Highlights

ITNR: Inversion Transformer-based Neural Ranking for Cancer Drug Recommendations

Shahabeddin Sotudian, Ioannis Ch. Paschalidis

- The proposed framework is a transformer-based model to predict drug responses using RNAseq gene expression profile, drug descriptors and drug fingerprints.
- ITNR utilizes a Context-Aware-Transformer architecture as its scoring function that ensures the modeling of inter-item dependencies.
- We introduced a novel loss function using the concept of Inversion and Approximate Permutation matrices.
- Our computational results indicated that our method leads to substantially improved performance when compared to the baseline methods across all performance metrics, which can lead to selecting highly effective personalized treatment.

ITNR: Inversion Transformer-based Neural Ranking for Cancer Drug Recommendations

Shahabeddin Sotudian[a], Ioannis Ch. Paschalidis[a,b,*]

[a]Department of Electrical and Computer Engineering, Division of Systems Engineering, Boston University, Boston, MA, USA
[b]Department of Biomedical Engineering, and Faculty of Computing and Data Sciences, Boston University, Boston, MA, USA

ARTICLE INFO

Keywords:
Learning-to-rank
Transformers
Inversion
Drug Response Prediction
Approximate Permutation

ABSTRACT

Personalized drug response prediction is an approach for tailoring effective therapeutic strategies for patients based on their tumors' genomic characterization. While machine learning methods are widely employed in the literature, they often struggle to capture drug-cell line relations across various cell lines. In addressing this challenge, our study introduces a novel listwise Learning-to-Rank (LTR) model named Inversion Transformer-based Neural Ranking (ITNR). ITNR utilizes genomic features and a transformer architecture to decipher functional relationships and construct models that can predict patient-specific drug responses. Our experiments were conducted on three major drug response data sets, showing that ITNR reliably and consistently outperforms state-of-the-art LTR models.

1. Introduction

Conventional Machine Learning (ML) algorithms are generally designed to minimize regression or classification errors [1, 2]. Many real-world applications, on the other hand, deemphasize prediction accuracy and prioritize the correct ordering among all the instances [3, 4]. The aim of ***Learning-to-rank (LTR)*** methods is to apply ML techniques to solve ranking problems and predict the optimal ordering among the instances according to their degrees of relevance, importance, or preference as defined in the specific application.

Ranking plays a central role in a wide variety of applications, including document retrieval, online advertisement, drug discovery, machine translation, feature selection, document summarization, definition search, question answering, and recommendation systems, among others [5, 6]. In these applications, it is highly desirable to design a model that places relevant/important items at the top of the ranking list. In this work, and without loss of generality, we focus on cancer ***Drug Response Prediction (DRP)*** where the goal is to prescribe an optimal therapeutic option for each patient based on their cancer's unique molecular fingerprints (i.e., the goal of "precision oncology"). Not every patient responds to medical treatment in the same way. Effectiveness of a medication is influenced by various factors including physiological, pathological, environmental, and genetic factors [7]. Since in-vitro experiments are exceedingly costly and time-consuming, DRP algorithms could serve as promising strategies for the accurate prediction of optimal drug therapies based upon the personalized molecular profiles of patient tumors [8].

In the current work, we seek to use an under-explored approach for drug response prediction problems, namely ***transformer*** models. The application of transformer-based techniques in LTR signifies the high capability of these models in various applications [9]. Equipped with this perspective, we make the following contributions. We developed our LTR framework using a context-aware scoring function and an inversion-based loss function. Unlike the majority of LTR models whose scoring functions score items separately, transformer models (i.e., a popular self-attention-based neural machine translation architecture) allow for the modeling of inter-item dependencies. We adopt the Context-Aware-Transformer [9] which is a special case of the encoder part of the transformer. Additionally, this architecture takes into account the inter-dependency of scores between items in the computation of items' scores. We also proposed a loss function using the concept of *Inversion* and *Approximate Permutation* matrices.

The remainder of this paper is organized as follows: Section 2 describes the related works. Section 3 outlines our method, presents the data we used, and discusses model training and performance evaluation. Section 4 presents our main results and compares our method to alternatives. Section 5 discusses the results and draws some conclusions.

Notational conventions: We use boldfaced lowercase letters to denote vectors, ordinary lowercase letters to denote scalars, boldfaced uppercase letters to denote matrices, and calligraphic capital letters to denote sets. All vectors are column vectors. For space saving reasons, we write $\mathbf{x}$ to denote the column vector $(x_1, \ldots, x_{\dim(\mathbf{x})})$, where $\dim(\mathbf{x})$ is the dimension of $\mathbf{x}$. We use prime to denote the transpose, and $[\![N]\!]$ for the set $\{1, \ldots, N\}$ for any integer N.

2. Related Works

The confluence of efficient computational tools and a significant number of samples has ushered in a new generation of ML models for drug recommendation. The literature offers a variety of traditional approaches from Support Vector Machines (SVMs); Principal Component Regression; Ridge, LASSO, and Elastic Net Regressions; to more advanced methods such as multiple-output [3], multiple-kernel [10], and multiple-task learning [11] techniques. As

*Corresponding author
sotudian@bu.edu (S. Sotudian); yannisp@bu.edu (.I.Ch. Paschalidis)
ORCID(s):

for traditional methods, comprehensive comparative studies [12, 8] demonstrated that Elastic Net or Ridge regression-based models will most likely yield the most accurate predictors. On the other hand, deep learning (DL) models have demonstrated their superiority in capturing the non-linear and complex relationships of biological data better than the traditional algorithms [13]. Representative DL-based algorithms include [14, 15, 9]. For instance, Ahmadian et al. [16] developed DSAPSO, a recommender system using deep sparse autoencoder and particle swarm optimization. It tackles the challenge of applying deep neural networks to diverse resources and integrates results effectively. DSAPSO learns latent features from ratings, trust relationships, and tags, optimizing weights with particle swarm optimization. Experimental results on two datasets show its superiority over existing recommender algorithms. DHSIRS [17] is a novel deep hybrid recommender system addressing the limitations of traditional models. DHSIRS combines side information and interaction matrix using a multilayer perceptron neural network, efficiently learning high-order non-linear relations between users and items. Experimental results show a 4.18% improvement over the second-best model, addressing challenges like data sparsity and capturing semantic relationships for more accurate item recommendations. In another study, Yengikand et al. [18] focused on deep learning-based collaborative filtering methods in recommendation systems. The objective was to efficiently map users and items to a common representation space using techniques such as multilayer perceptron (MLP) and stacked autoencoder networks. These methods aimed to overcome the limited expressiveness of the Dot product function in describing the diverse impacts of latent factors. The proposed recommender system, Deep-MSR, employed MLP and stacked autoencoder networks to extract latent factors from the user-item interaction matrix. It enhanced rating predictions by incorporating different weights to account for the varying importance of latent factors.

