LEVERAGING MULTI-VIEW NEURAL REPRESENTATION LEARNING METHODS FOR BIOMEDICAL APPLICATIONS

by

AISHWARYA JAYAGOPAL

AN INTERNAL CAPSTONE PROJECT SUBMITTED FOR THE DEGREE OF

MASTER OF COMPUTING

in the

GRADUATE DIVISION

of the

NATIONAL UNIVERSITY OF SINGAPORE

2022

Supervisor:
Assistant Professor Vaibhav Rajan

Examiners:
Assistant Professor Aditya Karanam

Declaration

I hereby declare that this project report is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this report.

This report has also not been submitted for any degree in any university previously.

Aishwarya

Aishwarya Jayagopal

25 April 2022

To my family

Acknowledgments

I would like to thank Professor Vaibhav Rajan for his guidance, support and encouragement throughout the course of this project. Working with Professor Rajan has been a great learning experience, getting to know both the bioinformatics domain as well as the varied ways in which deep learning is used for solving some of the challenges faced by clinicians. I would also like to thank Mr. Ragunathan Mariappan, who has always been a constant support over all the components of this project, especially in understanding the finer implementation details. The brainstorming sessions I had with both of them enabled me to expand my understanding of the subject.

I was fortunate to work with Dr. Lin Jing, from the National University Health System. Working with her helped me understand how the results would be eventually used by clinicians. Her inputs were invaluable to determine if I was proceeding on the right track. Dr. Folefac Aminkeng and Dr. Kee Yuan Ngiam were instrumental in understanding the dataset, especially the clinical aspects.

Professor Prasanna Karthik Vairam was always available to answer any project queries, big or small, and was a constant source of encouragement. I would also like to thank Ms. Geraldine Tang for helping with all the administrative details regarding the project throughout the year. I am grateful to Professor Aditya Karanam for evaluating this project, and his feedback.

This project would not have been possible without the support and encouragement of my family and friends.

Contents

Abstract

Leveraging Multi-View Neural Representation Learning Methods for Biomedical Applications

by

Aishwarya Jayagopal

Master of Computing in

National University of Singapore

In biomedical applications, there are often multiple sources of data for the same entity, that can be utilized to build predictive and exploratory models. For example, a patient can be characterized by medications they are prescribed, their clinical features, genomic characteristics and so on. Unsupervised multi-view representation learning methods like Neural Collective Matrix Factorization (NCMF) and Deep Collective Matrix Tri-Factorization (DCMTF) may be used to learn representations for these biomedical entities by integrating information from varied sources and data types, including auxiliary information. In this project, we evaluated the efficacy of representations from these methods in classification, regression and clustering tasks for multiple biomedical applications. NCMF was applied to the problem of adverse drug event prediction on benchmark datasets. DCMTF was used to mine clinically meaningful phenotypes, by clustering a dataset of cancer patients, obtained from the National University Health System, Singapore. Additionally, we developed a novel NCDIMF architecture to learn domain-invariant representations to solve a problem in cancer drug response prediction.

List of Figures

List of Tables

Chapter 1

Introduction

In the biomedical domain, information comes from varied sources and could be of varied data types. For example, a patient can be characterized by real valued clinical features(like age, height, weight), by categorical clinical features(like gender, smoking habits, genetic mutation information), through image data (like Magnetic Resonance Imaging scans, X-rays) and even through textual inputs from clinical notes. Thus the same patient may be viewed from different perspectives (views). This necessitates methods that can learn to represent these patients(or other biomedical entities) by integrating the information provided by each of these multiple views. Thus multi-view representation learning is of utmost importance in the field of healthcare.

The data from these varied sources can be depicted using heterogeneous networks. The entities under consideration form the nodes of the network, with edges denoting the value associated with an interaction. For example, in a drug - protein interaction network, the nodes could be drugs or proteins with the edges indicating the interactions between them. To obtain a full-fledged understanding of the entities making up the network, an integrated view is necessary [38]. One way to represent these heterogeneous networks is to construct adjacency matrices between each pair of connected entities [81]. These matrices would be used for learning lower dimensional representations of the entities, which would then be used for downstream tasks like classification, regression, clustering etc. Both methods of representation(heterogeneous graph and adjacency matrices) describe only pairwise interactions between entities.

1.1 Problem Statement

In this project, we tackle three main areas of machine learning in the biomedical domain. Each of these three areas have datasets organized in a multi-view setup, where the representations of the biomedical entities need to be learnt jointly.

1.1.1 Adverse Drug Event Prediction

Adverse Drug Events(ADEs) are the unintended side-effects caused by drugs, that are difficult to identify in clinical trials. These could arise due to consumption of multiple drugs simultaneously, due to the interactions between proteins and drugs, the genetic mutations in the patient etc. Thus learning representations for the associated entities become important for predicting if a given drug can cause a potential side-effect. For this task, the ADE prediction is modelled as a classification task, where publicly available datasets are used for training the Neural Collective Matrix Factorization model and learning representations for the entities of interest.

1.1.2 Phenotype Discovery

Identifying groups of similar patients is a common clustering problem. In this case, we obtain genetic mutation information and clinical attributes of patients suffering from Hodgkin's Lymphoma, who may or may not have displayed toxicity effects related to chemotherapy. The task of phenotype discovery is to cluster together patients and all other entities like genes, chromosomes, diseases and genetic variants which are similar, by also incorporating auxiliary data from clinical databases like dbSNP and ClinVar. Through the Deep Collective Matrix Tri Factorization clustering process, both representation learning and clustering of all entities is done.

1.1.3 Drug Response Prediction

Cancer patients react differently to varied drugs, due to the inherent differences in their individual genome. Precision medicine tries to use the genome information of patients to try and predict the response of each patient to various drugs, to improve patient outcome. In this task, we attempt to use 2 distinct domains to learn a domain invariant representation, to allow for generalization to unseen domains. We choose two domains from three available domains - cancer cell lines, patient derived xenograft and The Cancer Genome Atlas patient domains. The two domains follow different underlying distributions. This tackles a common problem in cancer therapy where there is a lack of sufficient clinical datasets, while a differently distributed pre-clinical dataset is often available. We develop a novel NCDIMF architecture to learn domain invariant representations for this task.

1.2 Representation Learning

[4] describes the need to have machine learning algorithms that can learn the underlying explanatory factors for values that are experimentally observed. Understanding this underlying distribution is the crux of representation learning. In terms of probability, a good representation learns the posterior distribution of the hidden variable, given the observed inputs.

Learning representations has many advantages such as dimensionality reduction and the reusability of learnt representations. A good representation, according to [4], is expressive and can represent a large number of possible input configurations with a fewer set of parameters, than the original input vector. These learnt representations can be used for varied downstream tasks like classification, regression, clustering since they are learnt in an unsupervised manner.

Unlike supervised learning methods like classification, where a clear goal is to minimize the misclassification error, representation learning has a different goal due to its unsupervised nature. One paradigm of representation learning involves the use of variational autoencoders, which is described in Chapter 2. The methods used in this project employ these networks to learn representations of biomedical entities.

1.3 Multi View Learning via Matrix Factorization

Technological advances, e.g., high-throughput sequencing and drug screening, have enabled rapid generation of data describing the many interconnected and interacting biomedical entities (e.g., genes, diseases, drugs). Biomedical data is naturally represented in the form of matrices, e.g.,gene expression counts in samples, presence or absence of a diagnosed disease in a patient, and drug response in cell lines. Entries in a matrix capture dyadic relations, typically as measurements from an experiment, between entity instances, along the row and column dimensions. Similarity or interaction matrices, such as protein-protein interactions, capture associations, often across the same entities.

[80] explains matrix factorization as a method of representation learning of the row and column entities making up the matrix, by decomposing the matrix into the product of two lower rank matrix factors. Matrix factorization based techniques (e.g., Principal Components Analysis and Non-negative Matrix Factorization) are used in applications such as identification of mutational signatures in cancer, single cell expression analysis and many more [66].

Previous studies have highlighted the merits of utilizing multiple data sources in revealing underlying biological mechanisms and in predictive modeling, e.g., [12, 77, 23]. Integrating biomedical data poses several challenges [85]. Biological data is often extremely sparse due to experimental measurements being available only on a small subset of exponentially large spaces of possible entity interactions. In many cases, there are a large number of measurements on a limited number of subjects leading to datasets with high dimensions and only few samples. Data is often incomplete, noisy and biased. These conditions necessitate combining direct measurements with indirectly related or auxiliary information. obtain richer entity representations, (or features), which can in turn be used to build predictive models. These representations can then be used to build classification or clustering models depending on the final task.

Numerous methods have been designed to integrate additional information into Matrix factorization based techniques such as those based on Inductive Matrix Completion [37, 36], Graph-regularized Matrix Factorization [82], and Collaborative Matrix Factorization [76]. The typical setting in these methods comprises a central matrix containing known and unknown values of the association of interest (e.g., gene-disease or drug-target) and various ways of incorporating auxiliary data (as features or networks) of the entities of interest (genes and diseases, or drugs and targets) within the modeling framework used. A recent survey can be found in [79].

The need for representation learning methods that can integrate biomedical data from varied sources has been established as a multi-view learning task. In this context, the requirement is to integrate not only data that is directly related to the entities of interest but also those that are indirectly related. For example, to build a model to identify genes that can be targeted by a drug, one may need to not only use direct information about genes and drugs, such as their associations and features, but also indirect information such as drug efficacy on patients and clinical data of patients. In multiview learning, views refer to measurements for the same subjects, that differ in source, datatype or modality. Canonical correlation analysis and matrix factorization have been the basis for many methods, including deep learning models such as DCCAE [74] and CDMF [73]. A recent survey can be found in [49]. These methods cannot model arbitrary collections of matrices.

Collective matrix factorization (CMF) [65] and extensions thereof are models designed to collectively learn from arbitrary collections of matrices. These models generalize the idea of matrix factorization to a collection of matrices. They learn a latent representation for each entity in a way that integrates information from multiple matrices seamlessly. These entity-specific latent representations can be used to perform matrix completion (to predict unknown associations). The latent factors are rich integrative representations that can be used in downstream tasks such as clustering, classification or relation prediction with standard machine learning models, thereby alleviating the resource-intensive task of manual feature engineering from heterogeneous data.

In this project, we describe 3 architectures for representation learning using matrix factorization on a collection of matrices - Neural Collective Matrix Factorization(NCMF), Deep Collective Matrix Tri-Factorization(DCMTF) and Neural Collective Domain Invariant Matrix Factorization(NCDIMF).

In Chapter 2, we discuss the neural building blocks that help create the structure of the architectures used in this project. In Chapter 3, we will explore the details of the adverse drug event prediction problem, the NCMF architecture and the results. Chapter 4 talks about the phenotype discovery task, the details of the dataset and preprocessing, the DCMTF architecture and the results. Chapter 5 details the architecture used for creating domain invariant representations using NCDIMF algorithm, the downstream drug efficacy prediction task and a summary of results obtained.

1.4 Summary of Contributions

- Evaluated the performance of the NCMF architecture for Adverse Drug Event(ADE) prediction on two benchmark datasets, and compared the results against existing state-of-the-art results
- Analyzed clinical and genetic data from patients diagnosed with Hodgkin's Lymphoma, using DCMTF architecture and found clinically meaningful clusters, from the resulting association matrices and cluster labels.
- Developed a model that can use multi-view domains to learn domain-invariant representations, to predict drug responses in cancer patients.

Chapter 2

Literature Survey and Model Architecture

In this chapter, we discuss the building blocks that help in representation learning. We also discuss the existing methods that enable representation learning from matrices, their caveats and the architectures that we use in the remaining sections for performing classification, clustering and regression, as well as their improvements.

2.1 Neural Building Blocks for Representation Learning

2.1.1 Basic Autoencoders

Basic autoencoders ([4]) consist of an encoder-decoder framework with a feed-forward encoder neural network learning the lower dimensional representation of the input and a feed-forward decoder network using the representation to reconstruct the original input. The key idea is to minimize the reconstruction loss J between the input and the reconstructed output, i.e.

$$J = \Sigma_t L(x(t), g_\theta(f_\theta(x(t))))$$

where $x(t)$ represents the input vector, g_θ is the decoder network and f_θ is the encoder network.

2.1.2 Variational Autoencoders

VAE is a generative model that models data Y by generating (lower dimensional) latent variables z, such that there is a high probability to recover the original data Y from z. The distribution $P(z|Y)$ is theoretically the best choice to generate z but is intractable [30]. So, a variational probability $Q(z|Y)$ is used to approximate the posterior by minimizing the Kullback-Leibler (KL) divergence $D[Q(z|Y)||P(z|Y)]$ which is equivalent to maximizing the variational lower bound $\log P(Y) - D[Q(z|Y)||P(z|Y)]$ given by:

$$\mathbf{E}_{z\sim Q}[\log P(Y|z)] - D[Q(z|Y)||P(z|Y)] \tag{2.1}$$

Neural networks are used as probabilistic encoder $Q(z|Y)$ and decoder $P(Y|z)$ and the reparameterization trick [18] is used to obtain samples from $Q(z|Y)$ and ensure differentiability with respect to z. VAEs have been used previously to integrate biomedical data, e.g., by [75, 64], but they do not model arbitrary collections.

2.1.3 DCA

DCA adapts the standard autoencoder architecture (not VAE) to model sparse count data. This is done by defining the reconstruction error as the likelihood of the distribution for a noise model instead of reconstructing the input data itself. The DCA architecture has a neural network as an encoder and decoder wherein the decoder outputs are the parameters of the noise model instead of the input to the encoder. Neural networks capture intrinsic dependencies, allow low-dimensional representation learning and the noise model allows unsupervised learning of noise and signal in the data. [18] recommend the use of the Zero Inflated Negative Binomial (ZINB) distribution for ordinal data.

2.1.4 Zero Inflated Loss

Two types of zero inflated loss are used in the architectures, based on the data type - zero inflated negative binomial distribution(ZINB) for ordinal data and zero inflated normal distribution(ZIN) for real valued data. These distributions enable modelling of sparse matrices, which is often the case in biomedical applications.

2.1.4.1 ZINB Distribution

ZINB is a mixture of 2 components: (i) a point mass at zero representing the zero values and (ii) a negative binomial component for ordinal values. The ZINB distribution is parameterized by the mean (Ω) and dispersion (Θ) of the negative binomial (NB) and the mixture coefficient (Π) that represents weight of the point mass: $\text{NB}(Y;\Omega,\Theta) = \frac{\Gamma(Y+\Theta)}{\Gamma(\Theta)} \cdot \left(\frac{\Theta}{\Theta+\Omega}\right)^{\Theta} \cdot \left(\frac{\Omega}{\Theta+\Omega}\right)^{Y}$ where Γ is the gamma function. $\text{ZINB}(Y;\Pi,\Omega,\Theta) = (\Pi \cdot \Delta_0(Y)) + ((1-\Pi)\cdot \text{NB}(Y;\Omega,\Theta))$ where Δ_0 is the point mass distribution at 0. Thus, ZINB models $Y = 0$ with probability Π and $Y \sim NB$ with probability $1 - \Pi$.

2.1.4.2 ZIN distribution

ZIN is a mixture of 2 components: (i) a point mass at zero representing the zero values and (ii) a Normal component for real values. The ZIN distribution is parameterized by the mean (Ω) and dispersion (Θ) of the Normal (N) and the mixture coefficient (Π) that represents weight of the point mass: $\text{N}(Y_{[e]}^{(m)};\Omega_{[e]}^{(m)},\Theta_{[e]}^{(m)}) = \frac{1}{\sqrt{2\pi\Theta}}\text{exp}\Big(-\frac{1}{2}\frac{(Y-\Omega)^2}{\Theta^2}\Big)$. $\text{ZIN}(Y_{[e]}^{(m)};\Pi_{[e]}^{(m)},\Omega_{[e]}^{(m)},\Theta_{[e]}^{(m)}) = \Big(\Pi \cdot \Delta_0(Y)\Big) + \Big((1-\Pi)\cdot N(Y_{[e]}^{(m)};\Omega_{[e]}^{(m)},\Theta_{[e]}^{(m)})\Big)$ where Δ_0 is the point mass distribution at 0. Thus, ZIN models $Y = 0$ with probability Π and $Y \sim N$ with probability $1 - \Pi$.

2.1.5 CORAL Loss

The CORAL (CORrelation ALignment) loss [67] was introduced as a method to perform unsupervised domain adaptation. It aligns the source and target domains by minimizing the difference in covariance of their input feature distributions. The loss can be mathematically denoted as

$$L_{CORAL} = ||C_S - C_T||_F \tag{2.2}$$

where C_S and C_T are the covariance matrices of the source and target domains respectively. $||.||_F$ is the Frobenius norm of the matrix.

2.2 Multi-View Representation Learning Algorithms

2.2.1 Matrix Factorization Algorithms

CMF is one of the popular matrix factorization algorithms available for a collection of matrices. Several extensions of CMF have been developed over the years such as a formulation that enables group-wise sparsity on the latent factors, gCMF [32], a penalized tri-factorization approach called Data Fusion by Matrix Factorization (DFMF) [84], and a recent neural approach DCMF [44].

2.2.1.1 CMF

For a single $m \times n$ dimensional matrix X, a low-rank factorization aims to obtain latent factors $U^{(r)} \in \mathbb{R}^{m \times l}, U^{(c)} \in \mathbb{R}^{n \times l}$, for its row and column entities respectively, such that $X \approx U^{(r)} \cdot U^{(c)^T}$, where $l < \min(m, n)$. This notion is generalized to arbitrary collections of matrices in Collective Matrix Factorization (CMF) [65]. The m^{th} matrix in the collection is factorized as $X^{(m)} \approx f_m(U^{[r_m]} \cdot U^{[c_m]^T})$ where r_m, c_m represent the entities along its row and column dimensions respectively and f_m is a matrix-specific link function applied element-wise after multiplying the learnt latent factors. Any two matrices sharing the same entity (e.g., the r_m^{th} entity) uses the same low-rank representation ($U^{[r_m]}$) as part of the approximation, which enables sharing information. This is also called augmented multiview learning since it generalizes the multiview setting – auxiliary data from any, even indirectly related, input matrix may be utilized to learn shared representations.

CMF is categorized as an intermediate integration fusion technique which learns joint entity-specific representations from multiple input datasets. These representations, in turn, can be used in downstream modeling. In contrast, in early integration, input matrices are fused into a single matrix, e.g., through projection or concatenation, at the data level before subsequent analysis. In late integration, each matrix is used independently for modeling and predictions from the models are combined, e.g., through majority voting. [85] provides a detailed description.

2.2.1.2 gCMF

To model view-specific noise and shared structure in some of the matrices, a Bayesian formulation with group-wise sparse priors was designed in gCMF [32]. It can model both Gaussian and non-Gaussian observations, including count and binary data. While gCMF models sparsity at the representation level, sparse inputs are not explicitly modeled.

