an arbitrary collection of matrices.

A.1 Introduction

Pairwise relational data, found in many domains, can be represented as matrices. Matrix completion, that predicts unknown entries in a matrix, is widely used in many applications, e.g. in recommender systems [5], computer vision [3] and bioinformatics [7], to name a few. Often, the matrices are high-dimensional, sparse, and with inherent redundancies. Sufficient information may be present in latent substructures, that can be approximated through low-rank factorizations, and used in predictive models.

When information from multiple heterogeneous sources is available, predictive models benefit from latent representations that model correlated shared structure. In multi-view learning, *views* refer to measurements for the same subjects, that differ in source, datatype or modality. Each matrix, representing a view, has a relationship between two *entity types*, along each matrix dimension, and entity types may be involved in multiple views. For example, in fig. 2(a), entity e_1 could be patients and clinical data from three different sources (notes $X^{(1)}$, images $X^{(2)}$, and diagnoses $X^{(3)}$) may be used to obtain patient representations for modeling risk of diseases. When auxiliary information about multiple entity types are present, they could be effectively utilized to obtain latent representations. For example, in hybrid recommender systems, where side information matrices about users and movies are used in addition to the historical user-rating matrix to obtain user and movie representations (in fig. 2(b), $X^{(1)}$ is the user-rating matrix, $X^{(2)}$ has user-features and $X^{(3)}$ has movie-features). These latent representations are then used to recommend movies to users.

Collective Matrix Factorization (CMF) is a general technique to learn shared representations from arbitrary collections of heterogeneous data sources [8]. CMF collectively factorizes the input set of matrices to learn a low-rank latent representation for each entity type from *all* the views in which the entity type is present. It can be used to simultaneously complete one or more matrices in the collection of matrices. Since CMF models arbitrary collections of matrices, this setting is also referred to as *augmented multi-view learning* [4]. Fig. 2(c) shows an example. Note that the augmented multi-view setting can generalize to any collection of matrices and subsumes the multi-view and recommendation settings.

Figure 2: Examples of (a) **multi-view** setting, (b) **recommendation** setting: 4 entities e_1, e_2, e_3, e_4 and 3 relations between the entities, matrices $X^{(1)}, X^{(2)}, X^{(3)}$; (c) **augmented multi-view** setting: 6 entities $e_1, e_2, e_3, e_4, e_5, e_6$ and 5 relations between the entities, matrices $X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}$.

Classical matrix factorization based approaches assume linearity in the interaction of latent factors which can be restrictive and fails to capture complex non-linear interactions. Modeling such non-linearities through neural models have significantly improved multi-view learning approaches with two views [11] and multiple (but not augmented) views [10]. A common approach is the use of deep autoencoders to obtain shared representations that form latent factors. However these methods cannot generalize to arbitrary collections of matrices, a limitation that we address in our method.

A.2 Background

Matrix Factorization. For a matrix $X \in \mathbb{R}^{m \times n}$, a low-rank factorization aims to obtain latent factors $U^{(1)} \in \mathbb{R}^{m \times K}$, $U^{(2)} \in \mathbb{R}^{n \times K}$, such that $X \approx U^{(1)} \cdot U^{(2)T}$, where the $K < \min(m, n)$ (see fig. 3(a)). The factors are learnt by solving the optimization problem: $\text{argmin}_{U,V} L(X, U^{(1)} \cdot U^{(2)T})$, where L denotes a loss function (e.g., $\|X - U^{(1)} \cdot U^{(2)T}\|_F^2$, where $\|\cdot\|_F$ is the Frobenius norm). Collaborative Filtering, for recommendations, uses such an approach where X is the rating matrix. A common approach to solving this is through a convex relaxation that minimizes the nuclear norm (the sum of singular values) of $U^{(1)} \cdot U^{(2)T}$, which is equivalent to solving: $\min_{X=U^{(1)} \cdot U^{(2)T}} \|U^{(1)}\|_F^2 + \|U^{(2)}\|_F^2$ [9].

