Deep reinforcement learning for personalized treatment recommendation

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Mingyang Liu

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Xiaotong Shen and
Wei Pan, Advisers

March 2023

Acknowledgements

First and foremost, I would like to take this opportunity to express my greatest appreciations to my thesis advisor Prof. Xiaotong Shen. Over the years, he has been generously providing me with valuable suggestions for my research and encouraging me to try different ideas. I'm deeply grateful and honored to be influenced by his insights and professionalism.

I am also grateful to Prof. Wei Pan, who not only helps me in how to resolve a statistical problem, but also offer me great opportunities to learn in different areas. Thanks to his constant support, help and patience for various aspects of my research and my life that made me enjoy my work and made this work possible.

I would also like to thank the rest of my committee members, Prof.Galin Jones and Prof.Adam Rothman. I'm grateful to have them as my committee members and I deeply appreciate their guidance and words of encouragement along the way.

I'm also grateful to my friends at the University of Minnesota for helping me and making my life at UMN a lot of fun, too many friends to say thank you here! Thanks to Xuetong Sun and Yunan Wu, for all your support whenever I needed., I also want to thank my roomate Jiahuan Ye, thank you for being my roomate for four years, I was so lucky to meet you and having you for more than 4 years!

And finally, I would like to thank my parents Guoying Liu and Xiaoling Cao and my husband Weiqiang Zhu for their unconditional love. Especially my husband, who has always pushed me forward without stop.

Dedication

This dissertation is dedicated to my father Guoying Liu and to my mother Xiaoling Cao.

Abstract

In precision medicine, the ultimate goal is to recommend the most effective treatment to an individual patient based on patient-specific molecular and clinical profiles, possibly high-dimensional. To advance cancer treatment, large-scale screenings of cancer cell lines against chemical compounds have been performed to help better understand the relationship between genomic features and drug response; existing machine learning approaches use exclusively supervised learning, including penalized regression and recommender systems.

When there is only one time point, it refers to individualized treatment selection, which is employed to maximize a certain clinical outcome of a specific patient based on a patient's clinical or genomic characteristics, given a patients' heterogeneous response to treatments. Although developing such a rule is conceptually important to personalized medicine, existing methods such as the L_1-penalized least squares (54) suffers from the difficulty of indirect maximization of clinical outcome, while the outcome weighted learning (109) directly maximizing the clinical outcome is not robust against any perturbation of the outcome. We will first propose a weighted ψ-learning method to optimize an individualized treatment rule, which is robust again perturbation of data near decision boundary through the notation of separation. To deal with nonconvex minimization, we employ a difference of convex algorithm to solve the non-convex minimization iteratively based on a decomposition of the cost function into a difference of two convex function. On this ground, we also introduce a variable selection method for further removing redundant variables for higher performance. Finally, we illustrate the proposed method through simulations and a lung health study, and demonstrate that it yields higher performance in terms of accuracy of prediction of individualized treatment.

However, it would be more efficient to apply reinforcement learning (RL) to sequentially learn as data accrue, including selecting the most promising therapy for a patient given individual molecular and clinical features and then collecting and learning from the corre-

sponding data. In this way, we propose a novel personalized ranking system called Proximal Policy Optimization Ranking (PPORank), which ranks the drugs based on their predicted effects per cell line (or patient) in the framework of deep reinforcement learning (DRL). Modeled as a Markov decision process (MDP), the proposed method learns to recommend the most suitable drugs sequentially and continuously over time. As a proof-of-concept, we conduct experiments on two large-scale cancer cell line data sets in addition to simulated data. The results demonstrate that the proposed DRL-based PPORank outperforms the state-of-the-art competitors based on supervised learning. Taken together, we conclude that novel methods in the framework of DRL have great potential for precision medicine and should be further studied.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Developing an individualized treatment rule becomes increasingly important given patients' heterogeneous response to treatments. Often an optimal treatment rule involves a patient's clinical or genomic characteristics, whose target is to maximize the expected clinical outcome for a specific patient. For example, (3; 40; 48) pointed out that over about 40% of colorectal cancers, make the tumor unresponsive to anti-epidermal growth factor receptor therapy with cetuximab or panitumumab, which makes certain therapy does not work in colorectal cancer. So experiments studying the correlation of mutations, microsatellite instability, and hypermethylation in tumors from individual patients is conducted. Thus we can identify subgroups which are likely or not likely to respond to a particular treatment regimen (6). By taking the individual variability into account, we just change from the traditional "one size fits all" approach to modern personalized medicine.