For more information, please refer to [19, 13] and [7] where a comprehensive analysis of DRP models and other related topics such as data integration, feature selection, experimental settings, combination therapy, and more have been provided.

3. Materials and Methods

3.1. Problem Formulation

Data in a drug ranking problem consist of a set of triples (cell line, drug, drug response score). A cell line-drug pair is represented by a feature vector. Our ultimate goal is to select the most effective drugs from a set of drugs based on their response. We characterize a drug ranking data set with tuples $\{(\mathbf{X}^q, \boldsymbol{\theta}^q)\}_{q=1}^{T}$ where $q \in [\![T]\!]$ indexes cell lines, and $\mathbf{X}^q$ and $\boldsymbol{\theta}^q$ represent the list of cell line-drug pairs and corresponding drug response scores, respectively. For the q-th cell line, we have n_q drugs. $\mathbf{X}^q \in \mathbb{R}^{n_q \times (N_D + N_C)}$ has rows $(\mathbf{x}_1^q, \ldots, \mathbf{x}_{n_q}^q)$, each of which is a $(N_D + N_C)$-dimensional cell line-drug vector, formed as the concatenation of an (N_C)-dimensional cell line feature vector (i.e., gene expression for cell lines) and an (N_D)-dimensional drug feature vector. The vector $\boldsymbol{\theta}^q = (\theta_1^q, \ldots, \theta_{n_q}^q) \in \mathbb{R}_+^{n_q}$ contains the corresponding ground-truth drug response scores. The drug response scores are in [0, 1] and a higher score (in our data) implies a more effective drug. Our goal is to learn a scoring function $f : \mathbb{R}^{(N_D+N_C)} \to \mathbb{R}$ from a training data set with given drug response scores that minimizes the empirical loss:

$$\hat{\mathcal{L}}(f) \triangleq \frac{1}{\sum_q n_q} \sum_{q=1}^{T} \sum_{d=1}^{n_q} \ell\big(\theta_d^q, f(\mathbf{x}_d^q)\big), \tag{1}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function. For a new cell line-drug matrix $\mathbf{X}^t \in \mathbb{R}^{n_t \times (N_D+N_C)}$, we can obtain the predicted ranking list by ranking the rows in $\mathbf{X}^t$ based on their inferred ranking scores $\hat{\boldsymbol{\theta}}^t = (f(\mathbf{x}_1^t), \ldots, f(\mathbf{x}_{n_t}^t))$. Two pivotal elements of an LTR framework are a scoring function and a loss function. In subsequent subsections, we describe the construction of our LTR framework using a context-aware scoring function and an inversion-based loss function.

3.2. Multi-Headed Self-Attention Scoring Function

Most algorithms in the DRP literature are trained to optimize a loss function that may not capture the interactions among drugs [8]. Moreover, they score them individually at inference-time without regard to any *mutual influences* among the drugs. Given that context-aware models based on transformers [9] have been successfully used for document retrieval, we adopt a similar approach of applying a context-aware ranker for our scoring function. The transformer architecture proposed by [9] plays the role of the ***Multi-Headed Self-Attention (MHSA)*** scoring function in our LTR framework. The self-attention mechanism of this architecture aims to handle long-range and inter-item dependencies. Since the MHSA scoring function scores a drug by considering all other drugs applicable to a cell line, it fully captures the interactions among drugs.

We consider the list's items as our tokens where item features are token embeddings. We feed these embeddings to an *Encoder Layer*. The attention mechanism is the core of the encoder layer to catalyze learning the higher-order representations of items in the list. Mapping query ($\mathbf{q}_i$), key ($\mathbf{k}_i$), and value ($\mathbf{v}_i$) vectors to a higher level representation by taking a weighted sum of the values over all items is the essence of the self-attention mechanism. We use the *Scaled Dot-Product* form to compute attention as follows:

$$\Psi(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}'}{\sqrt{d_m}}\right)\mathbf{V}, \tag{2}$$

where $\mathbf{Q}$ is a query matrix that contains all items (queries) in the list; $\mathbf{K}$ and $\mathbf{V}$ are the key and value matrices, respectively. To empower the model to leverage the order of the input tokens, we use fixed positional encodings for the input

embeddings as follows:

$$\begin{aligned}\mathbf{\Omega}_{(p,2i)} &= \sin(p/10000^{(2i/d_m)}),\\ \mathbf{\Omega}_{(p,2i+1)} &= \cos(p/10000^{(2i/d_m)}),\end{aligned} \tag{3}$$

where p refers to the position, and i is the dimension. Here, $\mathbf{\Omega}$ is a matrix and positional encoding is a system to encode each position into a vector. For instance, given that the model uses 256 dimensions for positional encoding ($d_m = 256$), we represent each element of the feature vector (i.e., token) as a 256-dimensional vector. In the display above, p is an integer from 0 to a pre-defined maximum number of tokens minus 1. Since we have 256 dimensions, we can define 128 pairs of sine and cosine values. Accordingly, the value of i goes from 0 to 127. By ensembling multiple attention modules (a.k.a Multi-Head Attention (MHA)), we can improve the ability of the model to learn representations from various subspaces of the input data. MHA can be expressed as:

$$\begin{aligned}&\text{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}(\mathbf{\Lambda}_1,\mathbf{\Lambda}_2,...,\mathbf{\Lambda}_r)\mathbf{W^O},\\ &\mathbf{\Lambda}_i = \Psi(\mathbf{QW}_i^{\mathbf{Q}},\mathbf{KW}_i^{\mathbf{K}},\mathbf{VW}_i^{\mathbf{V}}),\end{aligned} \tag{4}$$

where in the above equations, each head $\mathbf{\Lambda}_i$ (out of r heads) refers to the i-th attention mechanism of Equation (2), and $\mathbf{W}_i^{\mathbf{Q}} \in \mathbb{R}^{d_m \times d_q}$, $\mathbf{W}_i^{\mathbf{K}} \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^{\mathbf{V}} \in \mathbb{R}^{d_m \times d_v}$, and $\mathbf{W^O} \in \mathbb{R}^{rd_v \times d_m}$ are learnable matrices. Moreover, r refers to the number of parallel attention layers or heads. Here, Concat$(\cdot)$ represents a concatenation operation. It is extremely advantageous to perform the self-attention operation several times and concatenate the outputs. The main problem is the growing size of the resulting output vector. This can be solved by linearly projecting the matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ to d_q, d_k, and d_v-dimensional spaces r times, respectively. Note that we typically use $d_q = d_k = d_v = d_m/r$. We refer the interested reader to [20] for more information.