2.2.1.3 DFMF

A penalized tri-factorization approach that models matrix-specific associations across entities and constraints that relate objects of the same entity for arbitrary collections of matrices, called Data Fusion by Matrix Factorization (DFMF), was designed by [84]. Using three factors, i.e., $X^{(m)} \approx U^{[i]} \cdot S_{ij} \cdot U^{[j]^T}$, where $i = r_m, j = c_m$, allows DFMF to model asymmetric relations (e.g., a gene-disease matrix and a different disease-gene matrix) between the same two entities that differ in the third factor (S_{ij}). However, the general case of multiple matrices with the same row and column entities is not modeled. CMF models such inputs through link functions and that can be applied element-wise after multiplying the learnt latent factors: $X^{(m)} \approx f_m(U^{[r_m]} U^{[c_m]^T})$. [40] develop preprocessing techniques which can be used before applying CMF-based methods to handle such cases. where the same row and column entities are in multiple input matrices, that occur frequently in biomedical studies.

2.2.1.4 DCMF

Deep Collective Matrix Factorization (DCMF), a neural approach to CMF, by [44], uses a collection of autoencoders to obtain entity representations from arbitrary collections of matrices. For each entity, all the matrices containing the entity are concatenated (considering transposes wherever required to ensure that the entity remains along the row dimension). DCMF factorizes $X^{(m)} \approx g^{[r_m]}(C^{[r_m]}) \cdot g^{[c_m]}(C^{[c_m]^T})$, where g is the encoder corresponding to the entity and C represents the concatenated matrices containing the corresponding (row r_m or column c_m) entities. The concatenated matrices are passed through autoencoders to obtain lower-dimensional entity-specific representations. The input matrices are modeled as the product of these latent representations. Latent factors are multiplied to reconstruct the original matrices. The entire network is dynamically constructed – the number of autoencoders is equal to the number of entities in the input – and trained collectively by minimizing the sum of autoencoder reconstruction losses and matrix reconstruction losses for all entities and matrices respectively. Although the entity representation learning is non-linear (through g), the matrix reconstruction is a linear function of the representations. Note that in CMF, non-linear interactions may be partly modeled through the link function after the linear inner product. Thus, neither CMF nor DCMF adequately models non-linear interactions.

DCMF is closest to NCMF in terms of overall architecture and the improved modeling flexibility of NCMF is through several architectural innovations described later on. In comparison to DCMF, we explicitly model noise and sparsity through the Zero Inflated layers. Also, DCMF reconstructs each matrix through the (linear) product of the row and column entity representations and so, does not capture more complex non-linear interactions across the entities. NCMF overcomes this limitation by using a neural network. DCMF obtains entity representations from concatenated entity data across matrices, which makes it difficult to model heterogeneity across matrices. In contrast, NCMF can model multiple input datatypes and varying levels of sparsity and noise in the inputs. Through the fusion subnetwork architecture and by obtaining row and column entity representations of all input matrices, NCMF can model collections where the same row and column entities are in multiple matrices, which cannot be modeled in DCMF.

2.2.2 HNE methods

Arbitrary collections of relational data may be represented through Heterogeneous Information Networks (HIN) with multiple types of edges and nodes. Representation learning on HIN aims to obtain vectorial representations of the nodes, called Heterogeneous Network Embeddings (HNE), in a way that preserves the structure and semantics of the HIN. Such embeddings can then be used in downstream tasks such as clustering, node classification or link prediction with standard machine learning models. This approach alleviates the resource-intensive task of manual feature engineering from the graphs and obtains rich integrative representations.

A detailed survey of HNE methods can be found in [78] who categorize HNE methods into 3 groups: (i) Proximity-preserving methods: these are broadly based on random walk approaches (inspired by homogeneous graph embedding methods Deepwalk [51] and LINE [68]). Metapath2vec [15] and HIN2vec [20] use metapath schemes to define semantically relevant walks on HIN from which representations are learnt. Specifying metapaths often require domain knowledge that limits the use of these methods. (ii) Message-passing methods: These are typically based on graph neural networks. Examples include Relational Graph Convolutional Network (R-GCN) [54] and Heterogeneous Graph Transformer (HGT) [25]. (iii) Relation-learning methods: these methods have been developed to learn representations of Knowledge Graphs that are essentially HIN. One of the earliest methods, TransE [5] designs a method that preserves vectorial translational distance among the edge embedding and embeddings of the incident nodes. This idea has been refined and generalized in many subsequent works, e.g., the use of deep networks in ConvE [13].

A large number of HNE methods have been developed in many different domains and for diverse applications. Any HIN can also be represented as a collection of matrices: each edge type represents a relation (a matrix) and each node type is an entity. Matrices readily accommodate multiple datatypes which requires modeling multiple edge attributes (real-valued weights or categorical labels) in HNE methods. Vectorial node features can easily be added as additional matrices in CMF methods which is also not possible in many proximity-preserving and relation-learning HNE methods. Both these are particularly relevant for biomedical data that often contain edge information (such as real-valued significance values or ordinal level of an interaction) and node information (auxiliary features about entities). Proximity-preserving methods typically do not model node and edge attributes. Moreover, Relation-learning methods have mainly been designed for knowledge graphs and also, typically, do not model node and edge attributes.

Among the 3 categories of HNE methods, message-passing methods , usually based on graph neural networks, are more general and most can model node features. They can also model edge attributes – however, in most cases, this would require changes to the algorithm, in most cases. In contrast, in CMF-based methods, this just involves adding another input matrix without any algorithmic changes. Some methods, e.g., R-GCN, can be modified to model weighted edges. However, this often requires changes in their models. Such modifications are less straightforward in proximity-preserving and relation-learning methods.

In contrast, NCMF can model arbitrary number and types of node and edge features – for example, a new edge feature can be added as another input matrix with no change in the algorithm. Node features can easily be incorporated in these methods, but not edge attributes. Hence, a collection of matrices is a more flexible and generic model for biomedical data fusion.

2.3 Neural Collective Matrix Factorization

NCMF[46] is a matrix factorization method, that can learn representations from an arbitrary collection of matrices. This model is described in detail in this section, as it was the model of choice for addressing the ADE prediction problem.

The inputs to NCMF are (i) an arbitrary collection of M matrices containing relational data among N entities and (ii) an entity-matrix bipartite relationship graph $G(\mathcal{E}, \mathcal{X}, \mathcal{Q})$, where the vertices $\mathcal{E} = \{E^{[1]}, \cdots, E^{[N]}\}$ represent entities, $\mathcal{X} = \{X^{(1)}, \cdots, X^{(M)}\}$ represent matrices and edges $(E^{[e]}, X^{(m)}) \in \mathcal{Q}$ show the row and column entities in each matrix. Figure 2.1 shows an example – a collection with $M = 5, N = 6$ and its entity-matrix relationship graph ($E^{[1]}$ is shaded to show the corresponding node). Each entity has multiple instances corresponding to rows or

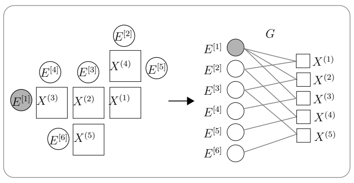

Figure 2.1: Collection with 6 entities, 5 matrices and its entity-matrix relationship graph G.

columns in one or more matrices. We use e to index entities and r_m, c_m represent the row and column entities, respectively, in the m^{th} matrix ($e, r_m, c_m \in \{1, \ldots, N\}$). We use $[e]$, (m) and (e, m) to denote entity $E^{[e]}$, matrix $X^{(m)}$ and edge $(E^{[e]}, X^{(m)})$ respectively. Let $d_{[e]}$ represent the number of instances of the e^{th} entity. NCMF collectively learns entity representations from the inputs and performs matrix completion to obtain:

1. $U^{[e]} \in \mathbb{R}^{d_{[e]} \times l}$: Latent l-dimensional representations of all instances of the e^{th} entity.

2. $X^{(m)'} \in \mathbb{R}^{d_{[r_m]} \times d_{[c_m]}}$: Reconstructed matrix corresponding to each of the matrices $X^{(m)}$.

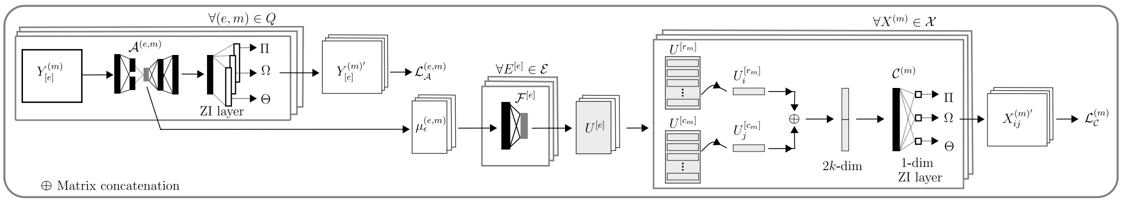

Figure 2.2: Schematic of NCMF Network Construction and Training

2.3.1 Dynamic Network Architecture

To enable learning from arbitrary collections, the NCMF network is dynamically constructed based on the inputs as shown in Figure 2.2. There are 3 subnetworks in our architecture: (i) Autoencoder Subnetwork: $|\mathcal{Q}|$ autoencoders to obtain row-wise and column-wise entity representations in each matrix; (ii) Fusion Subnetwork: $|\mathcal{E}|$ feedforward networks to fuse multiple encodings of entities; (iii) Matrix Completion Subnetwork: $|\mathcal{X}|$ feedforward networks to reconstruct the input matrices. Note that $|\mathcal{Q}| \leq 2M, |\mathcal{E}| \leq N, |\mathcal{X}| = M$.

Autoencoder Subnetwork

This subnetwork, denoted by $\mathcal{A}^{(e,m)}$, consists of autoencoders, one for each entity in each matrix. In the m^{th} matrix, the row vectors are inputs to the row-entity autoencoder and the column vectors are the inputs to the column-entity autoencoder. The same entity may be present in multiple matrices obtained from diverse sources which may differ in datatypes, noise and sparsity levels. So, we model them separately using different autoencoders. The same entity may also be present within a matrix, e.g., in adjacency matrices from graphs. By obtaining row and column entity representations separately we capture correlations across both dimensions.

Let $Y_{[e]}^{(m)}$ denote the e^{th} entity instances (in rows or columns) in the matrix $X^{(m)}$ along its rows. For each autoencoder in $\mathcal{A}^{(e,m)}$, we use the underlying probabilistic framework of VAEs to model complex, high-dimensional data and adapt it for sparse, noisy inputs. We assume a Gaussian latent representation $z_{[e]}^{(m)}$ for e^{th} entity of the m^{th} matrix.

We use a neural network f_ϵ as a probabilistic encoder $Q(z_{[e]}^{(m)}|Y_{[e]}^{(m)})$ and infer its mean and standard deviation. For the probabilistic decoder, $P(Y_{[e]}^{(m)}|z_{[e]}^{(m)})$, we use Zero-Inflated distributions to model noisy, sparse and overdispersed input data. We use a neural network f_δ as a probabilistic decoder to obtain $\bar{Y}_{[e]}^{(m)^*} = f_{\delta^{(e,m)}}(z_{[e]}^{(m)})$. This is used as input simultaneously to the final layer with 3 outputs to learn the 3 parameters of ZINB. The mean and dispersion parameters are always non-negative, so we use an exponential activation function. To ensure that the weight Π remains between zero and one, sigmoid activation is used.

- $\Pi_{[e]}^{(m)} = \text{sigmoid}\left(\bar{Y}_{[e]}^{(m)^*} \cdot W_{\Pi_{[e]}^{(m)}}\right)$
- $\bar{\Omega}_{[e]}^{(m)} = \exp\left(\bar{Y}_{[e]}^{(m)^*} \cdot W_{\Omega_{[e]}^{(m)}}\right)$
- $\Theta_{[e]}^{(m)} = \exp\left(\bar{Y}_{[e]}^{(m)^*} \cdot W_{\Theta_{[e]}^{(m)}}\right)$

The encoder can be viewed as a parameterized link function specific to an entity for each matrix. The latent variable mean $\mu_\epsilon^{(e,m)}$ is considered to be the representation of the e^{th} entity in the m^{th} matrix.

Fusion Subnetwork

For the final matrix reconstruction, we require a single entity representation that is obtained by the fusion subnetwork denoted by $\mathcal{F}^{[e]}$. If the same entity is present in more than one matrix, the final entity-specific representation is obtained by $U^{[e]} = f_{\eta^{[e]}}(\Gamma_m[\mu_\epsilon^{(e,m)}])$, where $\Gamma_m[.]$ represents the concatenation operation and the index m iterates over all matrices containing the e^{th} entity. If the entity is present in a single matrix, no fusion is required and we set $U^{[e]} = \mu_\epsilon^{(e,m)}$.

Matrix Completion Subnetwork

This subnetwork, denoted by $\mathcal{C}^{[m]}$, has one network for each matrix in the input. Each matrix $X^{[m]}$ is reconstructed using its row and column entity representations, $U^{[r_m]}, U^{[c_m]}$. To learn non-linear interactions between them, we use another feed-forward network, f_γ, for the ij^{th} matrix entry: $X_{ij}^{(m)'} = f_\gamma(\Gamma[U_i^{[r_m]}, U_j^{[c_m]}])$ where $\Gamma[.]$ represents the concatenation operator applied on the i^{th} row and j^{th} column entity representations. To model the noise and sparsity in the matrix, we add a final layer to learn three 1-dimensional ZINB parameters (or ZIN, in case of real-valued matrices) for each matrix entry, Weight (Π), Dispersion(Θ) and Mean(Ω). As described earlier, an exponential activation is used for mean and dispersion outputs and a sigmoid activation is used for the weight output. The reconstructed entry is the mean output from the final layer.

2.3.2 Network Training

The loss function to be minimized is $\sum_{(e,m)\in\mathcal{Q}} \mathcal{L}_{\mathcal{A}}^{(e,m)} + \sum_m \mathcal{L}_{\mathcal{C}}^{[m]}$. The first term sums the autoencoder losses in subnetwork $\mathcal{A}^{(e,m)}$ and the second term sums the matrix reconstruction losses in subnetwork $\mathcal{C}^{[m]}$.

Our $\mathcal{L}_{\mathcal{A}}^{(e,m)}$ corresponds to the negative of the variational lower bound given by Eq. 2.1. By assuming a Gaussian $z_{[e]}^{(m)}$, the expression for $D[Q(z_{[e]}^{(m)}|Y_{[e]}^{(m)})||P(z_{[e]}^{(m)}|Y_{[e]}^{(m)})]$ corresponding to Eq. 2.1 follows directly from those in [30]. A Monte Carlo estimate of the expectation in Eq. 2.1 can be obtained through the ZINB likelihood for a minibatch of samples. This is interpretable as the expected negative reconstruction error with the KL divergence term acting as a regularizer. Empirically we find that adding another ridge regularizer on the weight parameter improves performance, as suggested by [18]. Thus, we have $\mathcal{L}_{\mathcal{A}}^{(e,m)} = L_{\mathcal{R}} + L_{\mathrm{KLD}}$,

$$L_{\mathcal{R}} = \mathrm{NLL}_{\mathrm{ZINB}}\Big(Y_{[e]}^{(m)}; \Pi_{[e]}^{(m)}, \Omega_{[e]}^{(m)}, \Theta_{[e]}^{(m)}\Big) + \lambda||\Pi_{[e]}^{(m)}||_F^2,$$
$$L_{\mathrm{KLD}} = -0.5 * \sum(1 + log(\sigma_{\epsilon}^{(e,m)^2}) - \mu_{\epsilon}^{(e,m)^2} - \sigma_{\epsilon}^{(e,m)^2})$$

where $\mathrm{NLL}_{\mathrm{ZINB}}$ is the negative log likelihood of ZINB distribution and L_{KLD} is the KL Divergence term. The matrix reconstruction loss is also calculated using the ZINB likelihood and the ridge regularizer: $\mathcal{L}_{\mathcal{C}}^{[m]} = \mathrm{NLL}_{\mathrm{ZINB}}\Big(X^{(m)}; \Pi^{(m)}, \Omega^{(m)}\Big), \Theta^{(m)}) + \lambda||\Pi^{(m)}||_F^2$. In case of real inputs, ZIN likelihoods are used instead of ZINB.

For each matrix in the collection, we create mini-batches by considering a block within the matrix. A block is a set of row and column entity instances along with the corresponding matrix entries. All the matrix rows and columns that participate in the block are considered in each mini-batch, i.e., if the ij^{th} entry of the matrix is contained in a block then the entire i^{th} row and j^{th} column is considered in the mini-batch. The entire row (column) is used as input in the autoencoder corresponding to the row (column) of the matrix, in the autoencoder subnetwork. Only the entries in the block are reconstructed in the matrix completion subnetwork, using the representations of the corresponding i^{th} row and j^{th} column entity instances.

Algorithm 1: Neural Collective Matrix Factorization (NCMF)

```
Inputs  : Matrices $\mathcal{X} = X^{(1)}, ..., X^{(M)}$, Entity-matrix graph $G(\mathcal{E}, \mathcal{X}, Q)$,
          Entity representation dimension $l$
Outputs : Trained network $\mathcal{N}$ to generate Entity representations
          $\mathcal{U} = U^{[1]}, ..., U^{[N]}$, and Reconstructed Matrices
          $\mathcal{X}' = X^{(1)'}, ..., X^{(M)'}$
   // Dynamic Network Construction
1: Construct network $\mathcal{N}$ with $\mathcal{A}^{(e,m)}$ $\forall(e,m) \in Q$, $\mathcal{F}^{[e]}$ $\forall E^{[e]} \in \mathcal{E}$ and $\mathcal{C}^{(m)}$
   $\forall X^{(m)} \in \mathcal{X}$
   // Network Training
2: repeat
3:     for $m \in \{1, \cdots, M\}$ do
4:         for each mini-batch (block) in $X^{(m)}$ do
5:             Forward pass through subnetworks $\mathcal{A}^{(e,m)}$, $\mathcal{F}^{[e]}$, $\mathcal{C}^{(m)}$
6:             Backpropagate $\mathcal{L} \leftarrow \mathcal{L}_{\mathcal{A}}^{(e,m)} + \mathcal{L}_{\mathcal{C}}^{(m)}$ on sampled entries and
               update weights weights: $\epsilon_*$, $\delta_*$, $\eta_*$, $\gamma_*$
7: until Stopping Criteria Reached;
```

During loss computation, we found that subsampling the zero entries led to improved performance. Within each block, if there are n non-zero entries, we sample nk zero entries and use only these $n(1+k)$ entries for computing $\mathcal{L}_{\mathcal{C}}^{[m]}$. The loss $\mathcal{L}_{\mathcal{A}}^{(e,m)}$ is computed for the rows and columns used in the autoencoders.

We randomly initialize model parameters and optimize the model with AdamW [42]. We use cosine annealing with warm restarts for the learning rate [43] and gradient norm clipped to 1. Training terminates when the difference in loss in consecutive iterations is less than a fixed threshold or if the maximum number of epochs is complete. Algorithm 1 has an outline of NCMF.

