Learning Data-Driven Drug-Target-Disease Interaction via Neural Tensor Network

Huiyuan Chen and **Jing Li**
Department of Computer and Data Sciences, Case Western Reserve University
hxc501@case.edu, jingli@cwru.edu

Abstract

Precise medicine recommendations provide more effective treatments and cause fewer drug side effects. A key step is to understand the mechanistic relationships among drugs, targets, and diseases. Tensor-based models have the ability to explore relationships of drug-target-disease based on large amount of labeled data. However, existing tensor models fail to capture complex nonlinear dependencies among tensor data. In addition, rich medical knowledge are far less studied, which may lead to unsatisfied results. Here we propose a Neural Tensor Network (NeurTN) to assist personalized medicine treatments. NeurTN seamlessly combines tensor algebra and deep neural networks, which offers a more powerful way to capture the nonlinear relationships among drugs, targets, and diseases. To leverage medical knowledge, we augment NeurTN with geometric neural networks to capture the structural information of both drugs' chemical structures and targets' sequences. Extensive experiments on real-world datasets demonstrate the effectiveness of the NeurTN model.

1 Introduction

Data-Driven Drug Discovery. Personalized medicine recommendation is one of the most promising assets to treat human disease [Li *et al.*, 2015]. A critical step in personalized medicine is to understand drugs' mechanism of actions (MoAs) by exploring the biological interactions among drugs, targets, and diseases. *In vitro* experiments can be performed to identify potential associations of drug-target-disease, but such systematic screening remains an expensive and time-consuming process. It takes more than 13 years and $2.87 billion to bring a new drug into the market [Hauser *et al.*, 2017]. Researchers are thus resorting to machine learning to understand the drugs' MoAs by mining the emergence of large-scale chemical and genomic data [Chen and Li, 2018].

Two most prominent data-driven tasks among these recent developments are drug-disease and drug-target prediction (see [Ezzat *et al.*, 2018; Li *et al.*, 2015] for surveys). In these tasks, researchers have attempted to collect a variety of omics data, and predict new interactions of drug-disease or drug-target through network inference [Chen and Li, 2017], multi-view learning [Zheng *et al.*, 2013], and deep learning [Tsubaki *et al.*, 2018]. Beyond pairwise drug-disease or drug-target relationships, some recent studies have pointed at the importance of identifying triple-wise interactions of drug-target-disease in human metabolic systems [Capuzzi *et al.*, 2018; Chen and Li, 2019; Wang *et al.*, 2018]. Among different methods, tensor factorization is a commonly used method to infer the missing entries of a drug-target-disease tensor [Chen and Li, 2019; Wang *et al.*, 2018].

Tensor Factorization. Tensor factorizations aim to extract latent structure from high dimensional data [Kolda and Bader, 2009]. CP (CANDECOMP/PARAFAC) and Tucker are two popular tensor models with diverse variants being successfully applied in health data analysis [Wang *et al.*, 2015; Chen and Li, 2019]. However, the CP and Tucker models (or their variants) suffer from two weaknesses.

First, their performance can be limited by linearity, which might not be expressive well for nonlinear data manifolds. Recently, a series of studies have shown that nonlinear tensor factorizations have superior performances over multilinear tensor models [Xu *et al.*, 2012; Liu *et al.*, 2019; Wu *et al.*, 2019]. For example, InfTucker [Xu *et al.*, 2012] was proposed to use a nonlinear Gaussian kernel. However, they rely on a prior Gaussian process over tensor data, which might be difficult to estimate in practice [Zhe *et al.*, 2016].

The second drawback of CP or Tucker models is the data sparsity issue. To alleviate this issue, coupled tensor-matrix models are extended to jointly analyze tensor together with auxiliary information [Wang *et al.*, 2015; Narita *et al.*, 2012; Acar *et al.*, 2011]. However, these methods are inherently limited by encoding *feature matrices*, which require tedious feature engineering [Shan *et al.*, 2016]. As the number of features grows, designing and deploying them become challenging, especially for healthcare data.

Contributions. To tackle these challenges, we propose a Neural Tensor Network (NeurTN), which combines tensor algebra and deep neural networks to provide effective medicine recommendations. By replacing the multilinear multiplication with a neural network, NeurTN is able to characterize nonlinear dependencies among tensor data. Moreover, NeurTN incorporates various heterogeneous information to

alleviate the data sparsity issue. Instead of constructing them as feature matrices, NeurTN uses geometric neural networks to learn the embeddings from molecular graphs and target sequences, which allows to be trained end-to-end. Our data-driven model opens up opportunities to use large-scale omics data to discover drug' MoAs in pharmacological studies.

2 Preliminaries

2.1 Tensor Algebra

Tensors are multidimensional arrays that extend the concept of matrices [Kolda and Bader, 2009]. An N-order tensor is denoted as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, where the $(i_1, \ldots, i_N)$-th element of $\boldsymbol{\mathcal{X}}$ is denoted as $\boldsymbol{\mathcal{X}}_{i_1,\ldots,i_N}$. *Matricization* is the process of flattening a tensor into a matrix. The mode-n matricization of tensor$\boldsymbol{\mathcal{X}}$ is represented as $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \ldots I_{n-1} \times I_{n+1} \ldots I_N}$ and is arranging the mode-n fibers of the tensor as columns in $\mathbf{X}_{(n)}$.