In our transformer model, we use a more complex architecture by stacking multiple encoder blocks. The main components of an encoder block including an MHA layer with a skip connection, layer normalization, time-distributed feed-forward layer, and dropout layer can be seen in Figure 1. For more information regarding these components, please refer to [20]. Specifically, the encoder part of the Transformer includes N encoder blocks, H heads, and hidden dimension d_h. First, a shared fully connected (FC) input layer of size d_{fc} is applied to each item. Then, we feed hidden representations to the encoder part of the transformer. Our final scoring model can be achieved by iteratively stacking multiple encoder blocks where the output of each block is fed into the next one. The final model can be expressed as follows:

$$\mathbf{f}(\mathbf{x}) = \rho(\Pi_1(\dots\Pi_N(\rho(\mathbf{x})))), \tag{5}$$

where $\rho(\cdot)$ represents a projection onto a fully-connected layer and $\Pi_i(\cdot)$ refers to a single encoder block. We define a single encoder block $\Pi_i(\cdot)$ as:

$$\begin{aligned}&\Pi_i(\mathbf{x}) = N(\mathbf{z} + D(\rho(\mathbf{z}))),\\ &\mathbf{z} = N(\mathbf{x} + D(\alpha(\mathbf{x})))\end{aligned} \tag{6}$$

where $\alpha(\cdot)$ is the MHA module, $N(\cdot)$ refers to the layer normalisation, and $D(\cdot)$ is the dropout layer. Eventually, a score for each item is calculated using a shared fully-connected layer. Figure 1 provides a schematic overview of our LTR framework.

3.3. Inv-Rank Loss Function

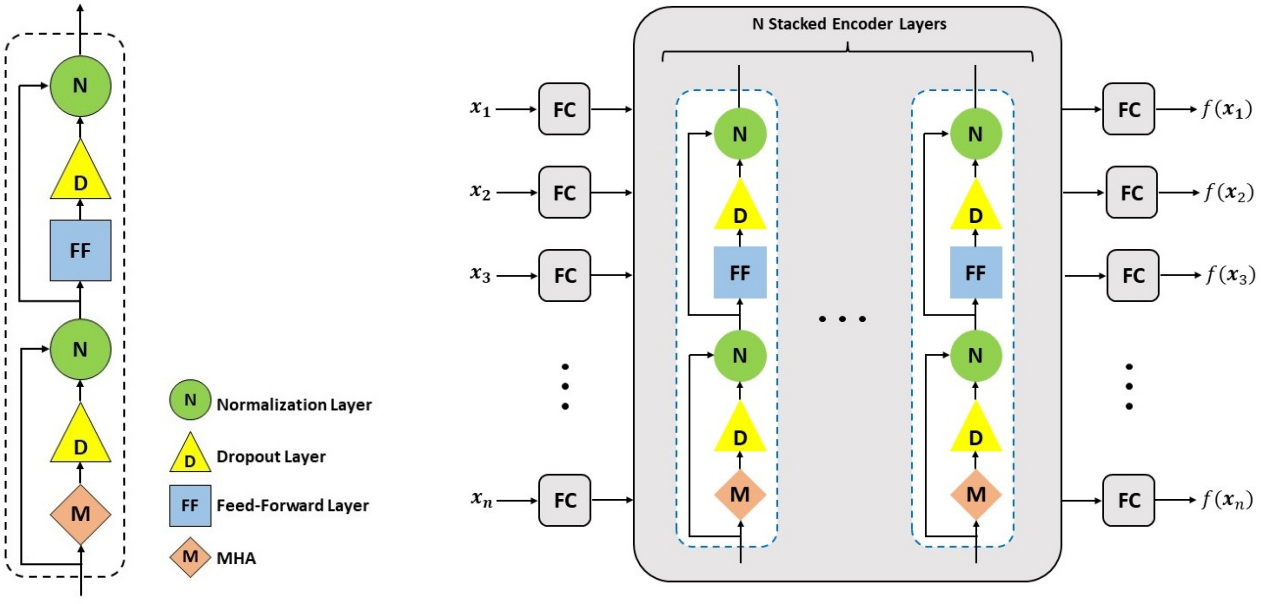
In the previous subsection, we presented the scoring function of our ranking framework. The architecture of this model allows the networks to exploit local features. More importantly, the final score for each item will be calculated by considering all other items on the list. Now, we can use the scores and the ground truth labels to optimize any desired ranking loss.

It has been demonstrated that the sorting operator can be approximated by the induced permutation matrix, $\boldsymbol{P}_{sort(\mathbf{s})}$ [9]. We use the concept of *permutation matrices* to define our proposed loss function. Permutation matrices are both **doubly-stochastic** (i.e., a square matrix with entries in [0, 1] where every row and column sum to one) and **unimodal** (i.e., a square matrix with entries in [0, 1] where each row sums to one, but also has the constraint that the maximizing entry in every row should have a unique column index) [21, 9]. Assume $\mathbf{A_s}$ represents the matrix of absolute pairwise score differences of $\mathbf{s}$ with the i,j-th element given by $\mathbf{A_s}[i,j] = |s_i - s_j|$. Grover et al. [21] proposed a continuous relaxation of the permutation matrix $\hat{P}_{sort(\mathbf{s})}$ in the space of unimodal row-stochastic matrices. The i-th row of $\hat{P}_{sort(\mathbf{s})}$ can be computed as follows:

$$\hat{P}_{sort(\mathbf{s})}[i,:](\tau) = \text{softmax}[((n+1-2i)\mathbf{s} - \mathbf{A_s}\mathbf{1}/\tau], \tag{7}$$

where $\mathbf{1}$ is an all-one vector, n is the number of items in a list, and τ behaves like a temperature knob that controls the degree of approximation and the variance of the gradients (i.e., lower τ leads to better approximation and higher variance). Moreover, softmax is a function that scales the values in the list and transforms them into values between 0 and 1 such that all values in the returned list sum to 1 (i.e., they can be interpreted as probabilities). Essentially, if we left-multiply a column vector of scores by its $\hat{P}_{sort(\mathbf{s})}$, we can achieve the approximated sorted list [22]. Note that we perform Sinkhorn scaling [23] to obtain *doubly stochastic* permutation matrices. The Sinkhorn scaling method normalizes all rows and columns; this process is repeated until convergence is achieved (i.e., 30 iterations or the greatest gap between the sum of a row or column and one must be maintained below 10^{-6}, whichever occurs first). Assume we use the MHSA scoring function to calculate scores $\mathbf{s}^q = f(\mathbf{X}^q)$ for the q-th cell line $\mathbf{X}^q$. We also have the ground truth labels $\boldsymbol{\theta}^q$ for the cell line. We define true and predicted *Weighted Approximate Permutation (WAP) matrices* using Equation (7) as:

- $\mathbf{S}_P = \hat{P}_{sort(\mathbf{s}^q)} \otimes \mathbf{w}(\boldsymbol{\theta}^q)$: predicted WAP matrix induced by the scores $\mathbf{s}^q$,
- $\mathbf{S}_T = \hat{P}_{sort(\boldsymbol{\theta}^q)} \otimes \mathbf{w}(\boldsymbol{\theta}^q)$: true WAP matrix induced by the ground truth labels $\boldsymbol{\theta}^q$,