(54) proposes a two-step procedure, which first estimates a conditional mean and then estimated the treatment rule based on the conditional mean. A rich conditional mean models is used to model the conditional mean correctly, variable selection was conducted through L_1-penalty. While the conditional mean approximation requires accuracy in specifying the conditional mean model, and this method emphasizes on predicting the clinical response instead of directly optimizing the decision rule. Different from building models on conditional mean outcomes or contrasts between mean outcomes, (109) proposed the outcome weighted learning(OWL), which directly estimating the decision rule. They demonstrated that the original problem could be regarded as a weighted classification problem and used hinge

loss to surrogate the empirical loss, which made the architecture like the support vector machine(SVM).

Although OWL fills the gap between the ITR estimation and machine learning technique, it has some disadvantages. It assumes that a shift in R will not change the estimation of optimal treatment rule, while it has already been proposed by (115) that the estimated ITR of OWL is affected by a simple shift of the outcome. Note that OWL had the assumption of R always being positive, which made the algorithm to achieve convex optimization much easier, is not appropriate for real data in randomized clinical trial. A direct explanation is that from ((2.2)) we can see when the outcome is positive, the treatment $\mathcal{D}(\boldsymbol{X})$ using OWL tends to match the treatment A_i that has been assigned to the patient, but in the randomized trial this is not ideal for real data. Furthermore, variable selection is not used. Since there are many predictive variables that may only contribute to the outcome but does not effect the treatment interaction, it is necessary to include variable selection to construct optimal individualized treatment rule. Third, when extended to non-separable cases, SVM becomes less solid, and generation errors becomes important but has not been taken into account by OWL method.

Based on the above limitations, we propose a ψ-learning method, weighted by clinical outcomes, allowing for positive and negative outcomes. The proposed method is advantageous for estimating the optimal ITR in two aspects. First, it is robust again data perturbation, that is, a local perturbation of data points has no impact or a small impact on classification. Second, it retains the large margin interpretation to increase the level of separation between two treatment outcomes. To further enhance the predictive performance, we equip the proposed approach with the capability of variable selection to remove non-informative clinical attributes. In a sense, the proposed method can be regarded a further generalization of the ψ-learning approach of (80) to weighted classification. Computationally, we develop a difference convex algorithm to treat nonconvex minimization.

When not limited for a single time point data, one main challenge in precision medicine (45) is to understand common disease at the molecular level to recommend individualized

therapies to patients, permitting high efficacy for different and possibly unknown disease subtypes. Another challenge in clinical practice is how to adapt treatment assignments to possible changes in patients' health states and preferences, including previous treatment history. Optimal treatment is determined by maximizing an evaluation signal indicating a long-term patient outcome, accounting for possibly delayed treatment effects and influences on future treatment choices.

An important resource for advancing precision medicine is through drug discovery via computational prediction of drug sensitivity for various cell lines based on their high-dimensional genomic features as wells as the chemical structures of drugs. (58) For such a purpose, large-scale screenings of cancer cell lines have been performed with detailed molecular profiles. For example, the Cancer Cell-Line Encyclopedia (CCLE) (7) provides large-scale molecular profiles including genomic (e.g. genetic mutations), transcriptomic (i.e. gene expression) and epigenomic (e.g. DNA methylation) data. Similarly, the Genomics of Drug Sensitivity in Cancer (GDSC) project (37) has been carried out for investigating drug responses of numerous cancer cell lines, which are measured by each drug's half-maximal inhibitory concentration (IC_{50}); the smaller IC_{50}, the more effective (potentially) the drug. GDSC also provides multi-omics data, including whole-exome sequencing (WES), copy number variation (CNV), and DNA methylation (MET) data. Whereas it is challenging to examine the patient's response to different clinical options, the Cancer Genome Atlas (TCGA) (97) collected follow-up records of around 10k pan-cancer patients for survival/recurrence in response to given treatments. (97) We will use CCLE and GDSC to develop and (internally) evaluate our proposed and other existing learning methods before using a TCGA dataset for external validation.