2.3.3 Comparison with other Matrix Factorization Algorithms

	CMF	gCMF	DFMF	DCMF	NCMF
Data Sparsity	–	–	–	–	✓
Noise	–	✓	–	–	✓
Non-linear interactions	partial	–	–	–	✓
Matrix Entity Multiplicity	✓	–	partial	–	✓
Multiple Datatypes	✓	✓	–	–	✓

Table 2.1: Modeling capabilities of previous CMF-based methods compared against NCMF.

NCMF has various advantages over existing matrix factorization methods. Previous CMF-based methods have numerous modeling inadequacies that limit their application on biomedical datasets, summarized in Table 2.1. Some of them cannot directly model collections where the same row and column entities are in multiple matrices, a condition we call Matrix Entity Multiplicity. For instance, one may want to collectively use two or more Drug-Disease matrices with different data types, representing different associations (e.g., side effects and treatments). DFMF and DCMF do not model mixed (real and categorical) datatypes and most of the previous techniques do not explicitly model noise in the data. None of the previous techniques explicitly model varying levels of sparsity in the matrices that may arise due to missing values. Biomedical datasets often require modeling all these conditions to obtain effective entity representations from multiple sources.

Another significant modeling limitation of previous methods is in their ability to model complex non-linear interactions. The CMF model proposes the use of a link function that is applied after a linear combination of latent factors. In DCMF, non-linearity is modeled with respect to entity representations, but not with respect to matrix factorization – each reconstruction is a linear function (inner product) of the entity representations. DCMF was found to empirically outperform CMF and gCMF. However, none of them can adequately capture complex non-linear dependencies.

In other domains, e.g., collaborative filtering [25], fully neural methods to generalize matrix factorization have been effective in modeling non-linear interactions. These methods are designed for recommendation systems and do not generalize to arbitrary collections of matrices. To our knowledge, there is no previous fully neural method for Collective Matrix Factorization.

In this project, we use Neural Collective Matrix Factorization (NCMF), that is, to our knowledge, the first fully neural approach to CMF that addresses all the limitations discussed above. NCMF has a novel architecture that is dynamically constructed based on the number of entities and matrices in the input collection. Through the use of Variational Autoencoders (VAE) where the decoded output is modeled using Zero-Inflated distributions, NCMF effectively models sparse and noisy inputs. Multiple such VAEs are used, one for each row and column for each matrix – this architecture effectively models varying datatypes, sparsity and noise levels across the matrices and matrix entity multiplicity. These VAE representations are fused to form entity-specific representations which are then used to reconstruct the matrix using a neural network, thereby modeling non-linearities in both entity representation learning and matrix reconstruction. The entire network can be trained in an end-to-end manner.

NCMF improves over the neural architecture of DCMF and overcomes several limitations of previous methods (summarized in Table 2.1). NCMF explicitly models noise and sparsity in input data through VAEs and Zero Inflated distributions. Through a fully neural architecture, and, in particular, neural layers for generalized matrix factorization, NCMF models non-linearity both at the entity representation and matrix factorization levels. The neural architecture also allows arbitrary number of input matrices (including those with the same row and column entities), and supports multiple input data types.

2.4 Deep Collective Matrix Tri Factorization

The details in this section are heavily derived from the work in [45].

2.4.1 Motivation

In a collection of matrices, denoting varied clinical concepts and their relationships, spectral clustering can be done to obtain clusters amongst all the entities. Deep Collective Matrix Tri Factorization (DCMTF) [45] can be used for multi-way spectral clustering of heterogeneous collections of relational data matrices to discover latent clusters in each input matrix, across both dimensions, as well as the strengths of association across clusters.

Let E be a set of N entities, where the e^{th} entity is denoted by $E_{[e]}$. Data about the entities is available to us through entity instances (observations along row or column dimensions) in a set of M matrices. The m^{th} matrix $X^{(m)}_{r_m,c_m}$ describes the relationships between the entities across the row and column dimensions denoted by r_m, c_m respectively (r_m, c_m ϵ 1, . . . , N). To show the dimensions of the matrix, we use the notation $X^{(m)}_{d_{r_m},d_{c_m}}$.We aim to collectively obtain the matrices:

- $U^{[e]}_{d_e,b}$ - Latent representations of each entity, where b is the common representation dimension.

- $I^{[e]}_{d_e,k_e}$ - Entity cluster indicators for each entity, where k_e is the number of disjoint clusters of entity e.
- $A^{(m)}_{k_{r_m},k_{c_m}}$ - Association matrix corresponding to each of the matrices $X^{(m)}$, containing the strength of association between clusters of the row and column entities.

We use the indices i, j for row and column indices in a matrix and the indices u, v for clusters. With $A^{(m)}$, we can find the pair of clusters that are strongly associated between the entities r_m and c_m. The u^{th} r_m cluster, $u_{r_m} = 1, ..., k_{r_m}$ refers to the u^{th} cluster in the row entity instances of the m^{th} matrix. The strength of association of the u^{th} r_m cluster of the row entities, with the v^{th} c_m cluster of the column entities is given by the element $A^{(m)}_{u_{r_m},v_{c_m}}$. Associations can be found across multiple matrices in the input collection in the form of a cluster chain. The same entity instances (and therefore the same cluster) in a different matrix allows us to "follow the chain" to find another entity cluster that is closely associated.

2.4.2 Existing Methods

2.4.2.1 Spectral Clustering

In spectral clustering, the graph is clustered by treating this process as a graph partition problem, into smaller sub-graphs. The idea is to group similar nodes together in the same sub-graph. This can be treated as a mincut problem, which involves finding k groups such that sum of edge weights across clusters is minimal. This need not work in practice since it can create some clusters with very few nodes. This can be avoided by penalizing smaller clusters (RatioCut algorithm) , but this is NP-hard [71]. A polynomial time relaxation that works well is described below.

Given k partitions, define vigorous cluster indicator vectors as follows J_{iu}, for i = 1, . . . , n and u = 1, . . . , k as follows:

$$J_{iu} = \begin{cases} \sqrt{1/|B_u|} & \text{if } X_i \epsilon B_u \\ 0 & \text{otherwise} \end{cases}$$

Note that although J $\epsilon R_{n,k}$, it still retains interpretability as cluster indicators by capturing the disjoint cluster memberships, and the columns of J are orthonormal, i.e., $JJ^T = I$, where I is the identity matrix. Consider the Graph Laplacian L constructed from the Similarity matrix: L = D S, where D is the n $\times$ n diagonal matrix where the i^{th} row's diagonal element contains the degree $\Sigma_{j=1}^{n} S_{ij}$. It can be shown that minimizing RatioCut is equivalent to solving the following trace minimization problem of $Tr(J^T LJ)$ subject to $JJ^T = I$. This problem is relaxed by allowing the matrix J to take arbitrary real values and denote this relaxed matrix by C with the same minimization problem. The solution to this problem can be obtained by choosing C as the matrix containing the k eigenvectors with the smallest eigenvalues (follows from the Rayleigh-Ritz theorem). To transform this real valued matrix to a binary matrix indicating the partitions, k-Means is used on the rows of C. This is the normalized spectral clustering algorithm of Shi and Malik [62].

2.4.2.2 SpectralNet

[56] allows for a scalable clustering mechanism and has the ability to work on out-of-sample data points. The eigen space of the graph Laplacian matrix is used for creating input embeddings. The clustering process is done in this lower dimensional space using constrained stochastic optimization. This enables scaling to larger dimensions and is done using a last output layer that ensures the output of the layer is orthogonal. This method uses a Siamese network to learn affinities, instead of Gaussian affinities.

2.4.2.3 Spectral Relational Clustering

This method proposed by [41] involves clustering of multiple related matrices, of multiple types. The goal is to simultaneously cluster each of the entities available into clusters. If E_1, ..., E_m are entities making up various matrices and k_1 ,..., k_m denote the number of clusters for each entity, this algorithm attempts to cluster E_1 into k_1 clusters, ..., E_m into k_m clusters. This approach is called Collective Factorization of Related Matrices. The algorithm tries to minimize the Frobenius norm between matrix X_m and its reconstruction from $I_{r_m} A_m I_{c_m}^T$, summed over all matrices. I_e matrix is a cluster indicator matrix for entity e, which must sum to 1 over all available clusters for the entity. A_m is the association matrix that quantifies pairwise association between clusters of various entities. The same relaxation is applied as in RatioCut.

2.4.3 DCMTF Architecture

DCMTF[45] aims to learn both the representations of various entities and to find cluster assignments for each entity. It uses a variational autoencoder to learn representations for the rows and columns for all input matrices. The encoder uses a Gaussian distribution for real values matrices and a Bernoulli distribution for binary valued matrices. Another fusion layer is present, which combines the representations learnt for the same entity from different matrices. This layer concatenates the representation learnt from each auto encoder for this entity and runs it by feed forward network. This network is similar to the fusion subnetwork in NCMF. If the entity exists only in one matrix, the learnt representations are used directly.

To perform multi-way clustering, a similarity matrix S is constructed using a Gaussian kernel. Like in spectral clustering, a Laplacian matrix is obtained and used in the trace minimization problem as before, with a vigorous cluster indicator matrix J. The same relaxation as the one in RatioCut is applied to relax the problem further, replacing J with C. This C matrix is learnt through a neural network, to which the learnt embeddings of each entity is passed. These embeddings are the result of the previous fusion layer. The architecture is shown in Figure 2.3.

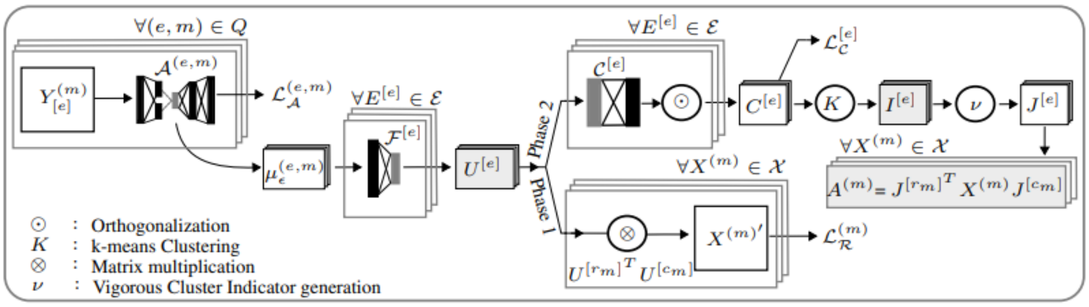
$\forall(e,m) \in Q$
$Y^{(m)}_{[e]}$
$\mathcal{A}^{(e,m)}$
$\mathcal{L}^{(e,m)}_{\mathcal{A}}$
$\forall E^{[e]} \in \mathcal{E}$
$\mu^{(e,m)}_{e}$
$\mathcal{F}^{[e]}$
$U^{[e]}$
Phase 2
Phase 1
$\forall E^{[e]} \in \mathcal{E}$
$\mathcal{C}^{[e]}$
$C^{[e]}$
$\mathcal{L}^{[e]}_{\mathcal{C}}$
K
$I^{[e]}$
ν
$J^{[e]}$
$\forall X^{(m)} \in \mathcal{X}$
$A^{(m)} = J^{[r_m]^T} X^{(m)} J^{[c_m]}$
$\forall X^{(m)} \in \mathcal{X}$
$U^{[r_m]^T} U^{[c_m]}$
$X^{(m)'}$
$\mathcal{L}^{(m)}_{\mathcal{R}}$
$\odot$: Orthogonalization
K : k-means Clustering
$\otimes$: Matrix multiplication
ν : Vigorous Cluster Indicator generation

Figure 2.3: Schematic of DCMTF Network Architecture, reproduced with permission from [45].

2.4.3.1 Dynamic Network Architecture

The architecture consists of three layers.

- VAE network - autoencoders are created for each row and column entity of each matrix. This network is denoted by $A^{(e,m)}$.
- Fusion network - this is used to concatenate entities present in multiple matrices and create uniform dimension embeddings for all entities. This is denoted by $F^{[e]}$
- A feedforward network is used to learn the orthogonal cluster indicator matrix, denoted by $C^{[e]}$. The last layer in this network ensures the output is orthogonal.

The loss being minimised is the sum of autoencoder reconstruction loss, tri factorization trace minimization loss and the matrix reconstruction loss.

$$L = \Sigma_{e,m} L_A^{e,m} + \Sigma_{e=1}^{N} Tr(C^{[e]T} L^{[e]} C^{[e]}) + l_R^{[m]}(X^{[m]}, U^{[r_m]} U^{[c_m]T}) \tag{2.3}$$

$l_R^{[m]}$ is binary cross entropy loss for binary inputs and Frobenius norm for real valued inputs.

To train the network, two phases are followed. In the first phase, the VAE and the fusion network weights are updated. The VAE encoder and decoder weights are updated first by backpropagating $L_A^{(e,m)}$. In the second step, the fusion network and VAE encoder are updated by backpropagating $L_R = l_R^{[m]}(X^{[m]}, U^{[r_m]} U^{[c_m]T})$.

In phase two, the VAE encoder and decoder weights are updated first by backpropagating $L_A^{(e,m)}$. To make sure $C^{[e]}$ is orthogonal, Cholesky decomposition is done. If the weight of the last layer of the feedforward network is set to $\tilde{H}^{[e]^{-1T}}$,

$$\tilde{H}^{[e]} \tilde{H}^{[e]^T} = Cholesky(\tilde{C}^{[e]} \tilde{C}^{[e]^T}) \tag{2.4}$$

$$C^{[e]} = \tilde{C}^{[e]} \tilde{H}^{[e]^{-1T}} \tag{2.5}$$

In the last step of phase two, the reconstruction network and clustering network weights are updated by backpropagating $L_C^{[e]} = Tr(C^{[e]T} L^{[e]} C^{[e]})$.

Thus, DCMTF is similar to CFRM, but has enhancements like the ability to handle multiple data types due to the use of VAEs, the simultaneous learning of representations and clustering.

2.5 Applications

In the subsequent chapters, NCMF is applied to the problem of classification and DCMTF is used for clustering, in the domains of adverse drug event prediction and phenotype discovery respectively. Subsequent chapters will also deal with the domain adaptation problem, using the neural building blocks discussed in this chapter.

Chapter 3

Adverse Drug Event Prediction

Adverse Drug Events(ADEs) are the unintended side-effects caused by drugs, that are difficult to identify in clinical trials. Clinical trials usually are conducted under controlled settings and on a subset of the population. As such, many of these ADEs are caught only in post-market surveillance, making pharmacovigilance important.

Adverse drug events can be caused due to the consumption of a single drug, or due to the simultaneous consumption of more than one drug(called polypharmacy). In this chapter, both these problems are addressed by using unsupervised representation learning, followed by downstream prediction tasks. We evaluate the performance of the NCMF architecture [46] for adverse drug event prediction on two benchmark datasets, and compare the results obtained against existing state-of-the-art results.

The problems addressed in this chapter were solved in collaboration with Mr. Ragunathan Mariappan (Research Associate, Department of Information Systems and Analytics, National University of Singapore) and Mr. Ho Zong Sien (Bachelor of Computing student, Department of Computer Science, National University of Singapore).

3.1 Problem Formulation

The adverse drug event prediction task was modelled as two separate experiments.

- Experiment 1 focused on predicting adverse drug events caused due to the consumption of a single drug, by modelling multi-view interactions between drugs, patients and diseases. This experiment was evaluated on a dataset collected from the MIMIC-III and OFFSIDES databases.

- Experiment 2 considered drug-drug interactions, protein-protein interactions and drug-protein interactions to predict the event of a side-effect arising from polypharmacy. This experiment was evaluated on the dataset described in [83] against the benchmark set by [7] and [10].

3.2 Dataset

3.2.1 MIMIC-OFFSIDES dataset

The MIMIC-III dataset is a publicly available dataset extracted from the EMR records of patients in the Intensive Care Unit in the Beth Israel Deaconess Medical Center in Boston, Massachusetts between 2001 and 2012. The extracted data was de-identified in compliance with HIPAA standards ([29]). MIMIC-III provides fine-grained clinical care information of over 60,000 episodes for ICU admissions, including structured data (i.e., demographics, laboratory tests, diagnosis, medications and so on) and unstructured clinical narratives (clinical notes and discharge summaries). The diagnosis and prescription table was used to obtain information on the diseases and drugs prescribed. The adverse side-effect associated with each drug was collected from the OFFSIDES database. MIMIC follows normalized NDC and short ICD9 codes for drugs and diseases respectively, while OFFSIDES follows RxNORM and decimal ICD9 codes for drugs and diseases respectively. Thus finding the common drugs and diseases between the two datasets were not straightforward. We used the mapping provided by [33] to map NDC to their RxNORM codes. We converted the decimal ICD9 codes in OFFSIDES to short ICD9 using the `icd9`. After mapping the drugs to diseases in both databases, 596 unique drugs and 1321 unique diseases were obtained.

There were 34,807 patients who were administered these 596 drugs and had 1-5 drugs with at least 5 or more side effects listed in OFFSIDES. Since ADEs are associated with longer Length of Stay (LoS) [11], we created a dataset containing patients with both low and high LoS. The mean LoS of all admissions was 258 hours. We sampled three groups of patients, one with LoS less than or equal to 258, another with LoS greater than or equal to 500 (mean + 1 standard deviation) and a third

Sample	**Patients**	**Diseases**	**Drugs**
1	5891	1321	596
2	5890	1321	596
3	5911	1321	596

Table 3.1: Statistics for MIMIC-OFFSIDES Data samples

Experiment	**X0**	**X1**	**X2**
PolyP1	645 x 621135	645 x 22583	22583 x 22583
PolyP2	645 x 645	645 x 837	837 x 837

Table 3.2: Statistics for Polypharmacy Data samples

group with LoS between 258 and 500 hours. These random selections were done three times to obtain 3 sampled datasets. A patient may have multiple admissions and we randomly selected one admission per patient. We randomly selected 10% of the 596 drugs. For each of the selected drugs, we randomly selected 10% of the associated side effects diseases among the 1321 diseases. Now for all drugs i and the side effects disease j selected, we hide the relationship (i,j) if it exists in the data. The hidden pairs becomes our test set.

For the three samples we created, we selected randomly 59 drugs (note that we select the same number of drugs(10% of 59) for all 3 samples, but the drugs selected vary between the samples) and the associated 120, 90, 74 diseases (for samples 1, 2 and 3 respectively). Then we created a test set containing 1742, 1859 and 1923 test entries (for samples 1, 2 and 3 respectively).

For the selected patients, the prescriptions table had 643 unique drugs containing 100 drugs not present in the 596 drugs common across MIMIC and OFFSIDES. We added these 100 drugs to create a patient-drug matrix with 698 drugs. Also, for the selected patients, the diagnosis table had 4534 unique diseases. We selected 100 most frequent diseases in addition to the 1321 diseases common across MIMIC and OFFSIDES to create a patient-disease matrix with 1421 diseases. The available information (presence/absence of side effect) for the chosen drugs and diseases from OFFSIDES formed the third drug-disease matrix. Further details are provided in table 3.1

3.2.2 Polypharmacy dataset

The benchmark polypharmacy dataset from [83] was used in this experiment. The dataset involved drug-drug, drug-protein and protein-protein interactions. A drug-drug interaction, of the form drugA-sideeffect-drugB, indicated that the simultaneous consumption of drug A and drug B could cause the sideeffect mentioned in the triplet. This data was obtained from SIDER and TWOSIDES databases.