(a)Single Encoder Layer (b)Ranking Framework

Figure 1: Architecture for the context-aware ranking model. After feeding an item's features into a fully connected layer, we pass its output through N encoders blocks. The essential elements of an encoder block comprise a Multi-Head Attention (MHA) layer with a skip connection, layer normalization, a time-distributed feed-forward layer, and a dropout layer. Finally, another fully-connected layer is used to calculate scores.

where $\mathbf{w}(\theta^q) = (g(\theta^q_1), g(\theta^q_2), \dots, g(\theta^q_{n_q}))$ is the drug importance vector, and $g(x) = 2^x - 1$ is a famous LTR gain function. Here, $\otimes$ indicates the multiplication of columns of $\hat{P}_{sort(\cdot)}$ by elements of the vector $\mathbf{w}(\cdot)$ (i.e., multiply the first column by the first element, second column by the second element and so on). The WAP matrix (i.e., $\mathbf{S}_P$ and $\mathbf{S}_T$) captures the relative position of a drug in a list. Each column of this matrix refers to a drug and the non-zero element of a column represents the predicted/true position of a drug in a ranking list. Note that if the ground truth labels of multiple drugs in a list are the same, then we have multiple ideal positions for each of them. Thus, multiple elements of those columns (i.e., drugs) will be non-zero which shows the potential predicted/true positions of those drugs in the list. We use $\mathbf{w}$ to force our model to focus on more sensitive drugs (i.e., it gives higher weights to more sensitive drugs) rather than insensitive drugs. Consequently, the model pushes sensitive drugs to the top of the ranking list.

We use the concept of *Inversion* to define our loss function. Inversion can be defined as a pair of elements that are out of their correct order in a permutation. Let π be a permutation. If $i < j$ and $\pi(i) > \pi(j)$, either the pair of places (i, j) or the pair of elements $(\pi(i), \pi(j))$ is called an inversion of π. We define $\mathbf{\Theta}$ to capture inversions in a permutation matrix as follows:

$$\mathbf{\Theta}(\mathbf{s}^q, \theta^q) = \left(\mathbf{I}^{'} \otimes \boldsymbol{\beta}\right)^{'}, \tag{8}$$

where $\mathbf{I} = \mathbf{S}_P\mathbf{S}'_T - \mathbf{S}_T\mathbf{S}'_T$ and $\boldsymbol{\beta} = (1, 2, 3, \dots, n_q)$. Precisely, $\mathbf{I}$ measures the deviation between predicted and true WAPs and penalizes the model for any inversion. The rows of a WAP matrix represent ranks. There exist various degrees of inversion. For instance, there is a big difference between $(\pi(1), \pi(3))$ and $(\pi(1), \pi(7))$. Our model should penalize $(\pi(1), \pi(7))$ more than $(\pi(1), \pi(3))$. To that end, we multiply the rows of the $\mathbf{I}$ matrix by the elements of $\boldsymbol{\beta}$. Therefore, our model places greater emphasis on high inversions. Eventually, we define the **Inv-Rank** loss of all cell lines in our training data set as follows:

$$\hat{\mathcal{L}}(f) \triangleq \frac{1}{\sum_q n_q} \sum_{q=1}^{T} \sum_{d=1}^{n_q} |\mathbf{\Theta}^d(\mathbf{s}^q, \theta^q)^{'} \mathbf{1}|, \tag{9}$$

where $\mathbf{\Theta}^d(\mathbf{s}^q, \theta^q)$ is the d-th column of $\mathbf{\Theta}(\mathbf{s}^q, \theta^q)$. Since $\mathbf{S}_P$ is differentiable with respect to the elements of $\mathbf{s}$ [21], it is easy to show that the proposed loss function is a differentiable function of scores. Therefore, the SGD method can be used to optimize the loss function.

3.4. Data sets and Pre-processing Steps

The current study focused on single-drug response prediction and we designed, trained, and evaluated all models using the cell line data and drug sensitivity data from the Predictive Oncology Model & Data Clearinghouse hosted at the National Cancer Institute [7]. Specifically, we utilized the Pilot 1 Cancer Drug Response Prediction Dataset [7],

which is a publicly available dataset representing a community effort to establish a benchmark for predicting tumor dose response across multiple data sources[1].

We used three main drug-cell line datasets from the Pilot 1 Cancer Drug Response Prediction Dataset [7], namely the *Cancer Cell Line Encyclopedia (CCLE)*, the *Genomics of Drug Sensitivity in Cancer (GDSC)*, and the *Genentech Cell Line Screening Initiative (gCSI)* studies. Specifically, Pilot 1 represents a cell line using its RNAseq gene expression profile [24] and employs drug descriptors and drug fingerprints to characterize a drug. Pilot 1 also includes the Area Under the drug response Curve (AUC) to quantify drug sensitivity (i.e., drug response). AUC $\in [0, 1]$ can be compared across studies, and a lower value indicates higher drug sensitivity. We adjusted drug sensitivities so that a higher value indicates a more effective drug (i.e., one minus AUC). The existence of low-quality data not only undermines the credibility and reproducibility of research findings but also impedes the effectiveness of machine learning algorithms employed in medical studies [25, 26]. Hence, meticulous preprocessing steps, including the elimination of incomplete, inconsistent, noisy, and redundant samples, are imperative for optimizing the performance of machine learning algorithms employed in medical research and diagnosis [27]. The details of the datasets and the preprocessing steps can be found in [7].

To follow the best practice of LTR [6], the continuous drug responses were converted to graded ones. Accordingly, we classified drugs into three categories 0 (i.e., "insensitive"), 1 (i.e., "sensitive"), and 2 (i.e., "highly sensitive"). For the k-th cell line, let P_{80} and P_{90} denote the 80-th and 90-th percentiles of its drug response values $\{r_{1k}, r_{2k}, \dots, r_{N_D k}\}$, respectively. Then, the drug relevance score for the i-th drug, $\hat{r}_{ik}$ $(i = 1, \dots, N_D)$, can be computed as:

$$\hat{r}_{ik} = \begin{cases} 2, & \text{if } r_{ik} \geq P_{90}, \\ 1, & \text{if } P_{80} \leq r_{ik} < P_{90}, \\ 0, & \text{otherwise.} \end{cases}$$

We also performed several pre-processing steps to reduce the training complexity and improve the overall performance. We standardize the features by subtracting the mean and scaling to unit variance [28, 29]. Originally, RNAseq gene expressions are represented by approximately 17,000 features. Several studies [30] demonstrated that the LINCS1000 gene set [31] can outperform or achieve similar performance compared to any other superset of LINCS1000. Therefore, we only included LINCS1000 genes in our analysis. Moreover, the data sets include 3838 molecular descriptors and 1024 path fingerprint features to represent a drug.