N-mode product. The *n-mode product* of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ is denoted as $\boldsymbol{\mathcal{X}} \times_n \mathbf{A}$ with size $\mathbb{R}^{I_1 \cdots \times I_{n-1} \times J \times I_{n+1} \cdots \times I_N}$. Also, we have:

$$(\boldsymbol{\mathcal{X}} \times_n \mathbf{A})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \mathbf{A}_{j i_n}$$

CP tensor factorization. The CP is the most common tensor model, which assumes a compact hidden structure in the data [Kolda and Bader, 2009]. Formally, the CP model approximates the element of a third-order $\boldsymbol{\mathcal{X}}$ as:

$$\hat{\boldsymbol{\mathcal{X}}}_{ijk} = f(i, j, k | \mathbf{U}, \mathbf{V}, \mathbf{W}) = \sum_{t=1}^{r} \mathbf{U}_{it} \mathbf{V}_{jt} \mathbf{W}_{kt}, \qquad (1)$$

where $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are the factor matrices; r denotes the tensor rank. The CP model has its ease of interpretation: each rank-one component serves as a latent concept or clusters for the data. However, the CP model adopts multilinear assumption, which may be insufficient to capture more complex and non-linear feature interactions. Our work here addresses this issue by designing the predictor $f(\cdot)$ using neural networks.

2.2 Problem Definition

The medicine recommendation task can be formulated as a tensor completion problem [Chen and Li, 2019; Wang *et al.*, 2018]. To be specific, the input can be organized as a drug-target-disease tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{M \times N \times L}$, where M, N, and L denote the number of drugs, targets, and diseases, respectively. An entry $\boldsymbol{\mathcal{X}}_{ijk} = 1$ if an interaction among drug i, target j, and disease k is observed; $\boldsymbol{\mathcal{X}}_{ijk} = 0$, otherwise. Our aim is to estimate the scores of unobserved elements $\hat{\boldsymbol{\mathcal{X}}}_{ijk}$, which can be used to infer novel interactions of drug-target-disease.

2.3 Input and Embedding Layer

Our goal is to estimate a score between drug i, target j and disease k, i.e., $\hat{\boldsymbol{\mathcal{X}}}_{ijk}$. We first describe the biological features with respect to drugs, targets, and diseases.

One-hot Embeddings

Given a drug i, a target j, and a disease k, their one-hot encodings $\mathbf{a}_i \in \mathbb{R}^M$, $\mathbf{b}_j \in \mathbb{R}^N$, and $\mathbf{c}_k \in \mathbb{R}^L$ can be obtained based on their identities. We can obtain their dense embeddings via three lookup tables:

$$\widehat{\mathbf{u}}_i \leftarrow \text{lookup}(\mathbf{a}_i), \quad \widehat{\mathbf{v}}_j \leftarrow \text{lookup}(\mathbf{b}_i), \quad \widehat{\mathbf{w}}_k \leftarrow \text{lookup}(\mathbf{c}_i), \qquad (2)$$

where $\widehat{\mathbf{u}}_i \in \mathbb{R}^{d_1}$, $\widehat{\mathbf{v}}_j \in \mathbb{R}^{d_2}$, and $\widehat{\mathbf{w}}_k \in \mathbb{R}^{d_3}$ are new embeddings for drug i, target j, and disease k, respectively.

Here we also incorporate medical knowledge of drugs and targets. We leave the exploration of diseases as a future work. Instead of constructing these auxiliary information as feature matrices [Zheng *et al.*, 2013; Chen and Li, 2019], we apply geometric neural networks to learn the structural information of both drugs' chemical structures and targets' sequences.

Graph Neural Network for Molecular Graph

The chemical structure of a drug determines its physicochemical properties, further affects its pharmacological activity. In this study, we extract features of chemical structures by using Graph Neural Networks (GNN) [Duvenaud *et al.*, 2015; Gilmer *et al.*, 2017]. A chemical molecule can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes denote the atoms and edges denote the chemical bonds between atoms. The key idea behind GNN is to iteratively aggregate information from atoms' local neighbors with convolutional operator. Such information can be further propagated across the graph by stacking multiple graph convolutional layers, resulting in a learned featurization for each molecule.

Formally, given a graph $\mathcal{G}_i$ for drug i, the GNN converts the graph $\mathcal{G}_i$ into a vector representation $\mathbf{r}^i_{\mathcal{G}}$ by aggregating the representation $\mathbf{r}_v$ of each atom node v in the graph [Duvenaud *et al.*, 2015]. GNN first obtains the representation $\mathbf{r}_v^{(t)}$ by the following transition function:

$$\mathbf{m}_v^{(t+1)} = \mathbf{r}_v^{(t)} + \sum_{u \in N(v)} \mathbf{r}_u^{(t)},$$
$$\mathbf{r}_v^{(t+1)} = \sigma(\mathbf{m}_v^{(t+1)} \mathbf{H}_{(t)}^{|N(v)|}),$$

where $\mathbf{r}_v^{(t)}$ is the representation of node v in the t-th step and $\mathbf{r}_v^{(0)}$ can be initialized by its atom features [Duvenaud *et al.*, 2015]. $N(v)$ is the neighbors of v and $|N(v)|$ denotes the number of neighbors around v. $\mathbf{H}_{(t)}^{|N(v)|}$ is a weight and $\sigma(\cdot)$ is the sigmoid. A global pooling function then acquires the graph representation by combining features from all the atoms as:

$$\mathbf{r}^i_{\mathcal{G}} = \sum_{t,v} \text{softmax}(\mathbf{W}_{gnn}^{(t)} \mathbf{r}_v^{(t)}), \qquad (3)$$

here $\mathbf{W}_{gnn}^{(t)}$ is a weight and $\mathbf{r}^i_{\mathcal{G}} \in \mathbb{R}^g$ is the dense vector representations for the graph $\mathcal{G}_i$.