CMF aims to jointly obtain low-rank factorizations of M matrices (indexed by m), $X^{(m)} = [x_{ij}^{(m)}]$, that describe relationships between E entities ($e_1, \dots, e_E$), each with dimension d_e . The entities corresponding to the rows and columns of the m^{th} matrix are denoted by r_m and c_m respectively. Fig. 2 shows three examples. Each matrix is approximated by product of low rank- K factors that form the representations of the associated row and column entities: $X^{(m)} \approx U^{(r_m)} U^{(c_m)T}$ where $U^{(e)} = [u_{ik}^{(e)}] \in \mathbb{R}^{d_e \times K}$ is the low-rank matrix for entity type e . Any two matrices sharing the same entity type use the same low-rank representations as part of the approximation, which enables sharing information. For example, in fig. 3(b), the same latent factor $U^{(e_1)}$ is used to reconstruct the three matrices $X^{(1)} \approx U^{(e_1)} U^{(e_2)T}$, $X^{(2)} \approx U^{(e_1)} U^{(e_3)T}$, $X^{(3)} \approx U^{(e_1)} U^{(e_4)T}$. The latent factors are learnt by solving the optimization problem:

$$\text{argmin}_{\{U^{(e)} \in \mathbb{R}^{d_e \times K}\}_E} \sum_{m=1}^M L(X^{(m)}, U^{(r_m)} U^{(c_m)T}) + \sum_{e=1}^E \mathcal{R}(U^{(e)}) \quad (3)$$

where M is the total number of input matrices, E is the total number of entities and $\mathcal{R}$ is a regularizer. For $\mathcal{R}(U^{(e)}) = \lambda \|U^{(e)}\|_F^2$, [1] show that this formulation generalizes the nuclear norm for a single matrix to a *collective nuclear norm* defined on an arbitrary set of matrices (with the reasonable assumption that a pair of entity types do not share more than one view). Although this is a non-convex problem, in practice, solutions obtained through Stochastic Gradient Descent yield good performance [1].

A.3 Deep Collective Matrix Factorization (dCMF)

We design a neural architecture for collective matrix factorization where shared latent representations of each entity are obtained through deep autoencoders. The representations are learnt to enable accurate reconstruction of the input matrices, with the autoencoder reconstruction loss as a regularizer in Eq. 3. Such autoencoder-based regularization has been used in DCCAE [11] for multi-view learning from two views. Our model differs in the way we use autoencoders for learning shared representations. Further, generalizing to augmented multi-view learning leads to non-trivial optimization challenges that we discuss and address in the following sections.

Model construction and training

The inputs to dCMF are (1) M matrices and (2) the relationship between these matrices and the constituent E entities, that can be provided as a bipartite entity-matrix relationship graph $G(V_E, V_M, D)$, where vertices V_E, V_M represent entities and matrices respectively. Edges $(e_i, X^{(m)}), (e_j, X^{(m)}) \in D$ are present if there exists an input matrix $X^{(m)} \in V_M$ capturing the relationship between the entities $e_i, e_j \in V_E$ (see fig. 3(a),(b)). There are two steps in dCMF model construction:

1. **Input Transformation:** For each entity e_i , we create a new matrix $C^{(i)}$, that we call *concatenated matrix*, by concatenating all the matrices containing entity e_i , i.e., all the matrices that are neighbors of e_i in $G(V_E, V_M, D)$. Note that we transform M input matrices to E concatenated matrices, and a single input matrix ($X^{(m)}$) may be used in multiple concatenated matrices ($C^{(i)}$). The concatenation ensures that for each entity, we use the information from all available input matrices to learn its representation.
2. **Network Construction:** We then use E (dependent) autoencoders to obtain the latent factors $U^{(i)}$ from the concatenated matrices $C^{(i)}$. For each entity e_i our network has an autoencoder whose input is

Figure 3: (a) Entity-matrix relationship graph for a single view (b) A collection of views and its entity-matrix relationship graph [square nodes: matrices, circular nodes: entities] (c) dCMF model construction for the example in fig. (b).

$C^{(i)}$, and the decoding is represented by $C^{(i)'}$. The bottleneck or encoding of each autoencoder, after training, forms the latent factor $U^{(i)}$. The latent factors are learnt by training all the autoencoders together by solving:

$$\underset{\{U^{(e)} \in \mathbb{R}^{d_e \times K}\}_E}{\operatorname{argmin}} \sum_{m=1}^M l_R(X^{(m)}, X^{(m)'}) + \sum_{e=1}^E l_E(C^{(e)}, C^{(e)'}) \quad (4)$$

where l_E is the reconstruction loss between the autoencoder's input $C^{(i)}$ and the decoding $C^{(i)'}$; l_R is the matrix reconstruction loss, where the reconstructed matrix $X^{(m)' } = U^{(r_m)} \cdot U^{(c_m)T}$ of the view $X^{(m)}$ is obtained by multiplying the associated row and column entity representations $U^{(r_m)}$ and $U^{(c_m)}$. We call the summations in equation (4) the matrix reconstruction loss ($\mathcal{L}_R$) and autoencoder reconstruction loss ($\mathcal{L}_E$) respectively.