To advance drug discovery, various approaches have been proposed for predicting drug responses using genomic features. (7; 37; 19; 38; 5; 1) For example, some (7; 37; 19; 38; 1) have proposed linear regression models to predict drug sensitivity (as measured by IC_{50}) or the area under the fitted dose response curve (AUC). Some non-linear models such as random forests, kernel regression, and neural networks have also appeared. (58; 31; 18)

These models are drug-specific as being trained for each drug separately and independently. A drug-specific model is limited by the small number of cell lines available with a given drug. To increase the sample size, others (19) have proposed a Bayesian multitask multiple kernel learning (BMTMKL) approach, achieving the best performance in a DREAM challenge for drug response prediction. This method has shown the importance of information sharing among the drugs. Besides, there is a matrix factorization approach based on cell line and drug similarities (SRMF), which, however, could not be used for previously unseen cell lines. (91) In a similar framework of matrix factorization, a recommender system (CaDRReS) to map the drugs and cell lines into a latent 'pharmacogenomic' space before predicting drug responses has been proposed. (82) However, most of these methods may not be most clinically relevant: they directly predict drug responses, differing from directly ranking drugs. Accordingly, kernelized rank learning (KRL) has been proposed to use a ranking metric called Normalized Discounted Cumulative Gain (NDCG@k) to recommend the top k most sensitive drugs. (33) Similarly, CaDRReS (82) is another ranking procedure. However, both KRL and CaDRReS optimize either an approximate evaluation metric or the mean square error (MSE) without directly optimizing the ranking evaluation metric NDCG. Recently, various deep learning (DL) models have emerged (21), showing that simple neural networks such as multi-layer perceptrons (MLPs) could be effective competitors to lasso or elastic net. (102) CDRscan (12) aims to explore various architectures of MLPs to predict responses from cell line genomic features and drug fingerprints.

However, there are potential limitations with these methods. Importantly, all the methods are supervised, including regression. In practice, it would be more efficient for a learning system to learn *sequentially and continuously* as the data accrue, rather than collecting all the data first before starting learning. Accordingly, we propose a deep reinforcement learning (DRL) based approach, called **PPORank** (50). PPORank treats ranking as a sequential decision-making process. We adopt a ranking evaluation metric, called NDCG, as the long-term reward, and take both the long-term effects of the ranking policy and the expected value of each ranking position into account. Thus we utilize more information from a ranking. One

novel component of PPORank is our adaptation of a model-free "actor-critic" scheme. We directly optimize the non-differentiable evaluation measure NDCG through policy gradient and achieve both stability and sample efficiency via the PPO optimization method. We design a state representation module using a deep neural network from a policy network, which is extensible to integrating heterogeneous data sources and is capable to capture complex non-linear relationships between cell lines/patients and drugs. We have carried out some experiments on cancer drug screening data, which are high-dimensional in the state and action spaces with only limited and sparse data. The results demonstrate that the proposed method outperforms the state-of-the-art competitors.

Our proposed PPORank and its DRL framework are related to, but differ from, the existing literature on estimating optimal dynamic treatment regimes (DTRs), (70; 71; 64) in which often only a small number of treatments or time points (called finite time horizons) are considered , (108; 110; 112; 47; 62; 103; 106) though a few exceptions with infinite time horizons are emerging. The methods in the first category are mainly based on Q-learning and A-learning, (75) that often based on parametric/linear regression models for the value functions. When there is only one time point, it reduces to individualized treatment selection, which is done in the framework of supervised learning, such as the outcome weighted learning, (109) ψ-learning (Liu et al.) and residual weighted learning. (115) Two representatives for infinite time horizons have appeared recently and are based on some postulated linear models for the Q-value function (24) and the V-value function (53) respectively; see (45) for a nice and up-to-date review. Our proposed method in the framework of DRL not only is more flexible by allowing both nonparametric modeling with DL and infinite time horizons, but also takes a state-of-the-art actor-critic approach based on the PPO algorithm. In addition, many other methods require that the data have been collected (from either randomized experiments or observational studies) at the time of their application, while RL can learn to both recommend treatments and thus collect data simultaneously, which might be more efficient and timely in applications such as mobile health (mHealth) for continuously learning and thus recommending better and better behavioral interventions, (53) though this will not

be further explored here.