Convolutional Neural Network for Target Sequence

Similarly, we can obtain vector representations of target sequences by using a convolutional neural network (CNN). Here we adopt a recent developed CNN model to map target sequences to a real-valued vectors by applying hierarchical filter functions [Tsubaki *et al.*, 2018]. Given that a target sequence can be treated as a string of amino acids, it can be

transformed into a list of *words* by using n-gram feature extraction technique. Let $\mathcal{S}_j = x_1, x_2, \ldots, x_{|\mathcal{S}_j|}$ be a sequence for target j, where x_i is the i-th word after transformation and $|\mathcal{S}_j|$ is the length of sequence. All words can be then translated into randomly *word embeddings* as:

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{S}_j|-1}, \mathbf{x}_{|\mathcal{S}_j|}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is the embedding of the i-th word. Let input $\mathbf{c}_i^{(0)} = \mathbf{x}_{i:i+w-1}$ refer to $[\mathbf{x}_i, \ldots, \mathbf{x}_{i+w-1}]$, the concatenation of w contiguous embeddings. The CNN uses one filter as:

$$\mathbf{c}_i^{(1)} = \text{ReLU}(\mathbf{W}_{cnn}\mathbf{c}_i^{(0)} + \mathbf{b}_{cnn}),$$

where $\mathbf{c}_i^{(1)} \in \mathbb{R}^d$ is a hidden vector, $\mathbf{W}_{cnn} \in \mathbb{R}^{d \times dw}$ is the weight and $\mathbf{b}_{cnn}$ is the bias. Note that the filter projects a dw-dimensional input vector into a d-dimensional hidden vector. As such, one can obtain a set of hidden vector: $\mathbf{C} = \{\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}, \ldots, \mathbf{c}_{\mathcal{C}}^{(t)}\}$ by applying the filters *hierarchically* [Tsubaki *et al.*, 2018]. Finally, an average function is used to aggregate features from hidden vectors as:

$$\mathbf{r}_{\mathcal{S}}^j = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\mathbf{c}_i^{(t)}, \tag{4}$$

here $\mathbf{r}_{\mathcal{S}}^j \in \mathbb{R}^d$ is the representation for the target sequence $\mathcal{S}_j$.

The Unified Embeddings

We fuse all available features for more sophisticated embeddings by using fully connected neural layers. Given drug's features $(\widehat{\mathbf{u}}_i, \mathbf{r}_{\mathcal{G}}^i)$, target's features $(\widehat{\mathbf{v}}_j, \mathbf{r}_{\mathcal{S}}^j)$, and disease's feature $\widehat{\mathbf{w}}_k$, we can obtain new embeddings $\mathbf{u}_i, \mathbf{v}_j$, and $\mathbf{w}_k$[1] via:

$$\mathbf{u}_i \leftarrow \text{FC}(\Theta_u; \widehat{\mathbf{u}}_i \oplus \mathbf{r}_{\mathcal{G}}^i), \quad \mathbf{v}_j \leftarrow \text{FC}(\Theta_v; \widehat{\mathbf{v}}_j \oplus \mathbf{r}_{\mathcal{S}}^j), \quad \mathbf{w}_k \leftarrow \widehat{\mathbf{w}}_k \tag{5}$$

where $\oplus$ is the concatenation. In addition, by choosing the parameters Θ_u and Θ_v properly, we can reshape all embeddings with the same size, i.e., $\{\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_k\} \in \mathbb{R}^r$. As such, a nonlinear tensor model can be built upon these embeddings.

3 The Proposed Model

The proposed NeurTN contains three components: MLP, GTF, and CTN as shown in Figure 1.

3.1 Multi-Layer Perceptron

It is straightforward to feed all the features into a Multi-Layer Perceptron (MLP) [He *et al.*, 2017; Wu *et al.*, 2019], in which each hidden layer can learn nonlinear feature interactions from $\mathbf{u}_i$, $\mathbf{v}_j$, and $\mathbf{w}_k$:

$$\begin{aligned} \mathbf{z}_L &= \text{MLP}(\mathbf{u}_i \oplus \mathbf{v}_j \oplus \mathbf{w}_k), \\ \hat{\boldsymbol{\mathcal{X}}}_{ijk} &= \sigma(\mathbf{W}_L\mathbf{z}_L + \mathbf{b}_L) \end{aligned} \tag{6}$$

where $\mathbf{W}_L$ and $\mathbf{b}_L$ denote the weight and bias. We choose the ReLU$(\cdot)$ as activation function in hidden layers and the sigmoid function $\sigma(\cdot)$ as predictor function.

The MLP is capable of learning nonlinearity of the concatenated features. Nevertheless, the concatenated features, i.e., $\mathbf{u}_i \oplus \mathbf{v}_j \oplus \mathbf{w}_k$, may lose some information in the original embeddings that are useful for later interaction learning. To avoid such information loss, we further propose two triple-wise layers for learning multi-aspect features.