The datasets contain 3838 molecular descriptors and 1024 path fingerprint features representing drugs. To reduce the dimensionality of our data, we used a regression model for feature selection [32]. With drug-cell line vectors and response values as variables, the model identified important features based on regression coefficients. We focused on 1000 cell line features and 4862 drug features, selecting the top 500 genes and 500 drug features with the highest coefficient values. This yielded 1000 selected gene-drug features for our experiments.

[1]The processed data and preprocessing steps can be accessed at https://modac.cancer.gov/assetDetails?dme_data_id=NCI-DME-MS01-8088592

3.5. Performance metrics

Two main LTR evaluation metrics, namely NDCG@k and MRR@k are used to assess the performance of the models. Let $D(s) = 1/\log(1 + s)$ be a discount function, $g(s) = 2^s - 1$, a monotonically increasing gain function, and $\mathcal{Z}_n = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ a set of items ordered according to their ground-truth labels, with $\mathbf{x}_i$ and y_i being an item feature vector and score, respectively. Moreover, let $\tilde{\mathcal{Z}}_n$ be a ranked list for $\mathcal{Z}_n$ according to the $\mathbf{y}$ scores. We define the *Discounted Cumulative Gain (DCG)* of $\tilde{\mathcal{Z}}_n$ as $\Phi(\tilde{\mathcal{Z}}_n) = \sum_{r=1}^{n} g(y_{\pi_r}) D(r)$, where π_r is the index of the item ranked at position r of $\tilde{\mathcal{Z}}_n$. The Ideal DCG (IDCG), $\Phi^I(\mathcal{Z}_n)$ is the DCG score of the ideal ranking result. NDCG normalizes DCG by the IDCG and can be calculated by $\Phi^N(\tilde{\mathcal{Z}}_n) = \Phi(\tilde{\mathcal{Z}}_n)/\Phi^I(\mathcal{Z}_n) \in [0, 1]$. To force the metric to focus on the top-k items, we use NDCG@k, which is the top-k version of NDCG, where the discount function is $D(s) = 0$ for $s > k$. Mean Reciprocal Rank (MRR) puts a high focus on the most relevant item of a list. Assume r_i denotes the rank of the most relevant item in the i-th list, then the reciprocal rank is defined as $R_i = 1/r_i$. For N lists, the MRR is the mean of the N reciprocal ranks, $\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} R_i$. MRR@k is simply the top-k version of MRR. The significant difference between MRR and NDCG is that NDCG distinguishes between "partially sensitive" and "highly sensitive" drugs while MRR only focuses on the most sensitive drugs. From now on, we use NDCG@k and MRR@k to denote the mean NDCG@k and the mean MRR@k (i.e., the mean of the performance metric for all lists in our test set).

3.6. Experimental settings and hyper-parameter optimization

We conducted our experiments based on the standard supervised LTR framework [33]. The authors of LETOR [6] partitioned the LTR data sets into five parts for five-fold cross-validation where three parts were used for training, one part for validation (i.e., tuning the hyperparameters of the learning algorithms), and the remaining part for evaluating the performance of the learned model. We followed the same procedure, partitioning our data sets into five-folds, and conducting five-fold cross-validation to train the models. The hyperparameters were tuned on the validation sets (i.e., we optimized the hyperparameters to maximize the NDCG score) and the average on the test sets over the 5 folds was reported in the various tables. For more details on the parameter-tuning procedure and experimental settings, please refer to Appendix.

3.7. Competing Methods

We compared our proposed ranking algorithm with three types of algorithms, namely Transformer-based Neural Ranking (TNR), Deep Neural Networks (DNN), and

traditional models. In recent years, the attention mechanism in transformers began a revolution in deep neural networks that led to major advances in the performance of many models obtained in this fashion. We compared our model with four TNR models (i.e., NDCGLoss2++ [34], ListMLE [35], ApproxNDCG [36], and RankNet [37]) to ensure its superior performance and stability compared to similar methodologies. Second, we also compared ITNR with other deep learning-based models. To that end, we compared ITNR with the so-called DNN-Sakellaropoulos (DNN-S) [15]. DNN-S has been reported as one of the best DNN models in the DRP literature [38, 8]. Third, several DRP comparative studies [39, 40, 41] reported tree-based models as one of the best-performing algorithms for drug recommendation. Specifically, $\text{LambdaMART}_{\text{MAP}}$ [42] has been shown repeatedly to surpass other LTR methods such as Coordinate Ascent, Random Forests, BoltzRank, RankBoost, AdaRank, SoftRank, and so on [40, 43, 34]. Finally, we compared ITNR with Elastic Net Regression (ENR) as a traditional DRP model. Multiple studies [44, 8, 45] have suggested that elastic net will most likely yield the most accurate predictors for drug response prediction.

4. Experimental Results

In Table 1, we summarized the performance of the models on various DRP data sets. We report NDCG@k and MRR@k, both computed out-of-sample (i.e., test set not used for training the model). The average on the test set over the 5 folds was reported. Bold and underlined numbers indicate the best performance among all methods for each metric. Bold numbers demonstrate the second-best performance among all methods for each metric.

The best model for the CCLE data set (474 unique samples and 24 unique drugs) is ITNR with an NDCG@10 of 94.03% and an MRR@10 of 93.90%, with the ENR model close behind (NDCG@10 of 93.85% and MRR@10 of 93.64%). While DNN-S achieved the highest MRR among the baseline models, ITNR outperformed it by a relatively large margin. Moreover, although NDCG@5 of TNR-RankNet is higher than our model, it does not have comparable performance considering other evaluation metrics.

On the gCSI data set (357 unique samples and 16 unique drugs), ITNR demonstrates consistent performance improvement overall baseline models across all performance metrics and any of the chosen rank cutoffs. Notably, we observe a 2.6% performance improvement compared to the average of baseline models (i.e., 78.34%) in terms of NDCG@5. Moreover, ENR which performed really well on the CCLE data set demonstrated poor performance on the gCSI data set. TNR-RankNet and LambdaMART models achieved moderate performance and are the second-best methods. The models trained on gCSI did not generalize well. This was not surprising as gCSI had the smallest number of drugs and was thus prone to overfitting. Nevertheless, our method is able to maintain its high performance.

In general, the results for these two data sets indicate that ITNR significantly outperforms other competing methods. All models except ENR and LambdaMART are Deep Learning (DL) based models. Since DL models are complex and have many learnable parameters, they tend to overfit easier than traditional models. Due to this, DL algorithms perform better when trained on large amounts of data. To that end, we also conducted experiments on the GDSC data set (670 unique cell lines and 233 unique drugs) which is one of the largest public DRP data sets. Generally, DL models outperformed other traditional methods (i.e., ENR), since a model like ENR may not fully capture the structural information within drugs. Among transformer-based baselines, we observed that ITNR achieved the best performance, with 83.36% for NDCG@5 and 92.74% for MRR@5. NDCG@25 of LambdaMART is higher than our model. However, this model is not even the second-best model considering other evaluation metrics. Among the competing models, TNR-RankNet outperformed other methods by a relatively large margin.