- Chapter 2 briefly introduces a framework for individualized treatment rule estimation and proposes a ψ-learning method.
- Chapter 3 briefly introduces the main components of the proposed DRL-based method.
- Chapter 4 shows our numerical results used in the ψ-learning method.
- Chapter 5 describes the numerical results used in the DRL-based method.
- Chapter 6 gives a conclusion and discussion of our methods.

Chapter 2

Outcome Weighted Learning

Consider a two-arm randomized trial, in which a treatment assignment $A \in \mathcal{A} = \{-1, 1\}$ is independent of any patient's prognostic variables, denoted by a p-dimensional $\mathbf{X} = (X_1, \cdots, X_p) \in \mathbb{R}^p$. Let R be an observed clinical outcome, called the "reward", which is used to measure the effectiveness of a treatment with a larger value of R being desirable; without loss of generality, assume that R is bounded. To develop an individualized treatment rule (ITR) $\mathcal{D}$, we construct a mapping from prognostic variables $\mathbf{X}$ to $\mathcal{A}$.

To construct $\mathcal{D}$, we maximize the expected reward : $\mathcal{V}(\mathcal{D}) = E^{\mathcal{D}}(R)$, where $E^{\mathcal{D}}$ is the expectation with the conditional distribution of $(\mathbf{X}, A, R)$ given $A = \mathcal{D}(X)$. To simplify $E^{\mathcal{D}}(R)$, note that $E^{\mathcal{D}}(R) = E\big(R\frac{dP^{\mathcal{D}}}{dP}\big) = E\big(\frac{I(A=\mathcal{D}(\mathbf{X}))}{P(A|\mathbf{X})}R\big)$, where $P^{\mathcal{D}}$ and P are the conditional distribution of $(\mathbf{X}, A, R)$ given $A = \mathcal{D}(X)$ and the joint distribution of $(\mathbf{X}, A, R)$. To derive $\frac{dP^{\mathcal{D}}}{dP}$, recall that $P \sim p_0(x)p(a|x)p_1(r|x, a)$ and $P^{\mathcal{D}} \sim p_0(x)I_{a=\mathcal{D}(x)}p_1(r|x, a)$. Then $\frac{dP^{\mathcal{D}}}{dP} = I_{a=\mathcal{D}(x)}/P(a|x)$ when $P[p(a|x) > 0] = 1$ for all $a \in \mathcal{A}$. Now

$$\mathcal{V}(\mathcal{D}) = E\big(R\frac{dP^{\mathcal{D}}}{dP}dP\big) = E\big(\frac{I(A = \mathcal{D}(\mathbf{X}))}{\pi(A, \mathbf{X})}R\big), \tag{2.1}$$

where $\pi(A, \mathbf{X}) := P(A|\mathbf{X})$.

To obtain an optimal ITR, we minimize $E\left(\frac{R}{\pi(A,\mathbf{X})}I(A \neq \mathcal{D}(\mathbf{X}))\right)$ with respect to $\mathcal{D}$. Given a random sample $\{(\mathbf{X}_i, A_i, R_i)_{i=1}^n\}$, we construct its empirical loss as follows:

$$n^{-1}\sum_{i=1}^{n}\frac{R_i}{\pi(A_i, \mathbf{X}_i)}I\left(A_i \neq \text{sign}\left(f\left(\mathbf{X}_i\right)\right)\right) = n^{-1}\sum_{i=1}^{n}\frac{R_i}{\pi(A_i, \mathbf{X}_i)}I\left(A_i f\left(\mathbf{X}_i\right) < 0\right), \tag{2.2}$$

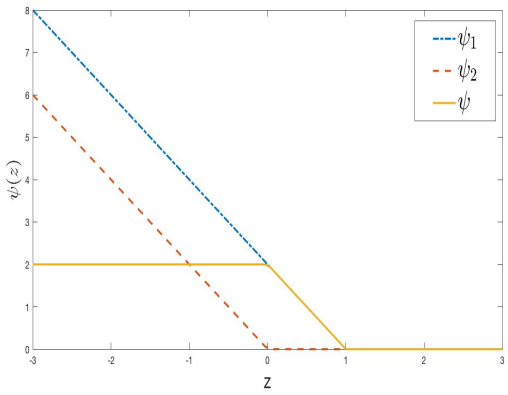

Figure 2.1: Plot of a DC decomposition of ψ as $\psi = \psi_1 - \psi_2$, a difference of two convex functions.

with $\pi(A_i, \mathbf{X}_i) = A_i\pi + (1 - A_i)/2$.