[1] $\mathbf{w}_k$ only contains one-hot feature since there is no auxiliary information for diseases in this study.

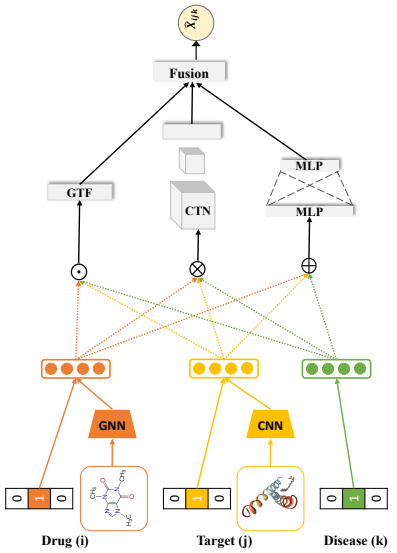

Figure 1: Overall architecture of NeurTN.

3.2 Generalized Tensor Factorization

Inspired by the success of tensor CP model in Eq. (1), we propose a Generalized Tensor Factorization (GTF) to learn nonlinear triple-wise feature interactions. Specifically, given the drug embedding $\mathbf{u}_i \in \mathbb{R}^r$, target embedding $\mathbf{v}_j \in \mathbb{R}^r$, and disease embedding $\mathbf{w}_k \in \mathbb{R}^r$, we define a novel triple-wise interaction layer as:

$$\boldsymbol{\phi}(\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_k) = \mathbf{u}_i \odot \mathbf{v}_j \odot \mathbf{w}_k \tag{7}$$

here $\odot$ denotes the element-wise product. Then, we can project the hidden vector $\boldsymbol{\phi}(\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_k)$ into an output layer:

$$\hat{\boldsymbol{\mathcal{X}}}_{ijk} = f_{out}(\mathbf{h}^T(\mathbf{u}_i \odot \mathbf{v}_j \odot \mathbf{w}_k)) \tag{8}$$

here $f_{out}(\cdot)$ and $\mathbf{h}$ denote the activation function and weights.

Relation to tensor CP model: Intuitively, if we use an identity function for f_{out} (e.g., $f_{out}(x) = x$) and enforce the weights $\mathbf{h}$ to be a uniform vector of 1, we can rewrite the Eq. (8) as $\hat{\boldsymbol{\mathcal{X}}}_{ijk} = \sum_{t=1}^{r}\mathbf{u}_{it}\mathbf{v}_{jt}\mathbf{w}_{kt}$, which exactly recovers the tensor CP model as Eq. (1). In other words, GTF extends the CP model into a nonlinear tensor machine. As neural networks have strong ability to fit the data, GTF is thus more expressive than the linear CP model. In this work, we implement the GTF layer by using the sigmoid function $f_{out}(x) = 1/(1+e^{-x})$ and learning $\mathbf{h}$ from the data.

3.3 Compressed Tensor Network

To better capture the triple-wise feature interactions among drug-target-disease, we further consider the outer product on the embeddings $\mathbf{u}_i, \mathbf{v}_j$, and $\mathbf{w}_k$:

$$\boldsymbol{\mathcal{E}} = \mathbf{u}_i \otimes \mathbf{v}_j \otimes \mathbf{w}_k \tag{9}$$

where $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{r \times r \times r}$ is a tensor feature map and $\otimes$ denotes the outer product. Our motivation for $\boldsymbol{\mathcal{E}}$ is straightforward. The map $\boldsymbol{\mathcal{E}}$ captures more signals than element-wise product in Eq. (7) since it encodes any triple-wise feature interactions. Such strategy has been widely used to

boost system performance in deep learning [He *et al.*, 2018; Cohen *et al.*, 2016]. Moreover, the feature map $\boldsymbol{\mathcal{E}}$ lies in a 3D tensor format, which allows us to design an efficient feedforward neural network by using tensor algebra latter.

The 3D tensor format of $\boldsymbol{\mathcal{E}}$ naturally suggests a CNN model to use shared parameters to perform "convolution". However, the feature $\boldsymbol{\mathcal{E}}$ from the outer product space does not have spatial locality. Thus, we cannot employ the CNN to $\boldsymbol{\mathcal{E}}$ by enforcing an unexpected local connectivity pattern. Inspired by the tensor algebra in *n-mode product* (Sec 2.1), we propose a simple but efficient Compressed Tensor Network (CTN) to perform feedforward computations. Basically, the tensor map $\boldsymbol{\mathcal{E}}$ is successively compressed by a sequence of three weight matrices along each mode with nonlinear transformation:

$$\begin{aligned}
&\boldsymbol{\mathcal{H}}_1 = \text{ReLU}(\boldsymbol{\mathcal{E}} \times_1 \mathbf{A}^{(0)} \times_2 \mathbf{B}^{(0)} \times_3 \mathbf{C}^{(0)} + \boldsymbol{\mathcal{B}}^{(0)}), ..., \\
&\boldsymbol{\mathcal{H}}_L = \text{ReLU}(\boldsymbol{\mathcal{H}}_{L-1} \times_1 \mathbf{A}^{(L-1)} \times_2 \mathbf{B}^{(L-1)} \times_3 \mathbf{C}^{(L-1)} + \boldsymbol{\mathcal{B}}^{(L-1)}), \\
&\hat{\boldsymbol{\mathcal{X}}}_{ijk} = \sigma(\mathbf{W}_o \times \text{Reshape}(\boldsymbol{\mathcal{H}}_L) + \mathbf{b}_o)
\end{aligned} \tag{10}$$

where $\boldsymbol{\mathcal{H}}_l$, $\{\mathbf{A}^{(l)}, \mathbf{B}^{(l)}, \mathbf{C}^{(l)}\}$, and $\mathcal{B}^{(l)}$ denote the hidden tensor, the weights, and bias in the l-th layer, respectively. $\text{Reshape}(\cdot)$ flattens $\boldsymbol{\mathcal{H}}_L$ into a vector. The output layer is a fully connected layer with the sigmoid function as predictor.