All in all, ITNR consistently outperforms all baseline methods across all metrics and data sets. In our experiment on three DRP data sets, TNR-RankNet and LambdaMART demonstrated reasonably good overall performance and they are the second-best methods. ITNR is not only able to push the most sensitive drugs to the top of the ranking list, but it can put them in the right order. Further, it can also generalize better and achieve superior performance on large DRP data sets like GDSC.

5. Discussion and Conclusion

Our study presented a novel transformer-based model, called Inversion Transformer-based Neural Ranking (ITNR), to predict cancer drug response. Although machine learning models are commonly used for drug response prediction, they often struggle to capture drug-cell line relations across various cell lines. Our model used the well-known Transformer architecture to extract a better drug-cell line representation. We also developed a novel loss function based on the concept of inversion and approximate permutation matrices. Hence, the proposed architecture can capture local context information and cross-item interactions that lead to a reliable drug recommendation system. Extensive experimental results demonstrated that our model is more effective than the current state-of-the-art methods highlighting the predictive capability of our model and its potential translational value in personalized medicine. Our expremental results suggest that the transformer network with multi-head attention is suitable for modeling the interactions of drug substructure and multi-omics data. With potential implications for pharmaceutical industries and healthcare providers, our work highlights the potential impact of genomics-driven drug recommendation models in advancing personalized medicine and improving patient outcomes. To effectively communicate our research findings to potential adopters, we need to tailor our approach to address the specific needs and interests of each stakeholder group. For pharmaceutical

Table 1
Performance Comparison of Ranking Methods on CCLE, GDSC, and gCSI Data Sets.

Algorithms	NDCG@5	NDCG@10	NDCG@25	MRR@5	MRR@10	MRR@25
	CCLE Data set					
ITNR	**92.15%**	**94.03%**	**94.09%**	**93.87%**	**93.90%**	**93.90%**
TNR-NDCGLoss2++	91.87%	93.71%	93.85%	92.93%	92.96%	92.96%
TNR-ListMLE	91.78%	93.41%	93.93%	93.62%	93.71%	93.71%
TNR-RankNet	**92.39%**	93.76%	**94.00%**	93.52%	93.52%	93.52%
TNR-ApproxNDCG	90.71%	92.40%	92.49%	88.05%	88.08%	88.08%
DNN-S	92.06%	93.63%	93.99%	**93.71%**	**93.74%**	**93.76%**
LambdaMART$_{MAP}$	92.03%	93.46%	93.72%	92.25%	92.28%	92.28%
ENR	91.73%	**93.85%**	93.94%	93.55%	93.64%	93.64%
	GDSC Data set					
ITNR	**83.36%**	**79.26%**	**82.40%**	**92.74%**	**92.76%**	**92.76%**
TNR-NDCGLoss2++	82.01%	78.47%	82.12%	91.33%	91.33%	91.33%
TNR-ListMLE	80.89%	77.08%	80.04%	88.77%	88.79%	88.79%
TNR-RankNet	**82.69%**	**78.59%**	82.38%	**92.16%**	**92.16%**	**92.16%**
TNR-ApproxNDCG	82.22%	77.84%	80.86%	90.79%	90.79%	90.79%
DNN-S	77.80%	76.18%	80.15%	83.96%	83.96%	83.96%
LambdaMART$_{MAP}$	81.31%	78.24%	**82.79%**	89.62%	89.64%	89.64%
ENR	76.97%	74.89%	78.39%	83.70%	83.82%	83.82%
	gCSI Data set					
ITNR	**80.36%**	**83.20%**	**83.35%**	**77.67%**	**77.71%**	**77.71%**
TNR-NDCGLoss2++	78.91%	82.62%	82.70%	76.38%	76.52%	76.52%
TNR-ListMLE	76.97%	81.08%	81.31%	75.38%	75.47%	75.47%
TNR-RankNet	79.59%	**82.95%**	**82.99%**	76.33%	76.47%	76.47%
TNR-ApproxNDCG	78.56%	82.50%	82.61%	76.09%	76.09%	76.09%
DNN-S	**79.59%**	82.67%	82.83%	76.41%	76.45%	76.45%
LambdaMART$_{MAP}$	78.36%	82.19%	82.29%	**77.67%**	**77.67%**	**77.67%**
ENR	76.45%	81.12%	81.17%	75.01%	75.10%	75.10%

industries, our research offers opportunities to optimize drug development pipelines and accelerate the identification of promising drug candidates. We emphasize the potential cost and time-saving benefits of leveraging genomics-driven drug recommendation models in streamlining drug discovery and development processes. Regarding healthcare providers, we underscore how our models can support clinicians in making more informed treatment decisions tailored to individual patient characteristics. This approach ultimately leads to improved patient outcomes and better healthcare delivery. Although our model is quite promising, some limitations could be explored and addressed in future studies. In the current model, we use an approximated permutation matrix (i.e., NeuralSort) as the sorting operator. We can replace NeuralSort with another approximation of a sorting operator such as the Optimal Transport [46] and SoftSort [47]. Furthermore, the current model assumes that all drugs have the same side effects and toxicity. We can enhance our recommendations by incorporating the toxicity of drugs into our predictions.

CRediT authorship contribution statement

Shahabeddin Sotudian: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation. **Ioannis Ch. Paschalidis:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Writing - review and editing.

Acknowledgement

This research was partially supported by the National Science Foundation (NSF) in the form of grants awarded to ICP (CCF-2200052, DMS-1664644, and IIS 1914792), the Office of Naval Research (ONR) in the form of a grant awarded to ICP (N00014-19-1-2571), the National Institutes of Health (NIH) in the form of grants awarded to ICP and the Boston University Clinical & Translation Science Institute (R01 GM135930, UL54 TR004130), and by Boston University funds. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. There was no additional external funding received for this study.

Appendix

A. Hyper-parameter optimization

We utilized grid search to fine-tune the parameters of the models. The list of hyper-parameters and their values for all ranking algorithms can be found in Table 2. In this table, L is the list length used for training (note that a list was either padded or sub-sampled to that length), d_{fc} is the dimension of the linear projection, N is the number of encoder blocks,

Table 2
The List of Hyper-parameters and Their Values.