2.1 ψ-learning

In ((2.2)), we replace the intractable indicator function $I(A_i f(\mathbf{X}_i) < 0)$ by its computational surrogate $\psi(A_i f(\mathbf{X}_i))$, where $\psi(z) = \frac{1}{\tau}$ if $z \leq 0$, $\psi(z) = \frac{1-z}{\tau}$ if $0 \leq z \leq 1$ and 0 otherwise, where τ is a tuning parameter and yields an approximation as $\tau \to 0^+$; see Figure Figure 2.1 for a display of the ψ function.

In the linear case, the decision function $f(\widetilde{\boldsymbol{X}})$ is parametrized as $\langle \mathbf{w}, \widetilde{\boldsymbol{X}} \rangle$, where $\widetilde{\boldsymbol{X}} = (\mathbf{X}', 1)'$ is an augmented vector to absorb the intercept, $\mathbf{w} = (\boldsymbol{w}^*, w_{p+1}) = (w_1, w_2, \cdots, w_{p+1}) \in \mathbb{R}^{p+1}$, and $\langle \cdot, \cdot \rangle$ represents the inner product in the Euclidean Space. The cost function to minimize here becomes

$$S(\boldsymbol{w}) = \frac{\kappa}{2}\|\boldsymbol{w}^*\|^2 + \frac{1}{n}\sum_{i=1}^{n} \frac{R_i}{\pi(A_i, \mathbf{X}_i)} \psi\left(A_i f(\widetilde{\boldsymbol{X}}_i)\right), \tag{2.3}$$

where $\kappa > 0$ is a tuning parameter controlling the balance between the separation margin and training, and $\tau > 0$ is a tuning parameter whose value should be close to zero to yield a good approximation of the ψ-function to the indicator function.

In the nonlinear case, $f(\widetilde{\boldsymbol{X}}) = \langle \mathbf{w}, \tilde{\mathbf{X}} \rangle \in \mathcal{H}_K$ (42), where $\mathcal{H}_K$ is a reproducing kernel

Hilbert Space (RKHS) associated a Mercer kernel function K, and $K(\cdot,\cdot)$ is a mapping from $\mathcal{X}\times\mathcal{X}$ to $\mathbb{R}$. Then we can represent $\widetilde{\boldsymbol{X}}$ as $\widetilde{\boldsymbol{X}} = (K(\cdot,\mathbf{X}),1)'$, $\mathbf{w} = (\boldsymbol{w}^*, w_{n+1}) = (w_1, w_2, \cdots, w_{n+1}) \in \mathbb{R}^{n+1}$, the norm is denoted by $\|\boldsymbol{w}^*\|_K^2 = \sum_{i=1}^n \sum_{j=1}^n w_i w_j K(\mathbf{X}_i, \mathbf{X}_j)$. Then the cost function becomes:

$$S(\boldsymbol{w}) = \frac{\kappa}{2}\|\boldsymbol{w}^*\|_K^2 + \frac{1}{n}\sum_{i=1}^{n} \frac{R_i}{\pi(A_i,\mathbf{X}_i)} \psi\left(A_i f(\widetilde{\boldsymbol{X}}_i)\right). \tag{2.4}$$

A most widely used nonlinear kernel is the Gaussian kernel, $K_\sigma(\mathbf{x},\boldsymbol{z}) = \exp(-\frac{1}{\sigma^2}\|\mathbf{x}-\boldsymbol{z}\|^2)$, where σ is a scale parameter for K_σ.