Drug-protein interactions indicated that the drug targeted the specific protein mentioned. Protein-protein interactions were indicative of experimentally documented interactions between proteins, recorded in humans. The preprocessing done in [83] was followed, by which only 963 most common sideeffects occurring in atleast 500 combinations were considered. This resulted in 645 drugs and associated side effects.

Two versions of the dataset were created for conducting experiments - PolyP1 and PolyP2, whose descriptions are provided below.

3.2.2.1 PolyP1 dataset

The PolyP1 dataset used the processed train-test split of the dataset provided by [7]. The resulting dataset had 645 drugs and 22583 proteins. The drug-protein and protein-protein matrices were constructed as binary matrices, where a 1 indicated the presence of an interaction and 0 the absence. For the drug-drug matrix, an exhaustive 645 x 621135 (i.e. 645 drugs * 963 sideeffects) dimension matrix was created. The row entities were the 645 drugs and the column entities represented (drug, side-effect) pairs. Given a triplet (drugA, sideeffect B, drugC), the cell in the matrix corresponding to row drugA and column (drugC, sideeffectB) was set to 1. So was the cell corresponding to row drugC and column (drugA, sideeffect B), since the resulting matrix must be symmetric. We do not address the cold start problem and assume that all drug and drugSE instances in the test data (425883 in total) are present at least once in the train set. The prediction task was to identify the sideeffect associated with a drug pair, from the available set of 963 side-effects, or an absence of side-effect in a subset of the drug-drug matrix. In PolyP1, [7] provides a dedicated train-validation-test split of the drug-drug interactions. The training dataset was made up of the train split of these drug-drug interactions alongwith the complete drug-protein and protein-protein interactions. The drug-drug interactions in the validation and test data splits were hidden from the model by setting the corresponding cell value in the matrix to 0.

3.2.2.2 PolyP2 dataset

In the PolyP2 dataset, we do not learn representations for individual sideeffects. Instead we model the drug-drug matrix also as a binary matrix with a 0 indicating an absence of interaction between the drugs in a drug pair and 1 the presence. For example, given a triple (drugA, sideeffectB, drugC), the cell corresponding to row drugA and column drugC was set to 1. So was the cell corresponding to row drugC and column drugA, since the resulting matrix must be symmetric. The same 645 drugs are used as in PolyP1, but only 837 of the proteins are used. Only proteins that interacted with more than 20 drugs were selected for generating the drug-protein and protein-protein matrices. These matrices were also binary valued, with a 1 indicating the presence of an interaction and 0 the absence. In PolyP2, the drug-drug matrix is divided into 80-20 train-test split by randomly sampling cells from the drug-drug matrix. The random sampling was done three times to create three separate samples. For every drug, we ensure that atleast one interaction is present in the training data. The test data points are hidden from the model, by setting them to 0.

The statistics associated with both experiments are detailed in 3.2

3.3 Experiments

For both datasets described in 3.2, the results obtained were compared against matrix factorization methods - CMF[65], gCMF[32], DFMF[84] and DCMF[44]- and benchmark heterogeneous network embedding(HNE) methods as in [78]. The training dataset was passed in as adjacency matrices for the matrix factorization methods, and as graphs to the HNE methods.

Three different methods were used to evaluate and compare the performance of NCMF on each dataset.

3.3.1 Link prediction

For the MIMIC-OFFSIDES dataset and the PolyP2 datasets, 20% of the matrix of interest, drug-disease and drug-drug matrix respectively,were hidden away from the NCMF model. The remaining 80% of the matrix was used for training the model in an unsupervised manner, to obtain representations. In PolyP1, the train and test splits were retained as in [7], with the train dataset being used for learning representations.

The test dataset was used for evaluating the representations learnt by NCMF. The evaluation procedure is the same as in [78]. For the evaluation, a downstream link prediction task was modelled using the test dataset, which comprised of both positive and negative links.

Thus, for the MIMIC-OFFSIDES dataset, the test dataset was made up of triples of the form (drug, sideeffect, label). For PolyP2, the test dataset was made up of triples of the form (drugA, drugB, label). In both cases, label was set to 0 to indicate negative link and 1 to indicate a positive link. For each triplet of the form (entityA, entityB, label), the representations for entityA and entityB were obtained from the NCMF trained model and a Hadamard product of the two representations formed the design matrix. A support vector classifier, from sklearn, was trained on these representations, with the label set to the label associated with the triplets. The scoring metrics used were AUC and MRR, which were averaged over five cross-validation folds.

In the PolyP1 experiment, to ensure that it would be comparable to the current state of the art in [7] and [10], the testing was done on the pre-defined test dataset. The test dataset was made of triples of the form (drugA, sideeffectB, drugC, label). The label was set to 0 to indicate an invalid triplet and 1 to indicate a valid triplet. A valid triplet is one where the simultaneous consumption of the participating drugs is responsible for the occurrence of the associated sideeffect in the triplet. The representation corresponding to each triplet was obtained by concatenating the NCMF representations of drugA and (drugC, sideeffectB) pair. The resulting representations were used, alongwith the labels to train a Random Forest classifier.

Since the memory requirements for the PolyP1 experiment were higher due to the presence of a 645 x 621135 matrix, we had to include an additional preprocessing step. Since we do not address the cold start problem, we ensure that all the entities in the test dataset are present atleast once in the train dataset. This meant that it was essential to retain 425883 of the (drug, sideeffect) pairs, thereby reducing the matrix dimensions down to 645 x 425883. Additionally this matrix was passed in through a ZINB autoencoder from [18], to obtain a 1024 dimensional representation for each of the 645 drugs, and a 32 dimensional representation for each of the (drug,sideeffect) pairs. The resulting 645 x 1024 drug-representation matrix was passed into NCMF, with the drug-protein and protein-protein matrix, to learn 50 dimensional representations for the drugs and proteins. For each of the test triplet of the form (drugA, sideeffectB, drugC), the corresponding representation was obtained by concatenating the 50-dimensional NCMF representation of drugA with the 32 dimensional DCA representation for (drugC, sideeffectB). The resulting representations and their labels were used to train a RandomForest classifer, to predict if the triplet was valid or not. The results were averaged over 963 sideeffects in the dataset, with AUC, AUPRC and AP@50 as the metrics.

3.3.2 Visualization and Clustering

Second, we visually inspect the representations learnt through NCMF and that of the best-performing baseline CMF method, for each dataset. We use hypertools [26] to view the representations in 2 dimensions, using the default option of Incremental PCA to obtain the plots. We also evaluate how well the learnt representations discern intrinsic clusters in the data. Since the representations are high-dimensional, we choose the first five principal components and use them to find clusters using K-Means. In the MIMIC-OFFSIDES dataset, we choose the patient entity labelled by the 3 categories of Length of Stay (LoS) as described in 3.2.

In the Polypharmacy dataset, we choose the drug entity. There is no single label provided in the data to categorize drugs. So, we consider the top 5 most frequently associated target proteins as (distinct) binary labels and report the mean performance. With respect to each of these labels, we compute the Adjusted Rand Index (ARI) to evaluate clustering performance.

		AUC	AUPRC	AP@50
PolyP1	NCMF	**0.954** ± 0.028	**0.927** ± 0.045	0.801 ± 0.182
	WDW	0.935 ± 0.02	0.913 ± 0.031	**0.909** ± 0.064

Table 3.3: Performance of NCMF on benchmark datasets, compared with the best previous methods. WDW: Weighted Deepwalk.

3.3.3 Runtime

Third, we compare the runtime of the methods on each of the datasets. Note that only the training data splits are used for representation learning.

3.4 Results and conclusion

The comparison between the matrix factorization methods and HNE methods were done using three standards - link prediction, visualization and clustering of embeddings and runtime comparison.

3.4.1 Link Prediction

For PolyP1, the results for link prediction are compared against the state of the art Weighted Deep Walk approach proposed in [10]. The results are documented in Table 3.3.

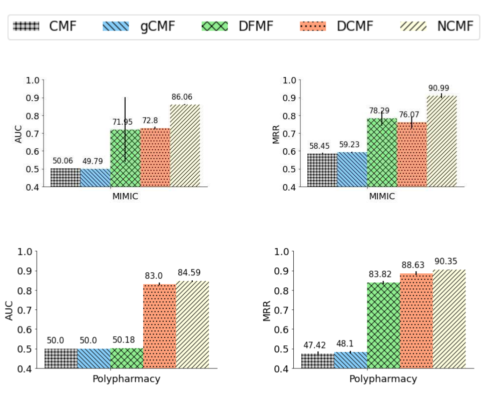

Figure 3.1: Relation Prediction: Comparison with CMF-based methods.

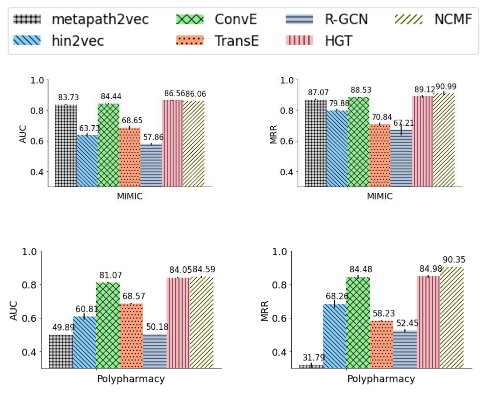

Figure 3.2: Relation Prediction: Comparison with HNE methods

For PolyP2 and MIMIC-OFFSIDES datasets, the results comparing NCMF against benchmark matrix factorization methods and HNE methods are depicted in Figure 3.1 and Figure 3.2. Fig. 3.1 shows the performance of NCMF and baseline

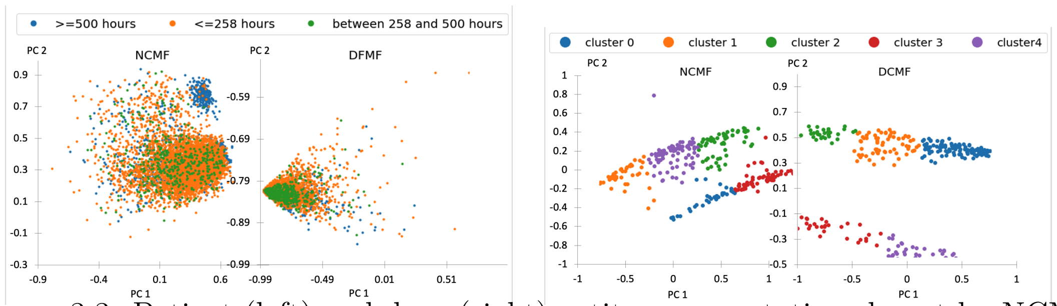

Figure 3.3: Patient (left) and drug (right) entity representations learnt by NCMF on MIMIC-OFFSIDES and PolyP2 datasets respectively; PC: Principal Component.

CMF methods on all datasets. On both metrics NCMF significantly outperforms other methods. DFMF is the second best performing method in the MIMIC-OFFSIDES dataset. DCMF, the other neural approach is the next best in the PolyP2 dataset and is comparable to that of DFMF in MIMIC-OFFSIDES.

3.4.2 Visualization and Clustering

Protein ID	1128	768	1269	1142	3269
NCMF	-0.043795	-0.001245	0.690685	-0.002904	0.055541
DCMF	-0.062869	0.008015	0.883528	0.000758	0.143492

Table 3.4: ARI for each of the 5 most frequent target proteins

Fig. 3.3 shows the 2-dimensional visualizations (obtained via PCA) of patient representations in the MIMIC dataset and drug representations in the PolyP2 dataset, where colors depict corresponding cluster labels. The representations from NCMF are compared with those from the next best performing method (with respect to relation prediction). In all cases, representations from NCMF appear to be well spread out. ARI values shown in Table 3.5 indicate better clustering in two datasets. These results indicate that the representations learnt from heterogeneous inputs capture correlations with respect to the chosen labels well.

MIMIC - OFFSIDES		PolyP2	
NCMF	DFMF	NCMF	DCMF
0.1535	0.1063	0.1396	**0.1945**

Table 3.5: ARI values for clustering with NCMF and next-best performing CMF method.

Worst Performing		Best Performing	
Side Effect	AUPRC	Side Effect	AUPRC
NAUSEA	0.804044	SPLENECTOMY	0.996205
ASPARTATE_AMINOTRANSFERASE_INCREASE	0.804452	METHAEMOGLOBINAEMIA	0.996333
ARTERIAL_PRESSURE_NOS_DECREASED	0.805014	NASAL_POLYP	0.996559
ANAEMIA	0.806016	CARBUNCLE	0.997082
BODY_TEMPERATURE_INCREASED	0.808163	TYMPANIC_MEMBRANE_PERFORATION	0.997171
FEELING_UNWELL	0.810073	SKIN_STRIAE	0.997221
DIFFICULTY_BREATHING	0.810612	CUTANEOUS_CANDIDIASIS	0.997649
DIARRHEA	0.812514	PRIMARY_BILIARY_CIRRHOSIS	0.998471
HEAD_ACHE	0.812977	COCCYDYNIA	0.999326
DRUG_TOXICITY_NOS	0.814319	FASCIITIS	0.999533

Table 3.6: Best and worst performing side effects based on AUPRC.

The ARI values in Table 3.5 for the PolyP2 dataset are averages obtained from the values in Table 3.4.2 for 5 most frequently associated target proteins in the data.

Table 3.6 shows the top 10 best performing side effects and bottom 10 worst performing side effects with respect to the AUPRC scores achieved by NCMF. Such an analysis was also done for their model Decagon by [83]. Similar to their analysis we find that side effects in the bottom 10 are those with non-molecular origins and/or are common complaints, such as feeling unwell, headache and diarrhea. Their and our top 10 lists have several side effects in common, such as carbuncle, coccydynia and tympanic membrance perforation. A clear distinction between our score ranges and theirs can be observed: our worst performing scores range from 0.804 to 0.814 while theirs range from 0.679 to 0.712; similarly our best performing scores range from 0.996 to 0.999, while theirs range from 0.908 to 0.964. This is not surprising since both ESP method of [7] and WDW of [10] were found to outperform Decagon on the same dataset and NCMF outperforms ESP and WDW.

Table 3.5 and Figure 3.3 provide the results of clustering and visualization on MIMIC-OFFSIDES and PolyP2 datasets. As can be observed, there seem to be distinct clusters formed based on the LoS duration, for the MIMIC-OFFSIDES dataset.

3.4.3 Runtime

The inference procedures in all the CMF methods are iterative by design. In the neural methods, DCMF and NCMF, training is done over multiple epochs. The number of iterations/epochs differ in each dataset depending on when the convergence criterion is met. The total time taken per dataset and the number of iterations/epochs are shown in Tables 3.8 and 3.9 respectively. The relative differences are seen in Table 3.7 itself, that shows the time taken per iteration or epoch. DFMF is the fastest; the neural methods NCMF and DCMF are comparable in speed and slower than DFMF. However, both DCMF and NCMF are faster than CMF and gCMF. Implementations of the methods are in R for CMF and gCMF, in python for DFMF and in pytorch (with GPU usage) for DCMF and NCMF. We note that a strict comparison would require ensuring uniformity in use of programming language and hardware. Nevertheless, this comparison provides a practical understanding of the runtimes involved with the available implementations.

Dataset	NCMF	DFMF	DCMF	CMF	gCMF
MIMIC - OFFSIDES	2.4	0.1	3.6	39.1	38.6
Polypharmacy	0.7	0.01	0.3	2.8	2.9

Table 3.7: Time in seconds per iteration/epoch

Dataset	NCMF	DFMF	DCMF	CMF	gCMF
MIMIC-OFFSIDES	2371 seconds	9.048 seconds	18 seconds	2.175 hours	1.072 hours
Polypharmacy	354 seconds	1.659 seconds	1.4077 seconds	9.508 mins	4.99613 mins

Table 3.8: Time per dataset

Dataset	NCMF	DFMF	DCMF	CMF	gCMF
MIMIC - OFFSIDES	1000	100	5	200	100
Polypharmacy	490	100	5	200	100

Table 3.9: Number of epochs

3.5 NCMF Experiment Settings

The code, data, and instructions needed to reproduce all experimental results are in the repository `https://github.com/ncmfsrc/ncmf`. Table 3.10 lists the hyperparameters used for NCMF in our experiments for each dataset. These were chosen through hyperparameter tuning, for which the ranges and choices made are listed in Table 3.11.

Dataset →	MIMIC-OFFSIDES	PolyPharmacy
Hyperparameter ↓		
Learning rate	1e-6	1e-4
Weight Decay	5e-1	1e-4
Batch Size	2048	1024
Number of Iterations	1000	1000
Number of encoding and decoding layers	3	3
Encoding/Decoding activation function	tanh	relu
Entity representation size	50	50
Convergence criterion	$-1e^{-3}$	$1e^{-6}$

Table 3.10: Final Hyperparmeters used to run NCMF with the datasets

Dataset →	MIMIC-OFFSIDES	PolyPharmacy
Hyperparameter ↓		
Learning rate	1e-6 to 1e-3	1e-6 to 1e-3
Weight Decay	1e-4 to 0.5	1e-4 to 0.5
Batch Size	1000 to 2800	1000 to 2800
Number of Iterations	100 to 1000	100 to 1000
Number of encoding and decoding layers	3	3
Encoding/Decoding activation function	relu,tanh	relu,tanh
Entity representation size	50,100	50,100

Table 3.11: Range of hyperparmeters tried with NCMF by trial and error, before manually selecting the final based on the validation set's RMSE of the matrix reconstruction

All the experiments were run using Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz and NVIDIA Tesla V100S PCIe 32 GB graphics card. To enable replication of results, we used random seeds and are hardcoded in the source files. Table 3.12 lists the methods (where applicable) and the corresponding random seed values.

Method	Random seed
NCMF	0
R-GCN	1
HGT	1
ConvE	17
DistMult	17

Table 3.12: Random seed values

3.6 Conclusions

NCMF is found to outperform previous CMF-based methods and several representative HNE methods. On the benchmark datasets, NCMF outperforms previous methods with best known results.

Chapter 4

Phenotype Discovery

Lymphoma is the most common type of cancer in adolescents, of which about two-thirds is categorized as Hodgkin's Lymphoma [57]. This accounts for about 0.5% of newly diagnosed cancer cases in the United States. The role of genetic mutations in causing cancer was first discovered in 1982 [53], when it was identified that a point mutation of the HRAS gene caused an activation of the oncogene in the T24 human bladder carcinoma cells. This led to the field of human genetics, that tries to find variations in the DNA sequence that can relate to diseases and other phenotypes [9].