Algorithms	Parameters	Values
All Transformer Models	L	120
	d_{fc}	128, 256
	N	2,4
	d_h	128, 512
	H	2, 4
DNN-S	D_r	0.1, 0.3, 0.5
	B	25
	E	50
	ℓ_2	0.01, 0.001, 0.0001
LambdaMART_MAP	η	0.1, 0.01, 0.001
	D_{max}	10, 100
	h_{min}	1, 10, 1000
	N_T	10, 100, 1000
ENR	α	0.01, 0.1, 1, 10, 100
	R_{ℓ_1}	0.2, 0.6, 0.8

Table 3
Best Parameters - Transformer Models

Algorithms	Data Set	L	d_{fc}	N	d_h	H
Proposed Method	CCLE	120	256	2	128	2
	GDSC	120	256	2	128	2
	gCSI	120	128	2	128	2
TNR-NDCGLoss2++	CCLE	120	128	2	128	2
	GDSC	120	256	2	128	2
	gCSI	120	128	2	128	2
TNR-ListMLE	CCLE	120	256	2	128	2
	GDSC	120	128	2	128	2
	gCSI	120	128	2	128	2
TNR-RankNet	CCLE	120	128	2	128	2
	GDSC	120	128	2	128	2
	gCSI	120	128	2	128	2
TNR-ApproxNDCG	CCLE	120	256	2	128	2
	GDSC	120	256	2	128	2
	gCSI	120	128	2	128	2

d_h is the transformer hidden dimension, H is the number of attention heads, D_r refers to the hidden dropout ratio, B is the batch size, E is the number of epochs, ℓ_2 is the parameter of the ℓ_2-norm penalty for the DNN-S model, η is the learning rate, D_{max} is the maximum depth of a tree, h_{min} is the minimum sum of the instance weight (Hessian) needed in a leaf, N_T is the number of estimators, α is the regularization strength of the model, and R_{ℓ_1} controls the contribution of the ℓ_1 and ℓ_2 penalties in the ENR model. The details of hyper-parameter settings of all methods for the five folds can be found in Tables 3, 4, 5, and 6.

Table 4
Best Parameters - DNN-S

Data Sets	Folds	B	E	D_r	ℓ_2
CCLE	Fold 1	25	50	0.3	0.0001
	Fold 2	25	50	0.5	0.01
	Fold 3	25	50	0.1	0.01
	Fold 4	25	50	0.3	0.001
	Fold 5	25	50	0.5	0.01
GDSC	Fold 1	25	50	0.3	0.001
	Fold 2	25	50	0.5	0.01
	Fold 3	25	50	0.3	0.01
	Fold 4	25	50	0.3	0.001
	Fold 5	25	50	0.3	0.01
gCSI	Fold 1	25	50	0.3	0.001
	Fold 2	25	50	0.3	0.001
	Fold 3	25	50	0.3	0.01
	Fold 4	25	50	0.3	0.01
	Fold 5	25	50	0.3	0.01

Table 5
Best Parameters - LambdaMART

Data Sets	Folds	η	D_{max}	N_T	h_{min}
CCLE	Fold 1	0.1	10	100	1
	Fold 2	0.1	10	100	100
	Fold 3	0.1	100	100	1
	Fold 4	0.1	10	1000	10
	Fold 5	0.001	100	1000	1
GDSC	Fold 1	0.1	10	1000	10
	Fold 2	0.001	10	100	1
	Fold 3	0.1	100	1000	10
	Fold 4	0.1	100	1000	1
	Fold 5	0.1	100	1000	1
gCSI	Fold 1	0.01	10	1000	1
	Fold 2	0.1	10	1000	1
	Fold 3	0.001	10	1000	1
	Fold 4	0.1	10	1000	1
	Fold 5	0.1	10	1000	1

Table 6
Best Parameters - ENR

Data Sets	Folds	α	R_{ℓ_1}
CCLE	Fold 1	0.01	0.2
	Fold 2	0.01	0.2
	Fold 3	0.01	0.2
	Fold 4	0.1	0.2
	Fold 5	0.01	0.2
GDSC	Fold 1	0.01	0.2
	Fold 2	0.01	0.2
	Fold 3	0.01	0.2
	Fold 4	0.01	0.2
	Fold 5	0.01	0.2
gCSI	Fold 1	0.1	0.2
	Fold 2	0.01	0.6
	Fold 3	0.1	0.6
	Fold 4	0.1	0.2
	Fold 5	0.01	0.6

References

[1] Boran Hao, Yang Hu, Shahabeddin Sotudian, Zahra Zad, William G Adams, Sabrina A Assoumou, Heather Hsu, Rebecca G Mishuris, and Ioannis Ch. Paschalidis. Development and validation of predictive models for covid-19 outcomes in a safety-net hospital population. *Journal of the American Medical Informatics Association*, 2022.

[2] Mohammad Hossein Fazel Zarandi, Shahabeddin Sotudian, and Oscar Castillo. A new validity index for fuzzy-possibilistic c-means clustering. *arXiv preprint arXiv:2005.09162*, 2020.

[3] Shahabeddin Sotudian, Ruidi Chen, and Ioannis Ch. Paschalidis. Distributionally robust multi-output regression ranking. *arXiv preprint arXiv:2109.12803*, 2021.

[4] Shahabeddin Sotudian, Israel T Desta, Nasser Hashemi, Shahrooz Zarbafian, Dima Kozakov, Pirooz Vakili, Sandor Vajda, and Ioannis Ch. Paschalidis. Improved cluster ranking in protein–protein docking using a regression approach. *Computational and structural biotechnology journal*, 19:2269–2278, 2021.

[5] Xiaoqing Ru, Xiucai Ye, Tetsuya Sakurai, and Quan Zou. Application of learning to rank in bioinformatics tasks. *Briefings in Bioinformatics*, 2021.

[6] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

[7] Fangfang Xia, Jonathan Allen, Prasanna Balaprakash, Thomas Brettin, Cristina Garcia-Cardona, Austin Clyde, Judith Cohn, James Doroshow, Xiaotian Duan, Veronika Dubinkina, et al. A cross-study analysis of drug response prediction in cancer cell lines. *Briefings in bioinformatics*, 23(1):bbab356, 2022.

[8] Shahabeddin Sotudian and Ioannis Ch. Paschalidis. Machine learning for pharmacogenomics and personalized medicine: A ranking model for drug sensitivity prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2021.

[9] Przemysław Pobrotyn, Tomasz Bartczak, Mikołaj Synowiec, Radosław Białobrzeski, and Jarosław Bojar. Context-aware learning to rank with self-attention. *arXiv preprint arXiv:2005.10084*, 2020.

[10] Jun Li, Qing Lu, and Yalu Wen. Multi-kernel linear mixed model with adaptive lasso for prediction analysis on high-dimensional multi-omics data. *Bioinformatics*, 36(6):1785–1794, 2020.

[11] Aman Sharma and Rinkle Rani. Drug sensitivity prediction framework using ensemble and multi-task learning. *International Journal of Machine Learning and Cybernetics*, 11(6):1231–1240, 2020.

[12] In Sock Jang, Elias Chaibub Neto, Justin Guinney, Stephen H Friend, and Adam A Margolin. Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. In *Biocomputing 2014*, pages 63–74. World Scientific, 2014.