Dropout: Dropout [Srivastava *et al.*, 2014] is a regularization technique for neural networks to prevent overfitting. The idea of dropout is to randomly turn off neurons with probability ρ during training, and use all neurons in the testing. Here we also apply a dropout layer on the feature $\boldsymbol{\mathcal{E}}$, i.e., randomly dropping ρ percent of its elements.

3.4 The Overall Model

Joint Training. We have developed three instantiations of nonlinear tensor models: MLP, GTF, and CTN. We present our unified model NeurTN by joint learning these three modules. Formally, let $\mathbf{z}_L$, $\boldsymbol{\phi}$, and $\boldsymbol{\mathcal{H}}_L$ denote the outputs of the last hidden layers of MLP, GTF, and CTN, respectively. Then, we have:

$$\begin{aligned}
&\mathbf{F} = \mathbf{z}_L \oplus \boldsymbol{\phi} \oplus \text{Reshape}(\boldsymbol{\mathcal{H}}_L), \\
&\hat{\boldsymbol{\mathcal{X}}}_{ijk} = \sigma(\mathbf{W}_f \mathbf{F} + \mathbf{b}_f)
\end{aligned} \tag{11}$$

here we choose the sigmoid function $\sigma(\cdot)$ as final predictor.

Relation to wide&deep learning: The proposed NeurTN is similar to the well-known Wide&Deep Learning [Cheng *et al.*, 2016; He *et al.*, 2017], which has the benefits of memorization and generalization. Our GTF can be regarded as a wide component whereas the MLP and CTN can be viewed as deep components.

Model Optimization. We adopt pairwise learning methods to optimize model parameters [Bordes *et al.*, 2013]. The idea behind pairwise learning is that an observed triplet should be predicted with a higher score than an unobserved one. This can be achieved by minimizing:

$$\mathcal{L}(\Theta) = \sum_{(i,j,k) \in \mathcal{D}^+} \sum_{(i',j',k') \in \mathcal{D}^-} \max(0, 1 + f(i',j',k') - f(i,j,k)) \tag{12}$$

here $f(\cdot)$ and Θ denote our predictive function and model parameters in Eq. (11). $\mathcal{D}^+$ denotes the set of positive triplets (e.g., $\boldsymbol{\mathcal{X}}_{ijk} = 1$), and $\mathcal{D}^-$ denotes the set of negative triplets corresponding to $\mathcal{D}^+$ by sampling from unobserved elements. Following [Bordes *et al.*, 2013], for each positive training triplet (i, j, k), we randomly sample one negative training triplet (i', j', k') in the training step.

It is worth mentioning that our model is highly instructive and allows the use of deep learning techniques, such as dropout regularization, which is more effective than the kernel-based nonlinear tensor models (see Sec 4.2).

4 Experiments

4.1 Datasets and Baselines

Datasets. We obtain data from three public databases [Chen and Li, 2019; Wang *et al.*, 2018]: CTD[2], DrugBank[3], and UniProt[4]. We only focus on drugs that have DrugBank identifier for later collecting auxiliary information. As such, we obtain $436,322$ triplets of drug-target-disease, involving $1,901$ drugs, $2,514$ targets, and $2,923$ diseases. For drugs, their SMILES, a string encoding of chemical structures, are downloaded from DrugBank. These SMILES strings can be converted to molecular graphs using RDKit tool[5], which can be then fed into the GNN module. For targets, their amino acid sequences are collected from UniProt and can be used by the CNN module without any pre-processing.

Baselines. We mainly compare with tensor-based models that can learn the drug-target-disease data. (1) CP and Tucker [Kolda and Bader, 2009]: both adopt multilinear assumption. (2) nTucker [Zhe *et al.*, 2016]: a nonlinear Tucker based on Gaussian process. (3) CoSTCo [Liu *et al.*, 2019]: a recent CNN-based tensor model. (4) CMTF [Acar *et al.*, 2011]: a tensor-matrix model regarding auxiliary information as feature matrices. (5) TFAI [Narita *et al.*, 2012]: a tensor model with mode regularization. (6) AirCP [Ge *et al.*, 2016]: a tensor-matrix model with graph regularization. (7) NTF [Wu *et al.*, 2019]: a neural network with MLP. (8) Rubik [Wang *et al.*, 2015]: a tensor model with non-negativity and sparsity constraints. (9) DTD [Chen and Li, 2019]: a recent tensor-matrix model in drug discovery.