2.1.1 DC Algorithms

There are some major advances in the computation of global optima when an objective function has a difference convex representation. ψ-learning introduced the non-convex loss function, where is currently no efficient algorithm to solve. But from Figure (Figure 2.2), we can see there is DC property (85) of the ψ-learning , and major advantages in computational of global optimization can be adopted here (85; 46). Hence to solve the non-convex minimization of the cost function in ((2.3)) and ((2.4)), we employ difference convex programming (109; 78) by decomposing the cost function into a difference of two convex functions, on which convex relation is performed by replacing the second part by its minorization at iteration m. A D.C. decomposition of $\psi(z)$ is $\psi(z) = \psi_1(z) - \psi_2(z)$, where $\psi_1(z) = \max(\frac{1-z}{\tau}, 0)$ and $\psi_2(z) = \max(-\frac{z}{\tau}, 0)$. Then $\psi\left(A_i f(\widetilde{\boldsymbol{X}}_i)\right) = \max(\frac{1-A_i f(\widetilde{\boldsymbol{X}}_i)}{\tau}, 0) - \max(\frac{-A_i f(\widetilde{\boldsymbol{X}}_i)}{\tau}, 0)$.

Here is a litter different from the original ψ-learning in (52), for the weight could be both positive and negative. Now we rewrite the loss function as

$$S(\mathbf{w}) = S_1(\mathbf{w}) - S_2(\mathbf{w}) \tag{2.5}$$

where

$$S_1(\mathbf{w}) = \frac{\kappa}{2}\|\boldsymbol{w}^*\|^2 + \frac{1}{n}\sum_{i=1}^{n}[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) + \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)]\frac{|R_i|}{\pi(A_i, \mathbf{X}_i)},$$

$$S_2(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}[\psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) + \psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)]\frac{|R_i|}{\pi(A_i, \mathbf{X}_i)}.$$

For simplicity, we use $\|\mathbf{w}^*\|$ to represent some norm for $\mathbf{w}^*$.

Given ((2.5)), we apply DC programming to solve a convex subproblem, which provides a sequence of a non-increasing upper approximation to the original problem. At iteration m, only a subproblem needs to be solved, or equivalently,

$$\min_{\mathbf{w}}(S_1(\mathbf{w}) - \langle \mathbf{w}, \nabla S_2(\mathbf{w}^{(m)}\rangle) \tag{2.6}$$

The details of solving the subproblem is provided in the Appendix.

The algorithm is as follows :

Algorithm

Step1 (Initialization): Supply an initial estimate $\hat{\mathbf{w}}^{(0)}$ and tolerance error $\epsilon > 0$.

Step2 (Iteration): At iteration m, computer $\hat{\mathbf{w}}^{(m+1)}$ by minimizing ((2.6)).

Step3 (Termination): Terminate when $|S(\hat{\mathbf{w}}^{(m+1)}) - S(\hat{\mathbf{w}}^{(m)})| \leq \epsilon$. Then the estimate $\mathbf{w}^s = \hat{\mathbf{w}}^{(m^*)}$, where m^* is the smallest index satisfying the termination criterion, and accordingly we can obtain the estimate ITR of $\hat{\mathcal{D}}\left(\widetilde{\boldsymbol{X}}\right) = \text{sign}(\langle \mathbf{w}^s, \widetilde{\boldsymbol{X}}\rangle)$.

We will derive the optimization algorithm for both linear and non-linear cases in the following sections.

2.1.2 Initial estimate

In general, there are many good or randomly selected initial estimates to obtain multiple solutions, especially for the non-convex optimization problems, from which a subset with

smaller objectives or more parsimonious models can be selected. Below we describe a simple way to obtain the initial estimate, which we can use in the following simulations. Taking the initial value of $\mathbf{w}$ as $\mathbf{0}$, and we only need to optimize $S_1(\mathbf{w})$ over $\mathbf{w}$ in (2.5). Thus we can have :

$$\mathbf{w}^{(0)} = \min_{\mathbf{w}} \frac{\kappa}{2}\|\boldsymbol{w}^*\|^2 + \frac{1}{n}\sum_{i=1}^{n}\left[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) + \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)\right]\frac{|R_i|}{\pi(A_i, \mathbf{X}_i)} \tag{2.7}$$

which is a convex function and can be easily handled. Another choice comes from the estimation from the OWL method.