Thus, while CMF factorizes each matrix as $X^{(m)} \approx U^{(r_m)} \cdot U^{(c_m)T}$, dCMF performs non-linear factorization using autoencoders as $X^{(m)} \approx g_{\theta}^{(r_m)}(C^{(r_m)}) \cdot g_{\theta}^{(c_m)}(C^{(c_m)})^T$, where g_{θ} is the encoder corresponding to the entity, with parameter set θ , obtained by collectively minimizing the sum of all the matrix reconstruction and autoencoder reconstruction losses as described above.

Illustration. In fig. 3(a) we show a single matrix $X_{1,2}^{(1)}$ and its two entities e_1 and e_2 . The corresponding

entity-matrix graph below has 2 circular nodes for two entities and 1 square node for the matrix. In fig. 3(b), we show the graph for the collection of 5 matrices and 6 entities (e_1 to e_6) (from fig. 2(c)). Consider, for instance, the entity e_1 . There exists 3 matrices with relationships of entity e_1 with three other entities e_2 , e_3 & e_4 . Hence there are 3 edges from the node representing $e_1 \in V_E$ to the nodes $X^{(1)}, X^{(2)}, X^{(3)} \in V_M$.

We illustrate dCMF model construction in fig. 3(c) for the example from fig. 3(b). We construct $E = 6$ autoencoders, one per entity. The autoencoder construction for entity e_1 is illustrated in the first column of fig. 3(c). We show the subgraph G_{e_1} consisting of 3 edges corresponding to the 3 views $X_{1,2}^{(1)}, X_{1,3}^{(2)}, X_{1,4}^{(3)}$. Hence $C^{(1)} = X_{1,2}^{(1)} \oplus X_{1,3}^{(2)} \oplus X_{1,4}^{(3)}$, where $\oplus$ denotes row or columnwise concatenation of the matrices. To pictorially illustrate this we show a miniature of the setup in fig. 3(b) on top of each column in fig. 3(c), greying out the boxes corresponding to the matrices involved in $C^{(i)}$ construction. We also show $C^{(i)}$ as a block containing concatenated boxes (equal to the number of matrices $C^{(i)}$ is composed of) with a label $C^{(i)}$ below each of the subgraphs which is also the input to the corresponding autoencoder. Similarly we construct the autoencoder for e_2 and the input $C^{(2)}$ by concatenating matrices corresponding to the edges of G_{e_2} as illustrated in the second column of fig. 3(c). Thus we have 6 columns in fig. 3(c) for each of the 6 entities in setup of fig. 3(b). To avoid clutter in fig. 3(c), we show only two examples of the autoencoder reconstruction loss l_E for entities e_1 and e_2 and one example matrix reconstruction loss l_R for the matrix $X^{(1)}$. In total there are $E = 6$ autoencoder reconstruction loss terms and $M = 5$ matrix reconstruction loss terms. Note that this construction can be generalized to any number of entities and matrices.

There are several autoencoder architectures [2] that can be used; we leave that investigation for future work. Here we choose the simplest architecture with multiple fully-connected hidden layers. Notice that the input dimension, which depends on $C^{(e)}$, is different for each autoencoder and the bottleneck layer dimension (the chosen low rank K) is common across all autoencoders. So, the number of layers for each autoencoder is a hyperparameter that is chosen adaptively for each autoencoder.

References

- [1] Guillaume Bouchard, Dawei Yin, and Shengbo Guo. Convex collective matrix factorization. In *Artificial Intelligence and Statistics*, pages 144–152, 2013.
- [2] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [3] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1, 2012.
- [4] Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. Group-sparse embeddings in collective matrix factorization. In *International Conference on Learning Representations (ICLR-14)*, 2014.
- [5] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.
- [6] Ragunathan Mariappan and Vaibhav Rajan. Deep collective matrix factorization for augmented multi-view learning. *Machine Learning*, pages 1–26, 2018.
- [7] Nagarajan Natarajan and Inderjit S Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12):i60–i68, 2014.
- [8] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 650–658, 2008.
- [9] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *International Conference on Computational Learning Theory*, pages 545–560, 2005.
- [10] Qi Wang, Mengying Sun, Liang Zhan, Paul Thompson, Shuiwang Ji, and Jiayu Zhou. Multi-modality disease modeling via collective deep matrix factorization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1155–1164, 2017.
- [11] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.