With the advent of next-generation sequencing, sequencing the entire human genome has become much faster compared to the traditional Sanger method [2]. Single Nucleotide Polymorphisms (SNPs) are variations in a single position of the DNA sequence and can vary from one individual to another. If the same variation occurs in more than 1% of the population, it is classified as a SNP. If the SNP occurs in a gene location, it can affect the phenotype of susceptibility to a particular disease [70]. However, since the human genome is approximately 3 billion base pairs long, the number of variations that have been documented is also large. Identifying groups amongst patients who exhibit these SNPs in their genome can potentially yield clinically meaningful phenotypes. Clustering is a commonly used approach for deriving the underlying patterns in these patient phenotype groups.

In the case of cancer patients, they can be characterized by various features. Clinical features like age, weight, serum creatinine, smoking habits and other features like gender, ethnicity can characterize a patient. Each patient can also be characterized by the nature of SNP present in various positions on their genome. [16] describes how chemotherapy using drugs like bleomycin, methotrexate, cyclophosphamide and others can lead to pulmonary diseases, i.e. chemotherapy toxicity. Features such as days to toxicity from first and last chemotherapy exposure, extent of toxicity (measured by CTCAE toxicity grade) are also features that can be collected for each patient.

Thus clustering methods must allow for multi-view datasets [50]. In this chapter, we use the Deep Collective Matrix Tri-Factorization(DCMTF) method [45] to analyze clinical and genetic data from patients diagnosed with Hodgkin's Lymphoma and find clinically meaningful clusters.

The work described in this chapter was derived from the DCMTF model developed by Mr. Ragunathan Mariappan (Research Associate, Department of Information Systems and Analytics, National University of Singapore). The dataset was curated by Dr. Folefac Aminkeng (Research Assistant Professor, Department of Medicine, National University of Singapore). Dr. Kee Yuan Ngiam (Group Chief Technology Officer, National University Health System) and Dr. Lin Jing (Senior Data Scientist(Manager), Department of Biomedical Informatics, National University Health System) provided valuable insights and guidance on the interpretation of the clinical and genetic information, as well as understanding the significance of the results obtained.

4.1 Problem Formulation

To discover clinically meaningful phenotypes in patients suffering from Hodgkin's Lymphoma, we model this as a clustering problem. The clustering algorithm must learn representations of patients based on various views of the patient, such as their clinical features(both numeric and categoric) and the information about the SNPs in their genome. Apart from this, each of the SNPs have associated auxiliary information, available in public databases like dbSNP [61] and ClinVar [34]. dbSNP provides auxiliary information like the gene where the SNP resides. ClinVar contains information on the clinical significance of specific SNPs, such as diseases found to be associated with the SNP. Thus phenotype discovery is formulated as a multi-view clustering problem.

4.2 Relevant Terminology

Single Nucleotide Polymorphism(SNP) are variations in the DNS, such that the nucleotide at one specific position of the DNA gets altered. These SNPs are associated with a specific position in the human genome.

PHRED Score is a measure of the quality of DNA sequencing. Higher values indicate better quality. A score of 20 indicates that in 1 in 100 base pairs are incorrectly identified.

Allele is a variant of a gene. Humans are considered diploid organisms since we have two copies of an allele at a location, one from each parent.

4.3 Dataset

The primary patient dataset was curated by a team of researchers at the National University Health System, and comprised of 249 patients diagnosed with Hodgkin's Lymphoma. Some of these patients displayed signs of toxicity due the cancer therapy they were administered. The dataset captures this in the associated clinical features, like CTCAE toxicity grade, days to toxicity from first and last bleomycin exposure etc. The three main type of features in the dataset are as listed below:

- Clinical Numeric features - age, weight, serum creatinine, creatinine clearance, cumulative bleomycin exposure, total number of bleomycin cycles, CTCAE toxicity grade and more

- Clinical Categoric features - Subtype of Hodgkin's Lymphoma, chemotherapy treatment protocol, site of radiation, concomitant chemotherapy, medicines prescribed, comorbidities like asthma, tuberculosis, category of lung injury and so on.

- SNP information - the allele pair present in various reference SNPs for each patient is documented, like rs1801133, rs121434295 and so on.

Apart from this, auxiliary information on each of the reference SNPs was obtained from dbSNP and ClinVar databases. This included information like the gene corresponding to each SNP, diseases found to be most associated to each variant (as per medical literature and empirical studies reported by clinicians), chromosome information corresponding to each SNP.

4.4 Data Collection from ClinVar and dbSNP

ClinVar is a freely accessible, public archive of reports of the relationships among human variations and phenotypes, with supporting evidence [34]. The Single Nucleotide Polymorphism database (dbSNP) is a public-domain archive for a broad collection of simple genetic polymorphisms [31].

An additional file, referred as the patient genetic marker file, provided by the researchers at NUHS, contained additional information on a set of 636121 SNPs, like the chromosome and position of the SNP in the genome, associated PHRED score(a measure of the quality of DNA sequencing), reference allele(nucleotide base present in the reference genome) and alternate allele(any other nucleotide base) identified. This file was used as one of the references to obtain auxiliary information from ClinVar and dbSNP.

For both ClinVar and dbSNP, the human genome build 37 was used as the reference. The Clinvar dataset consisted of 1,105,449 genetic variants, associated with 36 features each, such as the chromosome and position on the genome where the variant occurs, reference and alternate allele, clinical significance (benign, pathogenic etc), clinical review status, clinical diagnosis and more. Since ClinVar contains information on genetic variants apart, from single nucleotide polymorphisms, the first step was to filter out only the SNPs from the available dataset. This resulted in 985700 variants. Since we were only interested in the SNPs measured in the patients, a left join was done between the rows in the patient genetic marker file and the 985700 variants from ClinVar, based on chromosome and position of the SNP. In some cases, ClinVar had multiple rows associated with the same SNP - they varied mainly in the way the Chromosome and position were denoted. The duplicate rows were dropped and this resulted in 631557 unique ClinVar records, one per SNP.

Since the patient genetic marker file was manually curated and annotated, it could be prone to errors. To alleviate this, the reference and alternate allele features from ClinVar and the patient genetic marker files were compared, yielding 636559 SNPs. Also, the patient genetic marker file had variant names as the unique identifier, but these did not follow the convention used in dbSNP or ClinVar. This necessitated processing of the variant names used in the patient genetic marker file. In dbSNP and ClinVar, a unique identifier is the rsID, or the reference SNP ID, which is global across all assemblies. These are the SNPs that have been clustered, integrated and annotated. Extracting the rsID corresponding to each SNP in the patient genetic marker file was necessary to enable integration with the auxiliary information from dbSNP.

The variant names, which are the closest to rsID in the patient genetic marker file, was first grouped into three - rsID in ClinVar matched the variant name, rsID did not match variant name in ClinVar and rsID was null. For the rows where rsID was null, the variants were of the form GSA-rs62303689, seq-rs730880444 etc. All alphabets and special characters were removed from the variant name, to get the updated rsID. A left join was done between the rsIDs on the combined dataset of patient genetic marker file and ClinVar, with the data from dbSNP. This resulted in a combined auxiliary dataset, sourced from ClinVar and dbSNP, consisting of 696351 rows and 94 features. dbSNP also contains multiple rows for the same SNP, varying in the chromosome and position, thus the duplicate rows were dropped to obtain 636075 rows, one for each SNP, henceforth referred as the auxiliary dataset. The gene information(i.e. the gene to which the variant belongs to) is obtained from dbSNP, if not available in the patient genetic marker file.

To combine this information with the patient dataset, consisting of 249 patients and 759458 measured SNPs, first the SNPs with >50% missing values and only one unique value across all patients were dropped, resulting in 702058 SNPs. From the auxiliary dataset, the rows corresponding to insertion/deletion operations were removed, and the intersection of the remaining variants in the auxiliary dataset and the reduced patient SNPs was obtained. This resulted in 594267 SNPs, described by more than 90 features. Of these SNPS, 28456 SNPs had a PHRED score > 20 and 8297 SNPs had PHRED score > 30.

4.5 Preprocessing

The data collection step results in three sets of data as listed below:

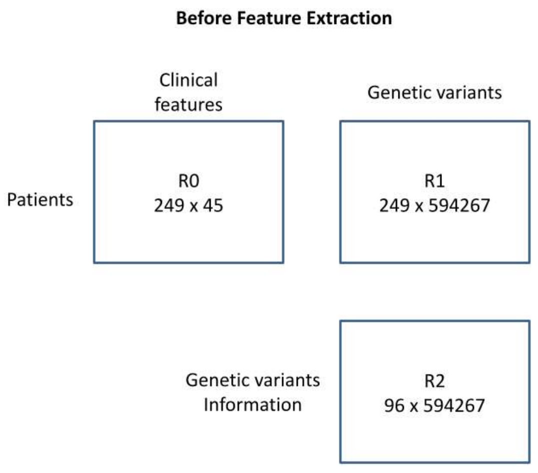

Figure 4.1: Dataset before preprocessing

- R0 - Clinical features associated with 249 patients (both real and nominal)
- R1 - Genetic variants present in these patients
- R2 - Auxiliary information about these genetic variants, from publicly available datasets like dbSNP and ClinVar.

Figure 4.1 represents the dataset available before processing.

4.5.1 Pre-processing and Feature Engineering in R0

The R0 dataset contains the clinical attributes associated with each of the 249 patients. Patients are characterized by features like the drugs administered during chemotherapy, creatinine clearance (for measuring kidney function), subtype of Hodgkin's Lymphoma diagnosed, gender, ethnicity, smoking and drinking habits etc. Chemotherapy induced toxicity is captured through features like age at which bleomycin treatment started, the days to toxicity from first and last bleomycin exposure, the level of toxicity, cumulative bleomycin exposure.

Diagnosis	CHL	NSHL	MCHL	LR	LD	NLPHL
Mixed Cellularity Subtype	1	0	1	0	0	0
Lymphocyte Rich Subtype	1	0	0	1	0	0
Nodular Lymphocytic Predominant with focal chronic lymphocytic leukaemia	0	0	0	0	0	1

Table 4.1: Examples for one hot encoding primary diagnosis based on Hodgkin's Lymphoma subtypes. Notations are CHL - Classical Hodgkin's Lymphoma, NSHL - Nodular Sclerosis Hodgkin's Lymphoma, MCHL - Mixed Cellularity Hodgkin's Lymphoma, LR - Lymphocyte Rich, LD - Lymphocyte Depleted, NLPL - Nodular Lymphocyte-predominant Hodgkin's Lymphoma.

4.5.1.1 Feature Engineering

Features like primary diagnosis, treatment protocol, concomitant chemotherapy, other concomitant therapy and co-morbidities were available only as textual inputs, since they were notes manually entered in by doctors. To convert these into feature vectors, feature engineering was done. Details of feature engineering are listed below.

Primary Diagnosis

The primary diagnosis was an indicator of the type of Hodgkin's Lymphoma the patient was diagnosed with. There are two disease groups in Hodgkin's Lymphoma - Classical Hodgkin's Lymphoma(HL) and Nodular Lymphocyte-predominant Hodgkin's Lymphoma(NLPHL) [19]. Classical Hodgkin's Lymphoma is further split into four categories - nodular sclerosis(NSHL), mixed cellularity(MCHL), lymphocyte-rich and lymphocyte-depleted. Based on the diagnosis provided by clinicians, an encoded vector is assigned for each patient. Examples of this encoding are available in table 4.1.

Treatment Protocol

The treatment protocol followed for various patients describes the number of bleomycin cycles administered and the chemotherapy regime used. For example, the a treatment protocol written as "ABVD 1 Cycle, 3 cycles ESC. BEACOPP" means that the patient was administered 1 cycle of Doxorubicin(Adriamycin), Bleomycin, Vinblastine, Dacarbazine (ABVD) and 3 cycles of escalated Bleomycin, Etoposide, Adriamycin(doxorubicin), Cyclophosphamide, Oncovin(Vincristine), Procarbazine, Prednisone (BEACOPP), making a total of 4 bleomycin cycles. Table 4.2 describes the mapping of each treatment protocol to the corresponding drugs administered.

Protocol	Drugs administered
ABVD	Adriamycin(doxorubicin), Bleomycin, Vinblastine, Dacarbazine
APVD	Adriamycin(doxorubicin), Procarbazine, Vinblastine, Dacarbazine
BEACOPP	Bleomycin, Etoposide, Adriamycin(doxorubicin), Cyclophosphamide, Oncovin(Vincristine), Procarbazine, Prednisone
CHOP	Cyclophosphamide, Adriamycin(doxorubicin), Vincristine, Prednisone
ICE	Ifosfamide, Carboplatin, Etoposide
CHOEP	Cyclophosphamide, Adriamycin(doxorubicin), Etoposide, Vincristine, Prednisone
ESHAP	Etoposide, Methylprednisolone, high dose Cytarabine, Cisplatin
AVD+A	Adriamycin(doxorubicin), Vinblastine, Dacrabazine, Brentuximab Vedotin
ABVE-PC	Adriamycin(doxorubicin), Bleomycin sulfate, Vincristine, Etoposide, Prednisone, Cyclophosphamide
EVA	Etoposide, Vincristine, Adriamycin(doxorubicin)

Table 4.2: Mapping between treatment protocol and drugs administered

To process this feature, this mapping is used to create an encoded vector depicting the drugs administered to the patient.

Concomitant Chemotherapy

Concomitant chemotherapy includes drugs or radiation therapy administered to reduce treatment time. This follows the same format as the treatment protocol. As a result, the same mapping highlighted in table 4.2 can be used to create one hot encoded vectors for each patient. In addition a radiation therapy component also may be introduced in the patient's treatment regime.

Since the features extracted from concomitant chemotherapy and treatment protocol only denotes if the patient has been administered the drug, drugs appearing in both features are retained only once. For example, if treatment protocol is ABVD and concomitant chemotherapy is CHOEP, the drug Adriamycin is only assigned a value 1 to denote that it has been administered to the patient.

Other Concomitant Medications

This includes drugs such as Filgrastim, Pegfilgrastim, Granulocyte colony-stimulating factor (GCSF), Pegylated GCSF etc that help the body to produce more immune cells and fight off any infections that could arise from chemotherapy. These drugs are represented through their dosage, for example, if the feature other concomitant medications is filled out by the clinician as "GCSF x 43 doses", then the GCSF dosage is set to 43, and the others to 0. Once the drug dosage is represented, the dosage for GCSF and Filgrastim are added together to create one feature, to replace the individual dosages. The same is done for Peg-GCSF and Pegfilgrastim.

Comorbidities

Comorbidities indicate the presence of other health conditions alongwith Hodgkin's Lymphoma and chemotherapy related toxicity. The comorbidities are also denoted using a one hot encoded vector indicating the secondary condition present in the patient. This dataset consists of patients with pneumonia, asthma, tuberculosis exposure, pyelonephritis (a condition affecting the kidneys), upper respiratory tract infection (URTI) and pneumothorax.

4.5.1.2 Pre-processing

The features in R0 were first divided up based on the data type, as real(matrix X0) and categorical(matrix X1).

For the real valued features in X0, like weight, serum creatinine, creatinine clearance, cumulative bleomycin exposure, missing values were imputed using the mean. Missing values in CTCAE toxicity grade were imputed with 0. In case of imputing missing values for the feature "days to toxicity from first bleomycin exposure", control patients(i.e. patients who do not show toxicity response to chemotherapy) are assigned a large value like 1000 days to indicate that they do not have an adverse reaction. For case patients with missing values for this feature, the mean of the other patients is used for imputation. A new feature to indicate the number of years that have passed since the patient was first administered bleomycin is also created.

The categorical valued features, like ethnicity, alcohol consumption habits, gender, smoking habits etc are grouped together in matrix X1. The patients are assigned to four broad ethnic groups, as shown in table 4.3. For the features like status of patient at last review, smoking and drinking habits the missing values are estimated with the most conservative option, i.e. alive for status of patient at last review and non smoker for smoking habits. All the categories were one hot encoded.

Category	Ethnicity values
Middle East	Arabian, Emirati, Omani, UAE
South Asia	Bangladeshi, Indian, Pakistani, Sri Lankan, Sikh
South East Asia	Vietnamese, Malay, Indonesian, Chinese, Boyanese
Europe	Caucasian, Eurasian

Table 4.3: Mapping of patient ethnicity into categories

After pre-processing R0, two matrices are obtained - real valued X0 of dimensions 249 x 13 and binary valued X1 of dimensions 249 x 106.

4.5.2 Pre-processing and Feature Engineering in R1 and R2

Human beings are diploid organisms, where for each genetic location two copies(alleles) are available - one inherited from the father and the other from the mother. An allele is a variant form of the gene. If the two alleles are the same at a given location, then it is termed homozygous and heterozygous otherwise. For each of the 249 patients, the nucleotide inherited from both biological parents is available at each of the 594267 SNP locations. For example, for the SNP rs3131972, patient 0 has a value G_G indicating that it is homozygous, while patient 4 has a value A_G making it heterozygous.

For each of the 594267 SNPs, the number of times each nucleotide occurs amongst the 249 patients is counted. The nucleotide occurring the least number of times is assigned as the minor allele for the SNP, and vice versa for the major allele. Its fraction over the number of alleles is considered to be the minor allele frequency. Additive encoding is done to convert the values into ordinal variables with the mapping - Homozygous major = 0, Heterozygous = 1 (for example, A_C and C_A), Homozygous minor = 2.

4.5.2.1 Feature selection

Since the matrix R1 consists of 594267 features and only 249 associated patients, it is necessary to do feature selection.

Initial attempts of feature selection involved the use of Lasso regression against the case/control status of patients (regarding their toxicity response to chemotherapy). The SNPs were divided based on the chromosome they belonged to and Lasso regression was run. Features with non-zero coefficient were retained, resulting in

5310 features. This approach was discarded as it created a dependence on the case/control feature. If the learnt representations had to be clustered and analyzed, this approach would not be appropriate.

Using SVD based truncation could reduce dimensionality, but lost the interpretable nature of the dataset. Since the aim was to find clusters even amongst the SNPs, the SVD approach to dimensionality reduction was not taken forward.

Other approaches to feature reduction included running a Chi-square test between an arbitrarily chosen outcome variable (like case/control status, status of patient at last review etc) to find relevant variables. However, this approach also suffered from the same issue as the Lasso regression based approach.