2.1.3 Linear decision function of ψ-learning for optimal ITR

For linear classification, minimization ((2.5)) can be rewritten as

$$\min_{\boldsymbol{w}^*, w_{p+1}} \frac{\kappa}{2}\mathbf{w}^{*T}\boldsymbol{w}^* + \frac{1}{n}\sum_{i=1}^{n}\frac{R_i}{\pi(A_i, \mathbf{X}_i)}\psi\left(A_i(\mathbf{w}^{*T}\mathbf{X}_i + w_{p+1})\right). \tag{2.8}$$

Supply an initial estimate $\hat{\mathbf{w}}^{(0)}$ by ((2.7)), then

$$\begin{aligned}\hat{\mathbf{w}}^{(m+1)} &= \min_{\mathbf{w}} S(\mathbf{w}) = \min_{\mathbf{w}} \left(S_1(\mathbf{w}) - \langle \mathbf{w}, \triangledown S_2(\hat{\mathbf{w}}^{(m)})\rangle\right) \\ &= \min_{\mathbf{w}} \frac{\kappa}{2}\mathbf{w}^{*T}\boldsymbol{w}^* + \frac{1}{n}\sum_{i=1}^{n}\left[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) + \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)\right]\frac{|R_i|}{\pi(A_i, \mathbf{X}_i)} \\ &\quad - \mathbf{w}^{*'}V_1^{(m)} - w_{p+1}V_2^{(m)},\end{aligned} \tag{2.9}$$

where $\mathbf{w} = (\boldsymbol{w}^*, w_{p+1}) = (w_1, w_2, \cdots w_{p+1} \in \mathbb{R}^{p+1})$ is defined in Section Section 2.1, and $V_1^{(m)}$, $V_2^{(m)}$ are defined in ((A.1)). Now $S(\mathbf{w})$ is strictly convex, so we can using existing

convex program package (CVX in Matlab).

Noticing that the subproblem is non-differentiable convex function, so we can also adapt the sub-gradient method, sub-gradient method is not a descent method, but is much simpler to be applied to non-differential convex problem (81; 8). As shown in the appendix, we use the sub-gradient method to obtain a unique minimizer $\widehat{\mathbf{w}}^{(m)}$; we repeat the process until convergence.

Nonlinear decision rule for optimal ITR

For nonlinear classification, we minimize ((2.4)), following a similar derivation to that used in the previous section of the linear case, the convex subproblem at the $(m+1)$th iteration of the DC algorithm can be solved as

$$\begin{aligned}\hat{\mathbf{w}}^{(m+1)} &= \min_{\mathbf{w}} S(\mathbf{w}) = \min_{\mathbf{w}} \left(S_1(\mathbf{w}) - \langle \mathbf{w}, \nabla S_2(\hat{\mathbf{w}}^{(m)}) \rangle\right) \\ &= \frac{\kappa}{2}\|\boldsymbol{w}^*\|_K^2 + \frac{1}{n}\sum_{i=1}^{n}\left[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) + \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)\right]\frac{|R_i|}{\pi(A_i, \mathbf{X}_i)} \\ &\quad - \mathbf{w}^{*'} V_1^{(m)} - w_{n+1} V_2^{(m)},\end{aligned} \tag{2.10}$$

where $\mathbf{w} = (\boldsymbol{w}^*, w_{n+1}) = (w_1, w_2, \cdots w_{n+1} \in \mathbb{R}^{n+1})$, as defined in Section Section 2.1, and $V_1^{(m)}$, $V_2^{(m)}$ are defined in ((A.3)). and $A_i f(\widetilde{\boldsymbol{X}}_i) = A_i(\sum_{j=1}^{n} w_j K(\mathbf{X}_i, \mathbf{X}_j) + w_{n+1})$, see more details in the Appendix.

2.2 Variable selection

In practice, the dimension of the covariates may be high for estimating optimal ITRs, some of which may not be informative(29), so we remove the redundant variables to improve the predictive performance.