[13] Delora Baptista, Pedro G Ferreira, and Miguel Rocha. Deep learning for drug response prediction in cancer. *Briefings in bioinformatics*, 22(1):360–379, 2021.

[14] Omid Bazgir, Ruibo Zhang, Saugato Rahman Dhruba, Raziur Rahman, Souparno Ghosh, and Ranadip Pal. Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks. *Nature communications*, 11(1):1–13, 2020.

[15] Theodore Sakellaropoulos, Konstantinos Vougas, Sonali Narang, Filippos Koinis, Athanassios Kotsinas, Alexander Polyzos, Tyler J Moss, Sarina Piha-Paul, Hua Zhou, Eleni Kardala, et al. A deep learning framework for predicting response to therapy in cancer. *Cell reports*, 29(11):3367–3373, 2019.

[16] Milad Ahmadian, Mahmood Ahmadi, Sajad Ahmadian, Seyed Mohammad Jafar Jalali, Abbas Khosravi, and Saeid Nahavandi. Integration of deep sparse autoencoder and particle swarm optimization to develop a recommender system. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2524–2530. IEEE, 2021.

[17] Amir Khani Yengikand, Majid Meghdadi, and Sajad Ahmadian. Dhsirs: a novel deep hybrid side information-based recommender system. *Multimedia Tools and Applications*, pages 1–27, 2023.

[18] Amir Khani Yengikand, Majid Meghdadi, Sajad Ahmadian, Seyed Mohammad Jafar Jalali, Abbas Khosravi, and Saeid Nahavandi. Deep representation learning using multilayer perceptron and stacked autoencoder for recommendation systems. In *2021 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2485–2491. IEEE, 2021.

[19] Jinyu Chen and Louxin Zhang. A survey and systematic assessment of computational methods for drug response prediction. *Briefings in bioinformatics*, 22(1):232–246, 2021.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[21] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

[22] Przemysław Pobrotyn and Radosław Białobrzeski. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *arXiv preprint arXiv:2102.07831*, 2021.

[23] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964.

[24] Shahabeddin Sotudian and Mohammad Hossein Fazel Zarandi. Interval type-2 enhanced possibilistic fuzzy c-means clustering for gene expression data analysis. *arXiv preprint arXiv:2101.00304*, 2021.

[25] Argyro Mavrogiorgou, Athanasios Kiourtis, Konstantinos Perakis, Dimitrios Miltiadou, Stamatios Pitsios, and Dimosthenis Kyriazis. Analyzing data and data sources towards a unified approach for ensuring end-to-end data and data sources quality in healthcare 4.0. *Computer methods and programs in biomedicine*, 181:104967, 2019.

[26] Behrouz Ehsani-Moghaddam, Ken Martin, and John A Queenan. Data quality in healthcare: A report of practical experience with the canadian primary care sentinel surveillance network data. *Health Information Management Journal*, 50(1-2):88–92, 2021.

[27] Amelia Ritahani Ismail, Nadzurah Zainal Abidin, and Mhd Khaled Maen. Systematic review on missing data imputation techniques with machine learning algorithms for healthcare. *Journal of Robotics and Control (JRC)*, 3(2):143–152, 2022.

[28] Ali Akbar Sadat Asl, Mohammad Mahdi Ershadi, Shahabeddin Sotudian, X Li, and S Dick. Fuzzy expert systems for prediction of icu admission in patients with covid-19. *Intelligent Decision Technologies*, (Preprint):1–10, 2022.

[29] Shahabeddin Sotudian, Aaron Afran, Christina A LeBedis, Anna F Rives, Ioannis Ch Paschalidis, and Michael DC Fishman. Social determinants of health and the prediction of missed breast imaging appointments. *BMC Health Services Research*, 22(1):1–11, 2022.

[30] Austin Clyde, Tom Brettin, Alexander Partin, Maulik Shaulik, Hyunseung Yoo, Yvonne Evrard, Yitan Zhu, Fangfang Xia, and Rick Stevens. A systematic approach to featurization for cancer drug sensitivity predictions with deep learning. *arXiv preprint arXiv:2005.00095*, 2020.

[31] Amar Koleti, Raymond Terryn, Vasileios Stathias, Caty Chung, Daniel J Cooper, John P Turner, Dušica Vidović, Michele Forlin, Tanya T Kelley, Alessandro D'Urso, et al. Data portal for the library of integrated network-based cellular signatures (lincs) program: integrated access to diverse large-scale cellular perturbation response data. *Nucleic acids research*, 46(D1):D558–D566, 2018.

[32] Boran Hao, Shahabeddin Sotudian, Taiyao Wang, Tingting Xu, Yang Hu, Apostolos Gaitanidis, Kerry Breen, George C Velmahos, and Ioannis Ch. Paschalidis. Early prediction of level-of-care requirements in patients with covid-19. *Elife*, 9:e60519, 2020.

[33] Tie-Yan Liu. Learning to rank for information retrieval. 2011.

[34] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1313–1322, 2018.

[35] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1192–1199, 2008.

[36] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval*, 13(4):375–397, 2010.

[37] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[38] JungHo Kong, Doyeon Ha, Juhun Lee, Inhae Kim, Minhyuk Park, Sin-Hyeog Im, Kunyoo Shin, and Sanguk Kim. Network-based machine learning approach to predict immunotherapy response in cancer patients. *Nature communications*, 13(1):1–15, 2022.

[39] Saad Haider, Raziur Rahman, Souparno Ghosh, and Ranadip Pal. A Copula Based Approach for Design of Multivariate Random Forests for Drug Sensitivity Prediction. *PLoS ONE*, 10(12), December 2015.

[40] Carlos De Niz, Raziur Rahman, Xiangyuan Zhao, and Ranadip Pal. Algorithms for Drug Sensitivity Prediction. *Algorithms*, 9(4):77, December 2016. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[41] Gregory Riddick, Hua Song, Susie Ahn, Jennifer Walling, Diego Borges-Rivera, Wei Zhang, and Howard A. Fine. Predicting in vitro drug sensitivity using Random Forests. *Bioinformatics*, 27(2):220–224, January 2011. Publisher: Oxford Academic.

[42] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[43] Sebastian Bruch. An alternative cross entropy loss for learning-to-rank. In *Proceedings of the Web Conference 2021*, pages 118–126, 2021.

[44] In Sock Jang, Elias Chaibub Neto, Justin Guinney, Stephen H. Friend, and Adam A. Margolin. Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. In *Pacific Symposium on Biocomputing*, pages 63–74. WORLD SCIENTIFIC, November 2013.

[45] Paul Geeleher, Nancy J. Cox, and R. Stephanie Huang. Clinical drug response can be predicted using baseline gene expression levels and in vitrodrug sensitivity in cell lines. *Genome Biology*, 15(3):R47, March 2014.

[46] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *Advances in neural information processing systems*, 32, 2019.

[47] Sebastian Prillo and Julian Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, pages 7793–7802. PMLR, 2020.