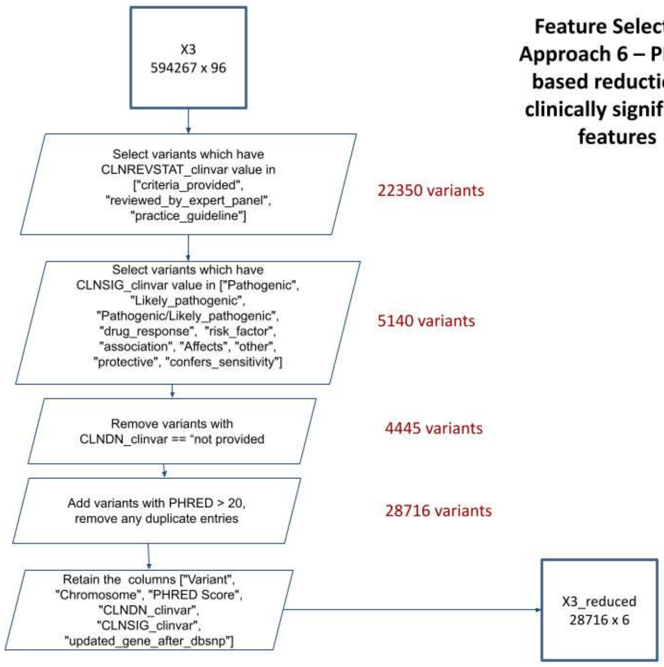
X3
594267 x 96
Feature Selection Approach 6 – PHRED based reduction + clinically significant features
Select variants which have CLNREVSTAT_clinvar value in ["criteria_provided", "reviewed_by_expert_panel", "practice_guideline"]
22350 variants
Select variants which have CLNSIG_clinvar value in ["Pathogenic", "Likely_pathogenic", "Pathogenic/Likely_pathogenic", "drug_response", "risk_factor", "association", "Affects", "other", "protective", "confers_sensitivity"]
5140 variants
Remove variants with CLNDN_clinvar == "not provided
4445 variants
Add variants with PHRED > 20, remove any duplicate entries
28716 variants
Retain the columns ["Variant", "Chromosome", "PHRED Score", "CLNDN_clinvar", "CLNSIG_clinvar", "updated_gene_after_dbsnp"]
X3_reduced
28716 x 6

Figure 4.2: Feature selection flowchart

The approach to feature selection that was chosen incorporated dimensionality reduction and retained the interpretability of the features. This involved combining the data available in both R1 and R2 matrices. This approach is detailed in figure 4.2. Only SNPs that are found to be clinically relevant as per the ClinVar database, with a PHRED score above 20 were selected. This resulted in an auxiliary matrix of dimensions 28716 x 6. The features retained from the R2 dataset were Chromosome, PHRED score, associated disease diagnosis from Clinvar, Clinical significance and gene information from dbSNP. The same 28716 SNPs were chosen from the R1 dataset to obtain a matrix X2 of dimensions 249 x 28716. All SNPs with only one unique value across all 249 patients were dropped. This resulted in an X2 matrix of dimensions 249 x 15962. The auxiliary matrix was also updated to match the SNPs.

The auxiliary matrix was divided up further into 3 matrices X3, X4 and X5 based on the gene, chromosome and disease diagnosis information. X3 matrix was made up of 6947 genes(rows) associated with each of the 15962 SNP(columns). It was a binary matrix with a 1 indicating the gene where each SNP could be found. X4 matrix was made up of 24 chromosomes (22 autosomes and one each for the X and Y chromosomes) and 15962 SNPs. The value in the matrix was 1 if the SNP along the column was located in the chromosome along the row, and 0 otherwise. X5 matrix was made up of 1732 diseases associated with the 15962 SNPs. The value was set to 1 if a disease along the row was found to be associated with a SNP along the column, and a 0 otherwise. The resulting matrix structure after processing is depicted on the figure 4.3.

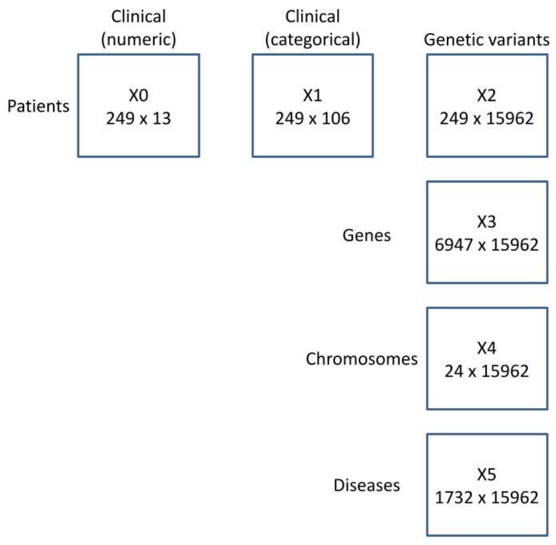

Figure 4.3: Matrix structure after processing

4.6 Experiment Setting

Once the matrices are processed and readied, they are passed into the DCMTF architecture, with a defined number of clusters for each entity. In this case, there are 7 entities - patients, clinical real features, clinical categorical features, SNPs, genes, chromosomes and diseases - in 6 matrices. Only the X0 matrix is set to use the Frobenius norm as the loss, since it has real values, while all the other matrices use the cross entropy loss since they are categoric in nature. The DCMTF algorithm returns an association matrix along with the clusters assigned to each of the entities. The hyperparameters set for the execution of DCMTF are listed in the table 4.4.

Hyperparameter	Value
Number of batches	50
Activation function	tanh
Learning rate	1e-8
Convergence threshold	1e-3
Maximum epochs	2000
Number of cluster	2

Table 4.4: Hyperparameters for DCMTF

4.6.1 Cluster number selection

To select the appropriate number of clusters in each entity, a quantitative analysis was conducted using silhouette score. The 6 matrices were passed through the NCMF architecture to obtain lower dimensional representations for each entity. For varying values of k between 2 and 9, the silhouette scores were obtained after clustering the NCMF representation associated with each entity. The cluster number with the maximum silhouette score was chosen in each case. This experiment yielded the maximum silhouette score for 2 clusters, for all entities. This value was used for the DCMTF clustering task. The silhouette score and inertia plots for all entities after clustering on NCMF representations are shown on table 4.5.

4.7 Evaluation and Results

The DCMTF algorithm returned both the association matrix A (in figure 4.4) and the cluster labels assigned to each of the entities. The association matrix A

k	E0	E1	E2	E3	E4	E5	E6
2	**0.9677**	**0.8125**	**0.8243**	**0.8280**	**0.9954**	**0.2963**	**0.9856**
3	0.2489	0.7326	0.7290	0.7538	0.8974	0.2342	0.9804
4	0.2422	0.6754	0.5818	0.6888	0.8373	0.2069	0.9202
5	0.1632	0.1758	0.5517	0.5544	0.1400	0.1975	0.8920
6	0.1621	0.1390	0.5295	0.3300	0.0911	0.1084	0.8920
7	0.1514	0.1190	0.5205	0.3151	0.1217	0.0858	0.6401
8	0.1084	0.0598	0.5119	0.3035	0.1164	0.1424	0.6389
9	0.1168	0.0601	0.5028	0.2008	0.0804	0.0745	0.4472

Table 4.5: Silhouette scores for various cluster numbers(k) after clustering the NCMF representations for patients(E0), clinical numeric features(E1), clinical categoric features(E2), SNPs(E3), genes(E4), chromosomes(E5) and diseases(E6).

denotes how well connected each cluster of an entity is with a cluster of another entity. These association values can be used to identify cluster chains amongst the entities.

The results obtained from DCMTF are evaluated in two ways -

- Adjusted Rand Index(ARI) score comparison between an experimental setup consisting of patient and auxiliary datasets and another experimental setup consisting of only the patient datasets (X0, X1 and X2 matrices)
- Evaluating the clinical significance of clusters obtained

4.7.1 ARI score comparison

DCMTF incorporates information from auxiliary data sources as well. As a result, the clusters formed from 6 matrices are expected to have a better ARI score than the ones formed from just 3 matrices. In both cases, the ARI is calculated between the patient clusters and the case/control status label, since one of the topics of interest is to identify the association with chemotherapy toxicity. It was found that the ARI scores of the 6 matrix setup was higher than that of the 3 matrix setup, indicating that the auxiliary datasets enhance the clustering capabilities. Comparing DCMTF against CFRM showed that DCMTF was able to learn better clusters with the presence of auxiliary matrices, which could not be done by CFRM. The results are highlighted on the table 4.6.

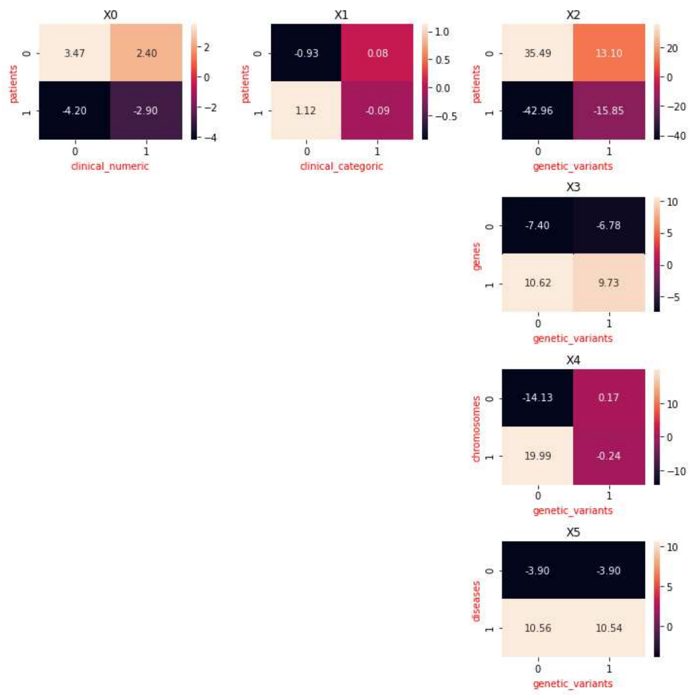

Figure 4.4: Cluster association matrix from DCMTF

Algorithm	6-matrix	3-matrix
DCMTF	0.0953	-0.0055
CFRM	-0.0053	-0.0053

Table 4.6: ARI scores with(6-matrix) and without(3-matrix) auxiliary information

A comparison of the cluster labels assigned to the 249 patients, compared against their case/control status is displayed in Figure 4.5. The patient representations learnt from DCMTF implementation are projected down to two dimensions using TSNE. There appears to be some association between the cluster labels and the case/control status of patients.

4.7.2 Clinical Significance of Clusters

The clusters can be analyzed by following the cluster chain based on the association matrix A (Figure 4.4) obtained from DCMTF. Since our interest lies in understanding the key differences between the patient clusters formed, the cluster chain analysis starts from the $X0$ matrix. The sub-matrices that have the maximum

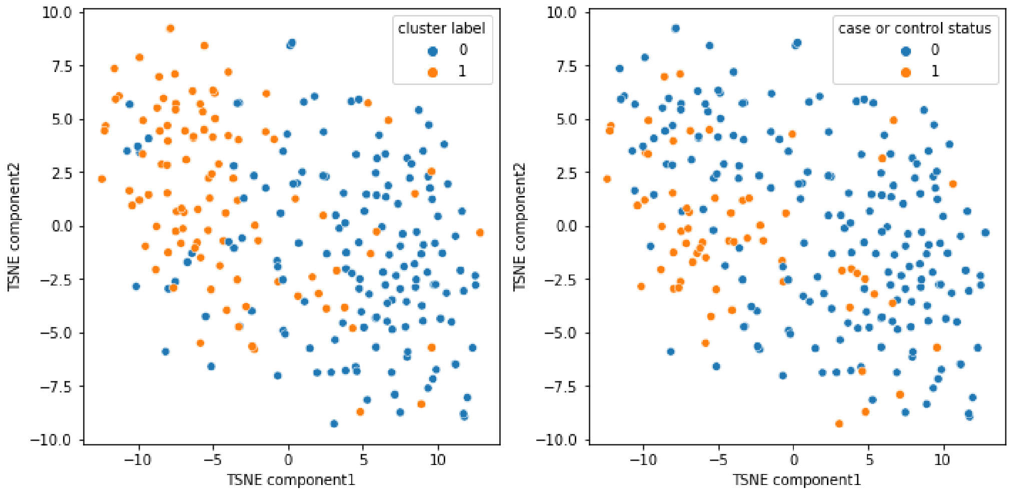

Figure 4.5: TSNE embeddings for patient representations obtained from DCMTF, differentiated by their cluster labels, (left) compared against the case or control status of the patients (right).

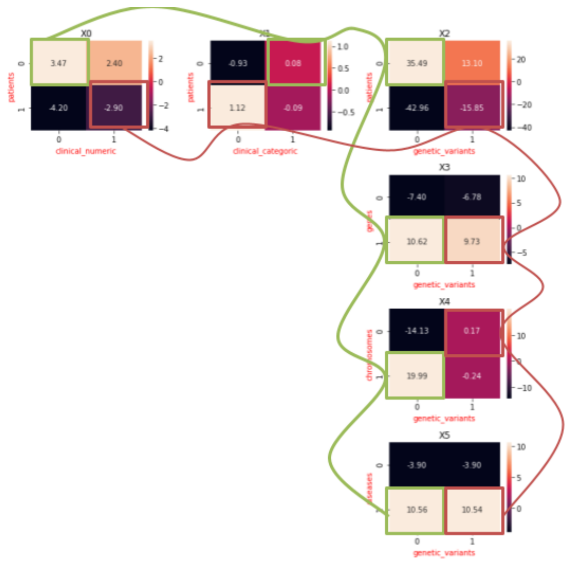

Figure 4.6: Cluster chains formed for patient clusters 0 (green chain) and 1 (red chain). We follow the matrix that has the maximum reported cluster association value.

association with the patient cluster chosen are selected for the analysis. The chains formed are shown in Figure 4.6.

Following cluster chain 1 in Figure 4.6, a set of SNPs(7195) were found to be associated with patient cluster 1. These SNPs are spread across various genes. The next step was to go to the $X3$ matrix and get the genes to which most of these

SNPs belong to. The same step was done on the $X5$ matrix. The BRCA2 and MYH7 genes were found to be connected to >20 SNPs. When we extended this to the $X5$ matrix, the diseases hereditary breast and ovarian cancer syndrome and hypertrophic myopathy were found to correspond to the most number of SNPs. These results were validated against the dbSNP database.

According to dbSNP[61] results for MYH7, *"Mutations in this gene are associated with familial hypertrophic cardiomyopathy, myosin storage myopathy, dilated cardiomyopathy, and Laing early-onset distal myopathy"* [14]. Likewise, for the BRCA2 gene, dbSNP results indicated *"Inherited mutations in BRCA1 and this gene, BRCA2, confer increased lifetime risk of developing breast or ovarian cancer."* [8]. Both these associations were captured in the cluster chains found by DCMTF and provide evidence supporting the clusters formed.

The notebooks and code are available at `https://github.com/ajayago/nuhs_pipeline/tree/pipeline`.

4.8 Conclusions

Thus, the clusters formed by DCMTF have been found to be effective in identifying clinically meaningful clusters(phenotypes) from multi-view datasets, incorporating both clinical information of patients and auxiliary data sources. The ARI scores, in 4.6, also show a significant improvement with the addition of auxiliary data sources, compared to existing multi view clustering methods like CFRM which do not improve even with the addition of auxiliary data.

Further studies could be focused on finding a good cluster number to be used in DCMTF. In this project, the cluster number was set using the cluster number corresponding to the maximum silhouette score for NCMF representations of each entity. Another possible direction, specific to the problem statement, would be to include gene ontology annotation tools, to understand how the genes in one cluster are linked through biological processes.

Chapter 5

Domain Invariant Representations: Drug Response Prediction

In this chapter, we propose a novel Neural Collective Domain Invariant Matrix Factorization(NCDIMF) architecture, to be applied to the problem of learning domain invariant representations. Learning domain invariant representations is important, especially in the field of precision oncology, due to the inherent differences between preclinical datasets available during training time and the actual dataset available during model deployment [59]. Since the preclinical datasets are mostly acquired under controlled laboratory settings, they would most probably follow a different distribution than the actual dataset on which the model would run, once deployed. We use NCDIMF to learn such domain invariant representations, to facilitate the task of predicting drug responses in cell line and patient derived xenografts, based on the genetic mutation and copy number variation data. The learnt representations must be indicative of the gene expression data, which is only available at train time, and not at test time.

The model developed in this chapter derives from the work done by Mr. Ragunathan Mariappan (Research Associate, Department of Information Systems and Analytics, National University of Singapore), Alicia Nanelia (Masters student, Department of Information Systems and Analytics, National University of Singapore) and Krishna Kumar Hariprasannan(Master of Computing student, National University of Singapore).

5.1 Domain Invariant Representations

5.1.1 Motivation

Traditional machine learning methods assume that the train dataset and the test dataset are both obtained *i.i.d.* This assumption allows the machine learning models to be trained on a subset of the dataset and tested on the other. The data seen by the model after deployment is assumed to follow the same distribution. If $P_{tr}(X, Y)$ and $P_{te}(X, Y)$ represent the joint probability distribution of train and test datasets respectively (X is the set of input features, Y is the outcome variable), the assumption is that $P_{tr}(X, Y) = P_{te}(X, Y)$ [60].

However, there could be a distribution shift between the train and test datasets, i.e. the two could be from two different ***domains***. Such a shift could be caused due to sampling methods used, time based shifts and more. This is especially true in the field of precision oncology. Learning a representation that can be domain invariant reduces the discrepancy between these domains, making feature space learnt from this approach work well on unseen domains.

5.1.2 Precision Oncology: Background

In the field of precision oncology, the focus is on using varied sources of patient data such as genetic mutation information, tumor and cell-free DNA profiling, RNA and proteomic analyses for targeted treatment therapies that can give the best patient outcome. As per [55], this involves ensuring the right patient is provided the right treatment, at the right time, in the right dosage. One of the main problems in precision oncology is to identify the "right drug". It has been found that since cancer is a genomic disease, caused due to alterations in the genome, moving to personalized treatment regimens rather than empirical cancer therapy is more effective [6]. Due to the differential responses to drugs amongst individuals, there is a need to predict the efficacy of various drugs on the individual, so as to prescribe the right drug. Pharmacogenomics focuses on using the genome of the patient to find the most effective treatment.

However, one of the main challenges faced in precision oncology is the dearth of clinical pharmacogenomic datasets. Issues such as public access, small size of patient cohorts and the low number of treatment therapies on which clinical trials have been conducted are the cause for this lack. Since it is not possible to get these clinical datasets, most often cell-line [58] and patient derived xenograft(PDX) [21] datasets are used as proxies, to predict drug response. The work done by [24] supports the need to have standardised drug responses and identification of reliable genomic predictors of drug response, thus highlighting the inconsistencies across large pharmocogenomic datasets. Other available datasets include The Cancer Genome Atlas dataset [69], which is also used in our project.

Here, our goal is to capture domain invariant representations of the cell-line, PDX and TCGA datasets, to allow the model to generalize well to unseen domains. [48] shows that while there are differences between the pre-clinical datasets and the human tumor datasets, there are some underlying shared characteristics that can be captured through domain adaptation methods. [48] provides two major methods for finding this domain invariant representation, as listed below:

- Data centric - a common subspace is found from both domains of data, by aligning their marginal distributions
- Subspace centric - first the domains are reduced down to a lower dimensional space and alignment is done on this feature space.

We choose the second method, since it captures variations in the two domains without comparing the actual distribution itself. In this project, we learn the lower dimensional representations of entities in two domain- cell-line and PDX, or cell-line and TCGA- domains using a novel architecture NCDIMF, where we use an alignment loss to ensure the feature space in both cases is domain invariant. The alignment loss used here is the same as the one in [59].

5.2 Dataset Description

The cell-line, PDX and TCGA datasets are publicly available, and were sourced from [1], [21] and [69] respectively. One of the key differences between the two datasets - cell line and PDX - and the human tumor datasets is the absence of the tumor microenvironment and vasculature in these pre-clinical models as opposed to what is observed in the human tumor cells. These differences have been studied in [22] and [3] for cell line and PDX respectively.