Parameter Settings. For tensor-matrix models CMTF, TEAI, AirCP, Rubik, and DTD, the feature matrices for auxiliary information are constructed using feature engineering as [Zheng *et al.*, 2013; Chen and Li, 2017], The parameter settings for all the baselines are carefully tuned to achieve optimal performance. For NeurTN, the embedding size r in Eq. (5) is searched within $[16, 32, 64, 128]$. For MLP and CTN, we both employ three hidden layers with dropout ratio $\rho = 0.3$ and each layer sequentially decreases the half size of inputs. Our models are built upon PyTorch[6] with Adam optimizer [Kingma and Ba, 2015]. We search the batch size and the learning rate within $\{128, 256, 512, 1024\}$ and $\{0.001, 0.005, 0.01, 0.05, 0.1\}$, respectively. We use grid-based search to find the best parameter settings. We tune

[2]http://ctdbase.org/downloads/
[3]https://www.drugbank.ca/
[4]https://www.uniprot.org/
[5]http://rdkit.org/
[6]https://pytorch.org/

Model	Hit@5	NDCG@5	Hit@10	NDCG@10	Hit@15	NDCG@15
CP	0.412	0.201	0.441	0.243	0.462	0.251
Tucker	0.423	0.212	0.452	0.249	0.459	0.254
nTucker	0.438	0.223	0.463	0.257	0.467	0.261
CoSTCo	0.437	0.230	0.461	0.259	0.469	0.266
MLP	0.428	0.219	0.456	0.255	0.461	0.258
GTF	0.459	0.241	0.478	0.262	0.473	0.269
CTN	0.462	0.250	0.482	0.269	0.480	0.271
NeurTN	**0.475**	**0.259**	**0.491**	**0.277**	**0.486**	**0.279**

Table 1: Results of different methods without auxiliary information.

model parameters using validation set and terminate training if the performance does not improve for 100 epochs.

Evaluation Protocols. We randomly split the dataset into 80% training, 10% validation, and 10% test sets. The validation set is used for tuning hyper-parameters and the final results are evaluated on the test set. To better construct negative test triplets, we adopt similar procedures as [Ge *et al.*, 2016; Chen and Li, 2019; Bordes *et al.*, 2013]: 1) Scenario 1 (random sample): for each positive test triplet (i, j, k), we randomly sample a negative triplet (i', j', k') such that (i', j', k') is unobserved from data, i.e., $\boldsymbol{\mathcal{X}}_{i'j'k'} = 0$; 2) Scenario 2 (sample drug mode): we corrupt the drug mode by replacing the drug i with a new drug i' so that the (i', j, k) is unobserved; 3) Scenario 3 (sample target mode): we corrupt the triplet (i, j, k) by (i, j', k) in the target mode; 4) Scenario 4 (sample disease mode): the triplet (i, j, k) is replaced with (i, j, k'). Such scenarios are meaningful in clinics. For example, given a pair of drug-target (i, j), we can find their potential indications for new disease k' [Hauser *et al.*, 2017]. In addition, we apply *filtered settings* [Bordes *et al.*, 2013] such that those test negative samples will not appear in the training step.

We use two common top-n metrics, Hit@n and NDCG@n, to evaluate recommendation performance [He *et al.*, 2017; Bordes *et al.*, 2013]. The metric of Hit@n measures if a test triplet is among the top-n ranked list, while NDCG@n is a position-aware metric which assigns larger weights on higher positions. Moreover, we use the strategy in [Bordes *et al.*, 2013] to avoid heavy computation on all triplets. For example, in Scenario 1, we randomly generate 100 negative samples (i', j', k') for each test (i, j, k). Based on the ranking of these triplets, Hit@n and NDCG@n can be computed.

4.2 Experimental Results

Pure Tensor Completion. We have proposed four variants of nonlinear tensor models: MLP (Eq. (6)), GTF (Eq. (8)), CTN (Eq. (10)), and the unified NeurTN (Eq. (11)). In this section, We first compare them with the baselines CP, Tucker, nTucker, and CoSTCo, which are pure tensor machines without any auxiliary data of drugs or targets. For fair comparison, we only use the one-hot features in Eq. (2) as inputs for the proposed models. Due to page limitation, we only show the top-n performances for Scenario 1, and similar trends can be observed under different scenarios. Table 1 shows the results in terms of Hit@n and NDCG@n.

It is observed that nonlinear tensor models consistently outperform the multilinear models. For example, nonlinear nTucker performs better than Tucker; GTF outperforms the CP model and gains average improvements of 7.39% on Hit@n and 11.63% on NDCG@n. These improvements are statistically significant and mainly stem from the powerful representation of nonlinear models. Second, the performance of CoSTCo is limited for drug-target-disease data that do not have local connectivity. As such, CoSTCo may be insufficient to capture nonlinear patterns for biological data using a CNN. Third, the performances of CTN are better than GTF, implying a good representations of a deeper neural network. Finally, NeurTN achieves the best performances, presumably this owes to the the benefits of its joint learning.

Drug (DrugBank)	Target (UniProt)
Fructose	Bis(5'-adenosyl)-triphosphatase
Nicotine	Neuronal acetylcholine receptor subunit alpha-2
Fructose	ATP synthase subunit beta, mitochondrial
Glucosamine	Matrix metalloproteinase-9
Atorvastatin	Histone deacetylase 2
Cannabidiol	Histamine H1 receptor
Calcitriol	Solute carrier family 12 member 3
Cyclophosphamide	Protein kinase C beta type
Bezafibrate	Nuclear receptor subfamily 1 group I member 2
Copper	Insulin receptor

Table 2: Top 10 novel drug-target pairs for diabetes' disease.