5.2.1 Relevant Terminology

Mutations are alterations of the genome DNA sequence. Mutations can occur within specific genes and can be point mutations, affecting a single nucleotide base pair. These mutations can cause cancer if they occur in specific genes. It has been found that mutations in the cancer drug target genes can greatly affect the efficacy of drugs [28].

Copy Number variations are alterations in the genome that are typically larger than single nucleotide polymorphisms. These could involve copy number gain or loss, indicating the addition of a set of bases in the genome or their deletion. The presence of CNVs cause variations in the protein formation cycle, thereby affecting cell function. [63] has found the association between copy number variation in cancer susceptibility and tumor progression.

Gene expression is the process by which the information in the genes are converted into functional entities like proteins, thereby causing a phenotype. The genes produce mRNA, which is translated into proteins. Measurement of mRNA is an important part of gene expression measurement [52].

5.2.2 Cell Line Dataset

The cell lines are laboratory cultured cancer cells that have been used extensively in cancer research and drug discovery [47]. Immortalized cell lines are a population of cells which can be kept alive under laboratory cultured conditions, while still keeping some phenotypes and associated functions [39]. These are widely used due to their homogeneous nature, ease of culturing to grow the population in a short span of time. These cells generally originate from a single ancestor cell.

In this project we obtain cell line dataset for 692 cell line entities. Here cell line entities refer to cell line donors. For each of these 692 cell line entities, the presence of genetic mutations and copy number variations(CNV) are documented. 324 genes are explored to obtain both these values. We also have the gene expression data for these 692 cell line entities, over 1946 genes. Thus we have three matrices - CNV matrix X0(692 x 324), mutation matrix X1(692 x 324) and gene expression matrix X2(692 x 1946).

In the cell line dataset, the drug efficacy is determined by the area under the drug response curve. This metric varies from 0 to 1; a lower value indicates a better response to the drug.

5.2.3 PDX dataset

[27] describes how patient derived xenografts have become *in vivo* models used for cancer research. Primary or metastatic human tumor cells, collected via surgery or biopsy, are implanted in mice. This *in vivo* process was developed based on the belief that the mice would simulate a similar environment to that of the human body, and help understand the spread of cancer better, than in their *in vitro* counterparts. These models have been used as a precursor for drug trials.

In this project, we obtain PDX dataset for 178 PDX entities, over 324 genes. Like for the cell line dataset, both mutations and copy number variation data is available. The gene expression data is also available over 322 genes. The three matrices available are CNV matrix X0(178 x 324), mutation matrix X1(178 x 324) and gene expression matrix X2(178 x 322).

In the PDX dataset, the treatment response is captured through RECIST scores [17]. These are usually distributed across four labels - complete response (CR), partial response (PR), progressive disease (PD) and stable disease (SD). Due to the limited number of available data points, in our project, we group the categories CR, PR and SD into one category and PD as a separate category. Thus, a positive/no change(label 1) in the patient is considered one category and an adverse outcome(label 0) is another category.

5.2.4 TCGA dataset

The Cancer Genome Atlas program [69] contains genomic alteration information for about 30 human tumors. This includes data such as mutations, copy number variations, gene expression to name just a few. For our study, we consider only these three sources of data. We collect the mutation and copy number variation data for the 324 genes as is available in cell line and PDX datasets, for 171 patients.

Since this dataset is based on actual patients, this could be a valuable source of information.

The drug response here is measured as being effective or ineffective, i.e. a binary value. The three matrices available for use are the CNV matrix X0(171 x 324), mutation matrix X1(171 x 324) and gene expression matrix(171 x 324).

5.3 Existing Architecture

Out of distribution generalization and domain generalization have been surveyed by [60] and [72]. [59] proposed the Velodrome architecture for a semi-supervised method for out of distribution generalization.

5.3.1 Velodrome Architecture

In Velodrome [59], the model learns from labeled cell line and unlabeled patient domains as inputs. The outcome variable is the drug response. The gene expression matrix for both domain entities are passed through a feature extractor network, followed by separate predictor networks for each domain. The feature extractor network projects both domains to a shared feature space and uses the separate predictor networks for drug response prediction.

It uses a combination of three losses to perform domain generalization - (i) supervised loss for ensuring drug response predictions are accurate (ii) consistency loss to exploit unlabelled samples in representation learning and (iii) alignment loss for making the learnt representation generalizable. This architecture surpasses transfer learning based models due to the ability to avoid the use of target domain during training. Since it follows a semi-supervised mode of learning representations, this method can leverage the large number of unlabelled source datasets available.

5.3.2 Supervised Autoencoders

Supervised autoencoder [35] have been found to perform well in semi-supervised settings. They predict the inputs, like in standard autoencoder, as well as the output. A supervised loss is added on the representations obtained from the encoder layer. The loss optimized is the sum of the input reconstruction as well as the supervised loss.

5.4 Neural Collective Domain Invariant Matrix Factorization

This section describes how our proposed architecture NCDIMF. This model combined the following paradigms into one architecture:

- multi view representation learning capabilities of NCMF
- domain invariant representation learning from Velodrome
- representation learning methodology from supervised autoencoders.

This model can learn domain-invariant representations from the datasets, in an unsupervised manner. This architecture requires the presence of gene expression, mutation and CNV datasets for both domains. We learn representations from mutation and CNV matrices, that can be representative of the gene expression matrix. Gene expression matrix is used as the target due to its popularity in cancer research, like in the case of Velodrome. It is only available at train time and not during testing.

5.4.1 Network Architecture

The NCDIMF architecture consists of three networks as listed below:

- Variational Autoencoder network - this is used for learning representations of the entities in the matrices passed in.
- Fusion network - this helps learn a joint representation for entities present in multiple matrices
- Gene expression reconstruction network - this is used to reconstruct the gene expression matrix from the learnt entity representations.

The variational autoencoder network and the fusion network are identical to the ones in the NCMF architecture, as in Section 2.3. The gene expression reconstruction network is similar to the matrix completion network of NCMF, except that it only considers the cell line, PDX or TCGA entity representation from the fusion network, instead of combining it with the gene entity representation. The matrices passed

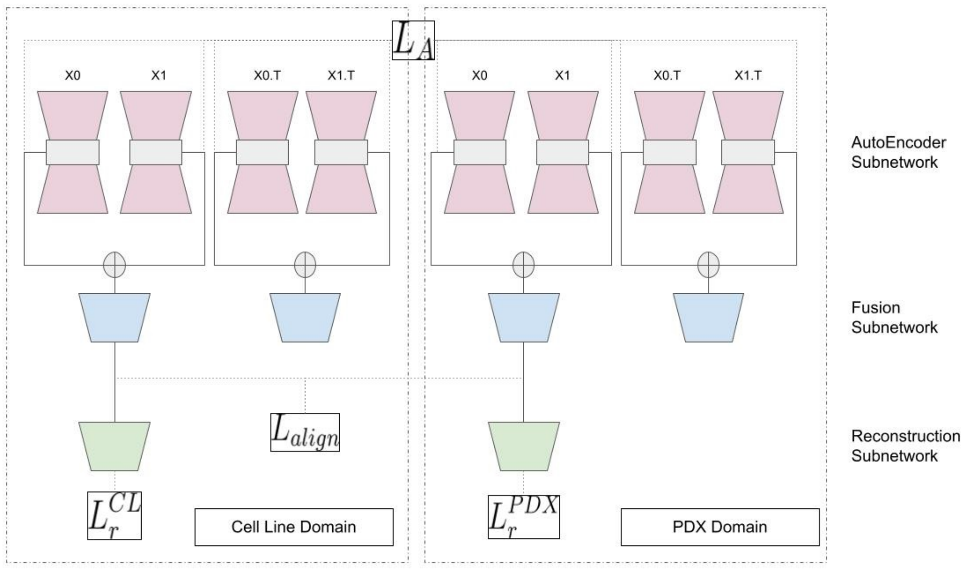

Figure 5.1: NCDIMF model architecture. The model has three subnetworks - autoencoder, fusion and reconstruction networks.

into this architecture is binary valued and hence the autoencoder network uses a ZINB loss. Apart from these networks, an alignment loss is added to ensure domain generalization is done while learning representations. The model architecture is available in Figure 5.1.

5.4.2 Network Training

Here, we will explain the training process with respect to the cell line and PDX datasets. The same can be extend to cell line and TCGA dataset as well, by replacing PDX with TCGA.

Consider the cell line CNV and mutation matrices to be denoted by X_0^{CL} and X_1^{CL}. The gene expression matrix for cell line is denoted by Y^{CL}. Similarly, let the PDX CNV and mutation matrices to be denoted by X_0^{PDX} and X_1^{PDX}. The gene expression matrix for cell line is denoted by Y^{PDX}. There are two entities in these domains - the entity forming the dataset(cell line or PDX) and the genes.

Two instances of the autoencoder network, the fusion network and the gene expression reconstruction network are constructed, one for each dataset. The PDX and cell line datasets are divided into mini batches, just as in NCMF, but ensuring that the number of batches across both domains is the same. Each minibatch is passed through the autoencoder, fusion and gene expression reconstruction networks.

For each entity e in matrix m, the autoencoder network returns a lower dimensional representation $U_m^{[e]}$. Since we are only interested in the cell line or PDX entity, we consider only the output from the fusion network for this entity. If the cell line or PDX entity is denoted by e_0, the fusion network f_θ returns the representation $g_{fusion} = f_\theta(\gamma(U_{X0}^{[e_0]}, U_{X1}^{[e_0]}))$, where $\gamma(.)$ represents the concatenation operation. The fusion network concatenates the entity representation for e_0 learnt from both X0 and X1 matrices, and converts it to a lower dimensional embedding. The gene expression reconstruction network uses the representation g_{fusion} for generating a gene expression matrix R^{CL} or R^{PDX}, of the same column dimensions as Y^{CL} and Y^{PDX} respectively.

There are three loss terms that are backpropagated for training the three networks simultaneously.

- $L_A^{(e,m)}$ - autoencoder reconstruction loss for each of the entities across all the matrices
- $L_r^{e_0}$ - gene expression reconstruction loss, calculated using the root mean square error between the Y^{CL} or Y^{PDX} and the reconstructed gene expression matrix R^{CL} or R^{PDX}.
- $L_{align}^{e_0}$ - alignment loss calculated by CORAL loss [67].
  $L_{align}^{e_0} = \Sigma_{N_{CL}} \Sigma_{N_{PDX}} ||g_{fusion}^{CL} - g_{fusion}^{PDX}||_2$, where N_{CL} and N_{PDX} represent the number of data points in the cell line and PDX minibatches.

$$L_{net} = L_A^{(e,m)} + L_r^{e_0} + L_{align}^{e_0} \tag{5.1}$$

5.4.3 Experiment Settings

For all datasets, the two matrices (X0 and X1) indicating CNVs and mutations are used by NCDIMF for learning representations. The values in the mutation matrix are binary, where a 1 indicates the presence of a mutation and 0 the absence. CNV matrix has -1, 0 and 1 indicating copy number removal, no change and addition respectively. These values are further one hot encoded to ensure that the matrices

Algorithm 2: Neural Collective Domain Invariant Matrix Factorization(NCDIMF)

```
   Inputs  : Matrices $\mathcal{X} = X_0^{(CL)}, X_1^{(CL)}, X_0^{(PDX)}, X_1^{(PDX)}$
   Outputs : Trained network $\mathcal{N}$ to generate Entity representations
             $\mathcal{U} = U_{e_0}^{[CL]}, U_{e_0}^{[PDX]}$, and Reconstructed Gene expression Matrices
             $R^{CL}$ and $R^{PDX}$
   // Dynamic Network Construction
1: Construct network $\mathcal{N}$ with $\mathcal{A}^{(e,m)}$ $\forall(e,m) \in Q\forall(CL, PDX)$,
     $f^{[e_0]}\forall e_0 \in (CL, PDX)$ and $r^{(m)}$ $\forall e_0 \in (CL, PDX)$
   // Network Training
2: repeat
3:     for $m \in \{1, \cdots, M\}$ do
4:         for each mini-batch (block) in $X^{(m)} \in CL, PDX$ do
5:             Forward pass through subnetworks $\mathcal{A}^{(e,m)}$, $f^{[e_0]}$, $r^{(m)}$
6:             Backpropagate $L_{net} \leftarrow L_A^{(e,m)} + L_r^{e_0} + L_{align}^{e_0}$ on sampled entries
                 and update weights of all networks
7: until Stopping Criteria Reached;
```

being passed into the NCDIMF architecture are binary. The gene expression matrix for both datasets contain real valued entries. The cell line dataset is split into 623 train and 69 test data points. The PDX dataset is split into 161 train and 17 test data points. TCGA dataset is split into 153 train and 18 test data points. Only two of the three domains are considered at a time. The gene expression matrices are scaled using a standard scaling method.

Once representations are learnt from any two domains taken together, these representations are used in a downstream prediction task, where drug fingerprints for 71 drugs are combined with the learnt representations to predict the drug response. In the cell line dataset, the output predicted is the area under the dose response curve, for TCGA dataset and PDX it is the binary class label.

5.4.4 Hyperparameters

The table 5.1 shows a list of hyperparameters used for training this model. These were set based on manual tuning, and the observed changes in the loss value.

Hyperparameters	Values
Weight decay	0.5
Learning Rate	1e-6
Convergence threshold	1e-5
Number of batch	10
Representation dimension	128

Table 5.1: Hyperparameters used for training the model

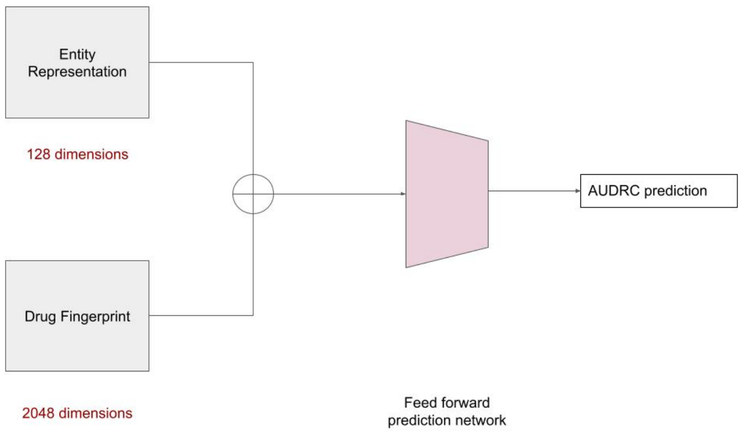

Figure 5.2: Evaluation Network for representations learnt by NCDIMF.

5.4.5 Evaluation

The model learns representations for the cell line, PDX and TCGA entities. These learnt representations are evaluated through a downstream drug response prediction task. The learnt representations are concatenated with drug fingerprints and used to train a feedforward network, which predicts the Area Under Dose Response Curve score (AUDRC) for cell lines and a binary outcome for TCGA.

The test splits of the datasets are passed through the autoencoder and fusion networks of the architecture. The resulting 128 dimensional representations are concatenated with 2048 dimensional drug fingerprints to calculate the drug response on these entities, using the feed forward network. Only 71 drugs, approved for use in Singapore, are evaluated here. The testing flow is shown in Figure 5.2.

For cell lines, the predicted area under the dose response curve(AUDRC) is used to order the drugs, such that drugs with a lower AUDRC is preferred. The top k drugs can be printed out for efficacy scores. The topk drugs are compared against the actual topk drugs that have been found to be effective on these cell lines, using the NDCG score. The NDCG score considers both the recommended drugs as well as the order in which they are recommended for evaluation. A higher score is an indication of a good recommendation.

For PDX, the only information available is the binarized RECIST score. The predictions are compared against the true labels(0/1) to obtain the area under the receiver operating curve. When we considered the number of (PDX entity, drug) pairs from the 17 test PDX entities and 71 approved drugs, which had a RECIST score available, we obtained only 10 such pairs, making this a very limited test set. Due to this limited size, we did not perform evaluation on this set.

For TCGA, we consider the binary labels provided, indicating if the drug was effective or not. This category is predicted by a downstream classifier. We had a set of 18 TCGA patients, which yielded 22 (TCGA patient, drug) pairs for testing.

5.5 Results

To evaluate the learnt representations, the drug efficacy was predicted on a set of held-out cell line entities. The NDCG score was used to evaluate the top k drugs that would yield the best results for the cell line under consideration. The k value was set to 1, 3, 5 and 10, and the corresponding NDCG score was calculated. The results are summarized in Table 5.2.

The results obtained were compared against the NDCG scores obtained when the cell entity representations were provided by a variational autoencoder (AEC model), from only the mutation information of the cell entity (referred as AEC model in Table 5.2).

Likewise, an evaluation was done on the TCGA test entities, where the AUROC score was used since it was a binary classification task. The results are available in Table 5.2. In the case where NCDIMF is trained on cell line and PDX datasets, we do not obtain the AUROC for the TCGA cell line entities, since the number of available features vary across both PDX and TCGA domains. PDX has 545 features while TCGA has 945 features, due to the one hot encoding done as part of pre-processing.

Model	NDCG @1	NDCG @3	NDCG @5	NDCG @10	AUROC
VAE model	0.6251 ±0.2858	0.855 ±0.0949	0.8117 ±0.0973	0.8549 ±0.0715	0.7426
NCDIMF (cell line + PDX)	0.6261 ±0.3245	0/7823 ±0.1357	0.8106 ±0.1172	0.8247 ±0.08	-
NCDIMF (cell line + TCGA)	0.6187 ±.3204	0.7823 ±0.1357	0.8103 ±0.117	0.8231 ±0.0807	0.5384

Table 5.2: NDCG scores on cell line test dataset and AUROC on TCGA dataset.

The relevant notebooks and code are available at `https://github.com/ajayago/druid`.

5.6 Conclusions

Thus, it can be seen that the proposed model does comparably with the existing variational autoencoder model. We have only used a set of 71 drugs that have been approved for use in Singapore. This set could be expanded further. Further analysis on the top k drugs recommended in both cases could provide insights on the reason for a reduced NDCG score.

Considering a different domain, like sourcing data from The Cancer Genome Atlas(TCGA), also showed comparable results on the cell line test dataset. Tuning of the hyperparameters during representation learning and in the prediction network could be tried with automated hyperparameter tuning methods, like the Ax framework.

Chapter 6

Conclusions and Future Scope

From the experiments conducted over the three problems (Chapters 3, 4 and 5) faced in the biomedical sector, it can be seen that multi-view neural representation learning methods perform well. Their ability to integrate the data from varied sources and data types is invaluable. Enhancing these models to incorporate forms of unstructured data, like text or images, is a possible direction to explore in the future. More specifically, the sections below describe the possible enhancements on the three architectures described in this project.

6.1 NCMF

NCMF is particularly useful in integrative modeling of biomedical data with arbitrary auxiliary information, as seen in our experiments. The main limitation of NCMF is its high running time, both for training and hyperparameter tuning, especially on large input collections. This may be partly alleviated through the use of high-performance hardware. More scalable optimization methods and effective hyperparameter tuning techniques may be developed in the future to improve its running time.