Tensor Completion with Auxiliary Information. Now we compare the overall performance of NeurTN with the baselines with auxiliary information. The results of CP, Tucker, nTucker, and CoSTCo are omitted due to their inferior performances without auxiliary data. Figure 2 shows the performance of Top-n performances, where $n = [5, 10, 15]$.

From the figures, we can observe that our proposed NeurTN outperforms coupled tensor-matrix factorization methods in all scenarios. The superior performance of NeurTN mainly benefits from its deep neural networks. It is intuitive that neural networks would have stronger ability to fit the data, while the multilinear assumptions in coupled tensor-matrix factorizations do not. More importantly, NeurTN utilizes geometric neural networks GNN and CNN, which can learn features from molecular graphs and protein sequences in the training process. Such end-to-end representation learning can potentially obtain more interpretable data-driven features instead of predefining hand-crafted feature matrices in coupled tensor-matrix factorizations. Given these encouraging results, we use our model to predict novel interactions in the diabetes-specific pathway.

Case Study on Diabetes. In clinics, given a disease, it is critical to know which drugs can treat this disease as well as the targets involving in the disease-specific pathway. The pairs of (drug, target) related to a special disease are important in personalized treatments. Here we use NeurTN to predict the novel (drug, target) pairs related to diabetes's metabolic pathway. The top-10 novel pairs of (drug, target) corresponding to diabetes are listed in Table 2. The predictive results can be then evaluated by domain experts to see whether such recommendations are clinically meaningful. They can be further validated *in vivo* experiments, such as animal studies and clinical trials. In summary, the proposed model provides a systematic approach to narrow down the search space for further clinical trials in drug discovery.

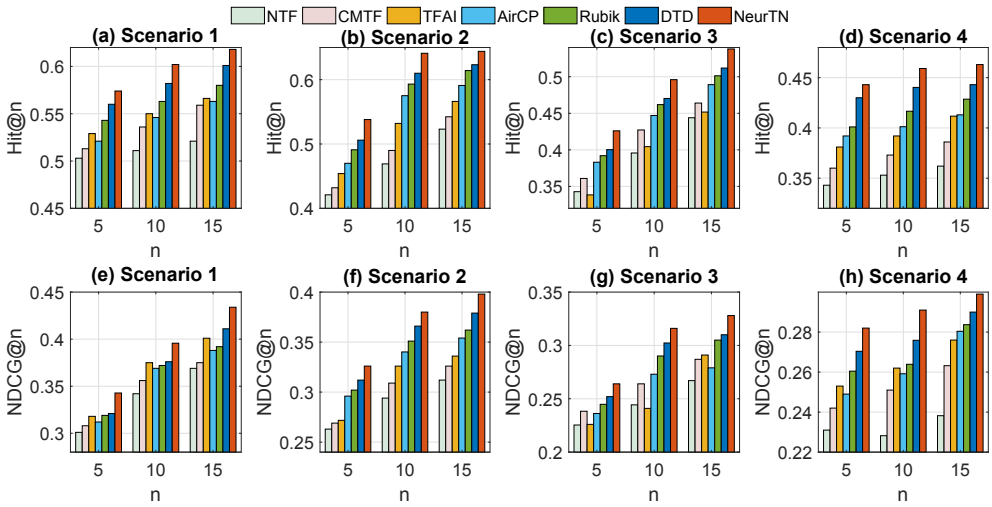

Figure 2: Evaluation of top-n performance for different scenarios in terms of Hit@n (a-d) NDCG@n (e-h).

4.3 Ablation Study

We perform some ablation studies to investigate the impact of each module in NeurTN. Table 3 shows the performance of our default method and its variants in the case of Scenario 1.

Our results are summarized as follows: (1) Remove MLP: Without MLP layers, we find that the performance is slightly worse. Although the black-box nature of MLP, its hierarchical structure is still helpful to learn more complex interactions; (2) Remove GTF: Not surprisingly, the results are worse than the default method, which suggests that the GTF can capture triple-wise feature interactions in a nonlinear fashion; (3) Remove CTN: This variant substantially decreases the overall performance with a large margin, verifying the effectiveness the CTN in capturing the useful feature interactions from outer product feature space; (4) Remove GNN: The chemical structure of a drug determines its pharmacological activity. Removing GNN thus decreases the overall performance; (5) Remove CNN: Similar, amino acid sequences determine the therapeutic function of targets. Deleting the CNN module thus hurts the final performance.

Embedding size of NeurTN: The embedding size r in Eq. (5) affects the representation ability of NeurTN. We vary r within $[16, 32, 64, 128]$. As shown in Figure 3(a), NeurTN benefits from a large embedding size in Scenario 1. Results on other scenarios have similar trends and are omitted.

Layers of NeurTN: We also conduct experiments to investigate the impact of the number of layers in NeurTN. We vary the number of layers in the NeurTN within $L = [2, 3, 4, 5]$.

Architecture	Hit@5	NDCG@5	Hit@10	NDCG@10
(0) Default	0.574	0.343	0.602	0.396
(1) Remove MLP	0.558	0.337	0.571	0.380
(2) Remove GTF	0.533↓	0.335	0.553↓	0.371↓
(3) Remove CTN	0.527↓	0.329↓	0.544↓	0.362↓
(4) Remove GNN	0.515↓	0.317↓	0.526↓	0.341 ↓
(5) Remove CNN	0.519↓	0.324↓	0.530↓	0.346 ↓

Table 3: Ablation analysis on our variant models. '↓' means a severe performance drop.

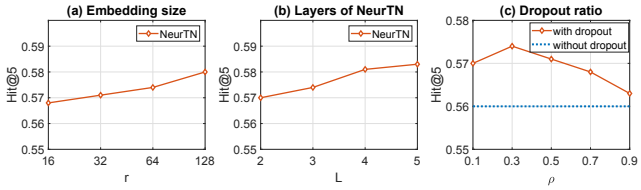

Figure 3: (a) The impact of embedding size r. (b) The impact of the number of layers L. (c) The impact of dropout ratio ρ.

As shown in Figure 3(b), stacking more layers generally improves the system performance, which demonstrates the usage of more neural layers is able to model complex drug-target-disease interactions.

Dropout regularization: Figure 3(c) shows the performances of NeurTN $w.r.t.$ dropout ratio. Our results show that dropout offers better performance. Specifically, using a dropout ratio $\rho \approx 0.3$ achieves an optimal accuracy.

5 Conclusion

A critical step in personalized medicine is to understand drugs' MoAs by exploring the biological interactions among drugs, targets, and diseases in human metabolic systems. In this study, we present a novel NeurTN, which seamlessly combines the tensor algebra and deep neural network to capture the nonlinear relationships among health data. In the future, we aim to incorporate the auxiliary information of diseases under the NeurTN.

Acknowledgments

This work has been supported in part by NSF CCF1815139.

References

[Acar *et al.*, 2011] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2013.

[Capuzzi *et al.*, 2018] Stephen J Capuzzi, Thomas E Thornton, Kammy Liu, Nancy Baker, Wai In Lam, Colin P O'Banion, Eugene N Muratov, Diane Pozefsky, and Alexander Tropsha. Chemotext: A publicly available web server for mining drug–target–disease relationships in pubmed. *Journal of chemical information and modeling*, 58(2):212–218, 2018.

[Chen and Li, 2017] Huiyuan Chen and Jing Li. A flexible and robust multi-source learning algorithm for drug repositioning. In *ACM-BCB*, 2017.

[Chen and Li, 2018] Huiyuan Chen and Jing Li. Drugcom: Synergistic discovery of drug combinations using tensor decomposition. In *ICDM*, 2018.

[Chen and Li, 2019] Huiyuan Chen and Jing Li. Modeling relational drug-target-disease interactions via tensor factorization with multiple web sources. In *WWW*, 2019.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, 2016.

[Cohen *et al.*, 2016] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *COLT*, 2016.

[Duvenaud *et al.*, 2015] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015.

[Ezzat *et al.*, 2018] Ali Ezzat, Min Wu, Xiao-Li Li, and Chee-Keong Kwoh. Computational prediction of drug–target interactions using chemogenomic approaches: an empirical survey. *Briefings in bioinformatics*, 2018.

[Ge *et al.*, 2016] Hancheng Ge, James Caverlee, Nan Zhang, and Anna Squicciarini. Uncovering the spatio-temporal dynamics of memes in the presence of incomplete information. In *CIKM*, 2016.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[Hauser *et al.*, 2017] Alexander S Hauser, Misty M Attwood, Mathias Rask-Andersen, Helgi B Schiöth, and David E Gloriam. Trends in gpcr drug discovery: new agents, targets and indications. *Nature Reviews Drug Discovery*, 16(12):829, 2017.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

[He *et al.*, 2018] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. In *IJCAI*, 2018.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kolda and Bader, 2009] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[Li *et al.*, 2015] Jiao Li, Si Zheng, Bin Chen, Atul J Butte, S Joshua Swamidass, and Zhiyong Lu. A survey of current trends in computational drug repositioning. *Briefings in bioinformatics*, 17(1):2–12, 2015.

[Liu *et al.*, 2019] Hanpeng Liu, Yaguang Li, Michael Tsang, and Yan Liu. Costco: A neural tensor completion model for sparse tensors. In *KDD*, 2019.

[Narita *et al.*, 2012] Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery*, 25(2):298–324, 2012.

[Shan *et al.*, 2016] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *KDD*, 2016.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *JMLR*, 2014.

[Tsubaki *et al.*, 2018] Masashi Tsubaki, Kentaro Tomii, and Jun Sese. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35(2):309–318, 2018.

[Wang *et al.*, 2015] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel Kho, You Chen, Bradley A Malin, and Jimeng Sun. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *KDD*, 2015.

[Wang *et al.*, 2018] Ran Wang, Shuai Li, Man Hon Wong, and Kwong Sak Leung. Drug-protein-disease association prediction and drug repositioning based on tensor decomposition. In *BIBM*, 2018.

[Wu *et al.*, 2019] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh V Chawla. Neural tensor factorization for temporal interaction learning. In *WSDM*, 2019.

[Xu *et al.*, 2012] Zenglin Xu, Feng Yan, and Yuan Qi. Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. In *ICML*, 2012.

[Zhe *et al.*, 2016] Shandian Zhe, Kai Zhang, Pengyuan Wang, Kuang-chih Lee, Zenglin Xu, Yuan Qi, and Zoubin Ghahramani. Distributed flexible nonlinear tensor factorization. In *NeurIPS*, 2016.

[Zheng *et al.*, 2013] Xiaodong Zheng, Hao Ding, Hiroshi Mamitsuka, and Shanfeng Zhu. Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In *KDD*, 2013.