6.2 DCMTF

DCMTF has been able to learn representations of all the entities, learn entity-specific cluster structure (co-clustering) and also learn the associations between clusters of various entities to derive useful cluster chains. Here, like for NCMF, the training time is high and future work can focus on improving this. Use of scalable optimization methods and hyperparameter tuning needs to be explored in this regard.

6.3 NCDIMF

NCDIMF has been found to perform comparably well with existing autoencoder based approaches. The results obtained from NCDIMF are preliminary and need improvement. To this end, ablation studies can be used to identify which loss component is affecting the training more, and use the representations thus learnt for the downstream prediction.

Bibliography

[1] Jordi Barretina et al. "The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity". In: *Nature* 483.7391 (2012), pp. 603–607.

[2] Sam Behjati and Patrick S Tarpey. "What is next generation sequencing?" In: *Archives of Disease in Childhood - Education and Practice* 98.6 (2013), pp. 236–238. ISSN: 1743-0585. eprint: `https://ep.bmj.com/content/98/6/236.full.pdf`. URL: `https://ep.bmj.com/content/98/6/236`.

[3] Uri Ben-David et al. "Patient-derived xenografts undergo mouse-specific tumor evolution". In: *Nature genetics* 49.11 (2017), pp. 1567–1575.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828.

[5] Antoine Bordes et al. "Translating embeddings for modeling multi-relational data". In: *Advances in Neural Information Processing Systems* 26 (2013).

[6] Reinhard Buettner, Jürgen Wolf, and Roman K. Thomas. "Lessons Learned From Lung Cancer Genomics: The Emerging Concept of Individualized Diagnostics and Treatment". In: *Journal of Clinical Oncology* 31.15 (2013). PMID: 23589544, pp. 1858–1865. eprint: `https://doi.org/10.1200/JCO.2012.45.9867`. URL: `https://doi.org/10.1200/JCO.2012.45.9867`.

[7] Hannah A Burkhardt et al. "Predicting adverse drug-drug interactions with neural embedding of semantic predications". In: *AMIA Annual Symposium Proceedings*. Vol. 2019. 2019, p. 992.

[8] Farid Cherbal et al. "BRCA1 and BRCA2 unclassified variants and missense polymorphisms in Algerian breast/ovarian cancer families". In: *Disease markers* 32.6 (2012), pp. 343–353.

[9] Melina Claussnitzer et al. “A brief history of human disease genetics”. In: *Nature* 577.7789 (Jan. 2020), pp. 179–189. URL: `http://dx.doi.org/10.1038/s41586-019-1879-7`.

[10] Soham Dasgupta et al. “Adverse Drug Event Prediction Using Noisy Literature-Derived Knowledge Graphs: Algorithm Development and Validation”. In: *JMIR Medical Informatics* 9.10 (2021), e32730.

[11] Emma C Davies et al. “Adverse drug reactions in hospital in-patients: a prospective analysis of 3695 patient-episodes”. In: *PLoS One* 4.2 (2009), e4439.

[12] Yifan Deng et al. “A multimodal deep learning framework for predicting drug–drug interaction events”. In: *Bioinformatics* 36.15 (2020), pp. 4316–4322.

[13] Tim Dettmers et al. “Convolutional 2D knowledge graph embeddings”. In: *32nd AAAI Conference on Artificial Intelligence*. 2018.

[14] Glauber M Dias et al. “MYH7 p. Glu903Gln Is a Pathogenic Variant Associated With Hypertrophic Cardiomyopathy”. In: *Circulation: Genomic and Precision Medicine* 14.5 (2021), e003476.

[15] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. “Metapath2vec: Scalable representation learning for heterogeneous networks”. In: *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017.

[16] Megan M. Dulohery, Fabien Maldonado, and Andrew H. Limper. “71 - Drug-Induced Pulmonary Disease”. In: *Murray and Nadel’s Textbook of Respiratory Medicine (Sixth Edition)*. Ed. by V. Courtney Broaddus et al. Sixth Edition. Philadelphia: W.B. Saunders, 2016, 1275–1294.e17. ISBN: 978-1-4557-3383-5. URL: `https://www.sciencedirect.com/science/article/pii/B9781455733835000713`.

[17] E.A. Eisenhauer et al. “New response evaluation criteria in solid tumours: Revised RECIST guideline (version 1.1)”. In: *European Journal of Cancer* 45.2 (2009), pp. 228–247.

[18] Gökcen Eraslan et al. “Single-cell RNA-seq denoising using a deep count autoencoder”. In: *Nature Communications* 10.1 (2019), pp. 1–14.

[19] Debra L. Friedman. “Chapter 21 - Hodgkin Lymphoma”. In: *Lanzkowsky’s Manual of Pediatric Hematology and Oncology (Sixth Edition)*. Ed. by Philip Lanzkowsky, Jeffrey M. Lipton, and Jonathan D. Fish. Sixth Edition. San Diego: Academic Press, 2016, pp. 429–441. ISBN: 978-0-12-801368-7. URL: `https://www.sciencedirect.com/science/article/pii/B9780128013687000211`.

[20] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. “Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017.

[21] Hui Gao et al. “High-throughput screening using patient-derived tumor xenografts to predict clinical trial drug response”. In: *Nature medicine* 21.11 (2015), pp. 1318–1325.

[22] Jean-Pierre Gillet, Sudhir Varma, and Michael M. Gottesman. “The Clinical Relevance of Cancer Cell Lines”. In: *JNCI: Journal of the National Cancer Institute* 105.7 (Feb. 2013), pp. 452–458. ISSN: 0027-8874. eprint: `https://academic.oup.com/jnci/article-pdf/105/7/452/7676553/djt007.pdf`. URL: `https://doi.org/10.1093/jnci/djt007`.

[23] Betül Güvenç Paltun, Hiroshi Mamitsuka, and Samuel Kaski. “Improving drug response prediction by integrating multiple data sources: matrix factorization, kernel and network-based approaches”. In: *Briefings in Bioinformatics* 22.1 (2021), pp. 346–359.

[24] Benjamin Haibe-Kains et al. “Inconsistency in large pharmacogenomic studies”. In: *Nature* 504.7480 (2013), pp. 389–393.

[25] Xiangnan He et al. “Neural collaborative filtering”. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 173–182.

[26] Andrew C Heusser et al. “HyperTools: a Python toolbox for gaining geometric insights into high-dimensional data”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 5589–5594.

[27] Manuel Hidalgo et al. “Patient-derived xenograft models: an emerging platform for translational cancer research”. In: *Cancer discovery* 4.9 (2014), pp. 998–1013.

[28] Jing Jin et al. “Identification of Genetic Mutations in Cancer: Challenge and Opportunity in the New Era of Targeted Therapy”. In: *Frontiers in Oncology* 9 (2019). ISSN: 2234-943X. URL: https://www.frontiersin.org/article/10.3389/fonc.2019.00263.

[29] Alistair EW Johnson et al. “MIMIC-III, a freely accessible critical care database”. In: *Scientific Data* 3.1 (2016), pp. 1–9.

[30] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations (ICLR)* (2014).

[31] Adrienne Kitts. “The Single Nucleotide Polymorphism Database (dbsnp) of nucleotide sequence variation”. Feb. 2011. URL: https://www.ncbi.nlm.nih.gov/books/NBK21088/.

[32] Arto Klami, Guillaume Bouchard, and Abhishek Tripathi. “Group-sparse Embeddings in Collective Matrix Factorization”. In: *International Conference on Learning Representations (ICLR)*. 2014.

[33] Fabrıcio SP Kury and Olivier Bodenreider. “Mapping US FDA National Drug Codes to Anatomical-Therapeutic-Chemical Classes using RxNorm.” In: *American Medical Informatics Association*. 2017.

[34] Melissa J Landrum et al. “ClinVar: improving access to variant interpretations and supporting evidence”. In: *Nucleic Acids Research* 46.D1 (Nov. 2017), pp. D1062–D1067. URL: http://dx.doi.org/10.1093/nar/gkx1153.

[35] Lei Le, Andrew Patterson, and Martha White. “Supervised autoencoders: Improving generalization performance with unsupervised regularizers”. In: *Advances in neural information processing systems* 31 (2018).

[36] Jin Li et al. “IMCHGAN: Inductive Matrix Completion with Heterogeneous Graph Attention Networks for Drug-Target Interactions Prediction”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2021).

[37] Jin Li et al. “Neural inductive matrix completion with graph convolutional networks for miRNA-disease association prediction”. In: *Bioinformatics* 36.8 (2020), pp. 2538–2546.

[38] Michelle M Li, Kexin Huang, and Marinka Zitnik. “Representation learning for networks in biology and medicine: advancements, challenges, and opportunities”. In: (2021).

[39] Z. Li. “5.43 - In Vitro Micro-Tissue and -Organ Models for Toxicity Testing”. In: *Comprehensive Biotechnology (Second Edition)*. Ed. by Murray Moo-Young. Second Edition. Burlington: Academic Press, 2011, pp. 551–563. ISBN: 978-0-08-088504-9. URL: `https://www.sciencedirect.com/science/article/pii/B9780080885049005031`.

[40] Herty Liany, Anand Jeyasekharan, and Vaibhav Rajan. “Predicting synthetic lethal interactions using heterogeneous data sources”. In: *Bioinformatics* 36.7 (2020), pp. 2209–2216.

[41] Bo Long et al. “Spectral Clustering for Multi-Type Relational Data”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 585–592. ISBN: 1595933832. URL: `https://doi.org/10.1145/1143844.1143918`.

[42] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[43] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic gradient descent with warm restarts”. In: *International Conference on Learning Representations (ICLR)* (2017).

[44] Ragunathan Mariappan and Vaibhav Rajan. “Deep collective matrix factorization for augmented multi-view learning”. In: *Machine Learning* 108.8 (2019), pp. 1395–1420.

[45] Ragunathan Mariappan and Vaibhav Rajan. “Multi-way Spectral Clustering of Augmented Multi-view Data through Deep Collective Matrix Tri-factorization”. In: *CoRR* abs/2009.05805 (2020). arXiv: `2009.05805`. URL: `https://arxiv.org/abs/2009.05805`.

[46] Ragunathan Mariappan et al. “Neural Collective Matrix Factorization for Integrated Analysis of Heterogeneous Biomedical Data”. In: *bioRxiv* (2022). eprint: `https://www.biorxiv.org/content/early/2022/01/22/2022.01.`

20.477057.full.pdf. URL: https://www.biorxiv.org/content/early/2022/01/22/2022.01.20.477057.

[47] John RW Masters. “Human cancer cell lines: fact and fantasy”. In: *Nature reviews Molecular cell biology* 1.3 (2000), pp. 233–236.

[48] Soufiane Mourragui et al. “PRECISE: a domain adaptation approach to transfer predictors of drug response from pre-clinical models to tumors”. In: *Bioinformatics* 35.14 (July 2019), pp. i510–i519. ISSN: 1367-4803. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/14/i510/28913240/btz372.pdf. URL: https://doi.org/10.1093/bioinformatics/btz372.

[49] Nam D Nguyen and Daifeng Wang. “Multiview learning for understanding functional multiomics”. In: *PLoS Computational Biology* 16.4 (2020), e1007677.

[50] Seyoung Park, Hao Xu, and Hongyu Zhao. “Integrating Multidimensional Data for Clustering Analysis With Applications to Cancer Patient Data”. In: *Journal of the American Statistical Association* 116.533 (2021), pp. 14–26. eprint: https://doi.org/10.1080/01621459.2020.1730853. URL: https://doi.org/10.1080/01621459.2020.1730853.

[51] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “Deepwalk: Online learning of social representations”. In: *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 701–710.

[52] Katerina Pierouli et al. “Introductory Chapter: Gene Profiling in Cancer in the Era of Metagenomics and Precision Medicine”. In: *Gene Expression Profiling in Cancer*. Ed. by Dimitrios Vlachakis. Rijeka: IntechOpen, 2019. Chap. 1. URL: https://doi.org/10.5772/intechopen.84462.

[53] E. Premkumar Reddy et al. “A point mutation is responsible for the acquisition of transforming properties by the T24 human bladder carcinoma oncogene”. In: *Nature* 300.5888 (Nov. 1982), pp. 149–152. URL: http://dx.doi.org/10.1038/300149a0.

[54] Michael Schlichtkrull et al. “Modeling relational data with graph convolutional networks”. In: *European Semantic Web Conference*. Springer. 2018, pp. 593–607.

[55] Lee Schwartzberg et al. “Precision Oncology: Who, How, What, When, and When Not?” In: *American Society of Clinical Oncology Educational Book* 37 (2017). PMID: 28561651, pp. 160–169. eprint: `https://doi.org/10.1200/EDBK_174176`. URL: `https://doi.org/10.1200/EDBK_174176`.

[56] Uri Shaham et al. “SpectralNet: Spectral Clustering using Deep Neural Networks”. In: (Jan. 2018).

[57] Satish Shanbhag and Richard F. Ambinder. “Hodgkin lymphoma: A review and update on recent progress”. In: *CA: A Cancer Journal for Clinicians* 68.2 (2018), pp. 116–132. eprint: `https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21438`. URL: `https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21438`.

[58] Hossein Sharifi-Noghabi et al. “Drug sensitivity prediction from cell line-based pharmacogenomics data: guidelines for developing machine learning models”. In: *Briefings in Bioinformatics* 22.6 (Aug. 2021). bbab294. ISSN: 1477-4054. eprint: `https://academic.oup.com/bib/article-pdf/22/6/bbab294/41088388/bbab294.pdf`. URL: `https://doi.org/10.1093/bib/bbab294`.

[59] Hossein Sharifi-Noghabi et al. “Out-of-distribution generalization from labelled and unlabelled gene expression data for drug response prediction”. In: *Nature Machine Intelligence* 3.11 (2021), pp. 962–972.

[60] Zheyan Shen et al. “Towards out-of-distribution generalization: A survey”. In: *arXiv preprint arXiv:2108.13624* (2021).

[61] S. T. Sherry. “dbSNP: the NCBI database of genetic variation”. In: *Nucleic Acids Research* 29.1 (Jan. 2001), pp. 308–311. URL: `http://dx.doi.org/10.1093/nar/29.1.308`.

[62] Jianbo Shi and J. Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905.

[63] Adam Shlien et al. “Excessive genomic DNA copy number variation in the Li–Fraumeni cancer predisposition syndrome”. In: *Proceedings of the National Academy of Sciences* 105.32 (2008), pp. 11264–11269.

[64] Nikola Simidjievski, Cristian Bodnar, et al. “Variational autoencoders for cancer data integration: design principles and computational practice”. In: *Frontiers in Genetics* 10 (2019), p. 1205.

[65] Ajit P Singh and Geoffrey J Gordon. “Relational learning via collective matrix factorization”. In: *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2008, pp. 650–658.

[66] Genevieve L Stein-O’Brien et al. “Enter the matrix: factorization uncovers knowledge from omics”. In: *Trends in Genetics* 34.10 (2018), pp. 790–805.

[67] Baochen Sun and Kate Saenko. “Deep coral: Correlation alignment for deep domain adaptation”. In: *European conference on computer vision*. Springer. 2016, pp. 443–450.

[68] Jian Tang et al. “Line: Large-scale information network embedding”. In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pp. 1067–1077.

[69] Katarzyna Tomczak, Patrycja Czerwińska, and Maciej Wiznerowicz. “The Cancer Genome Atlas (TCGA): an immeasurable source of knowledge”. In: *Contemporary oncology* 19.1A (2015), A68.

[70] Eva Vallejos-Vidal et al. “Single-Nucleotide Polymorphisms (SNP) Mining and Their Effect on the Tridimensional Protein Structure Prediction in a Set of Immunity-Related Expressed Sequence Tags (EST) in Atlantic Salmon (Salmo salar)”. In: *Frontiers in Genetics* 10 (2020). ISSN: 1664-8021. URL: `https://www.frontiersin.org/article/10.3389/fgene.2019.01406`.

[71] Dorothea Wagner and Frank Wagner. “Between Min Cut and Graph Bisection”. In: *Mathematical Foundations of Computer Science 1993*. Ed. by Andrzej M. Borzyszkowski and Stefan Sokołowski. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 744–750. ISBN: 978-3-540-47927-7.

[72] Jindong Wang et al. “Generalizing to unseen domains: A survey on domain generalization”. In: *arXiv preprint arXiv:2103.03097* (2021).

[73] Qi Wang et al. “Multi-modality disease modeling via collective deep matrix factorization”. In: *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 1155–1164.

[74] Weiran Wang et al. “On Deep Multi-view Representation Learning”. In: *Proceedings of the 32nd International Conference on Machine Learning.* 2015, pp. 1083–1092.

[75] Gregory P Way and Casey S Greene. “Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders”. In: *Proceedings of the Pacific Symposium on Biocomputing.* 2018.

[76] Tian-Ru Wu et al. “MCCMF: collaborative matrix factorization based on matrix completion for predicting miRNA-disease associations”. In: *BMC Bioinformatics* 21.1 (2020), pp. 1–22.

[77] Siwen Xu et al. “Integrative analysis of histopathological images and chromatin accessibility data for estrogen receptor-positive breast cancer”. In: *BMC Medical Genomics* 13.11 (2020), pp. 1–12.

[78] Carl Yang et al. “Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark”. In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[79] Le Ou-Yang et al. “Matrix factorization for biomedical link prediction and scRNA-seq data imputation: an empirical survey”. In: *Briefings in Bioinformatics* (2021).

[80] Hai-Cheng Yi et al. “Graph representation learning in bioinformatics: trends, methods and applications”. In: *Briefings in Bioinformatics* 23.1 (Sept. 2021). bbab340. ISSN: 1477-4054. eprint: `https://academic.oup.com/bib/article-pdf/23/1/bbab340/42229638/bbab340.pdf`. URL: `https://doi.org/10.1093/bib/bbab340`.

[81] Zi-Chao Zhang et al. “A graph regularized generalized matrix factorization model for predicting links in biomedical bipartite networks”. In: *Bioinformatics* 36.11 (Mar. 2020), pp. 3474–3481. ISSN: 1367-4803. eprint: `https://academic.oup.com/bioinformatics/article-pdf/36/11/3474/33329309/btaa157.pdf`. URL: `https://doi.org/10.1093/bioinformatics/btaa157`.

[82] Zi-Chao Zhang et al. “A graph regularized generalized matrix factorization model for predicting links in biomedical bipartite networks”. In: *Bioinformatics* 36.11 (2020), pp. 3474–3481.

[83] Marinka Žitnik, Monica Agrawal, and Jure Leskovec. “Modeling polypharmacy side effects with graph convolutional networks”. In: *Bioinformatics* 34.13 (2018), pp. i457–i466.

[84] Marinka Žitnik and Blaž Zupan. “Data fusion by matrix factorization”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.1 (2014), pp. 41–53.

[85] Marinka Žitnik et al. “Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities”. In: *Information Fusion* 50 (2019), pp. 71–91.