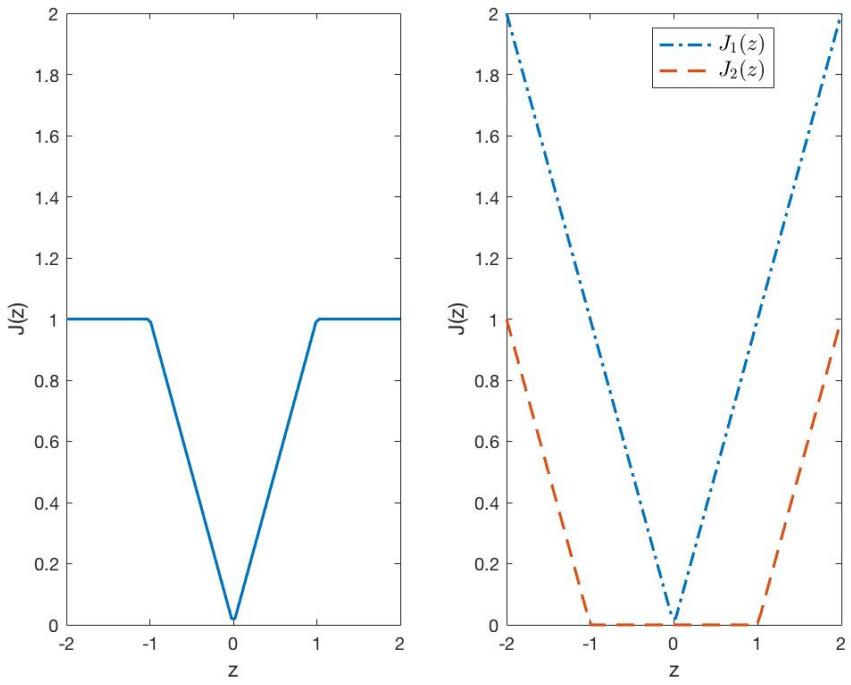

Figure 2.2: Plot of a DC decomposition of the truncated L_1 function $J(z)$ as $J = J_1 - J_2$.

2.2.1 Variable selection for linear classification

There is a large body of literature on variable selection particularly for linear classification problem. For instance, (79) proposed a truncated L_1 penalty (TLP), $J(|z|) = \min(|z|, 1)$ which corrects the Lasso bias through adaptive shrinkage.

Here we use the TLP and a ridge penalty to achieve sparse solutions, expressed as $\frac{\kappa}{2}\|\boldsymbol{w}^*\|^2 + \frac{\lambda}{\tau}\sum_{j=1}^{p}\min(|w_j|, \tau)$, where τ is a tuning parameter to control the degree of approximation, see Figure Figure 2.2 , and λ controls the degree of shrinkage. The TLP penalty is written as a difference of two convex functions :

$$\frac{\lambda}{\tau}\sum_{1}^{p}\min(|w_j|, \tau) = \frac{\lambda}{\tau}\sum_{1}^{p}|w_j| - \frac{\lambda}{\tau}\sum_{1}^{p}\max(|w_j| - \tau, 0)$$

for any $\tau > 0$. Then our cost for the linear case is

$$\frac{\kappa}{2}\|\boldsymbol{w}^*\|^2+\frac{\lambda}{\tau_2}\sum_{j=1}^{p}\min(|w_j|,\tau_2)+\frac{1}{n}\sum_{i=1}^{n}\frac{R_i}{\pi(A_i,\mathbf{X}_i)}\psi\left(A_i f(\widetilde{\boldsymbol{X}}_i)\right), \tag{2.11}$$

where $\psi\left(A_i f(\widetilde{\boldsymbol{X}}_i)\right) = \max(\frac{1-A_i f(\widetilde{\boldsymbol{X}}_i)}{\tau_1}, 0) - \max(\frac{-A_i f(\widetilde{\boldsymbol{X}}_i)}{\tau_1}, 0)$, λ, τ_1 and τ_2 are non-negative tuning parameters to be determined.

The convex subproblem at the (m+1)th iteration is:

$$\begin{aligned}
\hat{\mathbf{w}}^{(m+1)} &= \min_{\mathbf{w}}(S_1(\mathbf{w}) - \langle \mathbf{w}, \triangledown S_2(\mathbf{w}^{(m)})\rangle) \\
&= \min_{\mathbf{w}} \quad \frac{\kappa}{2}\mathbf{w}^{*T}\boldsymbol{w}^* + \frac{\lambda}{\tau_2}\sum_{j=1}^{p}|w_j| + \frac{1}{n}\sum_{i=1}^{n}[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) \\
&+ \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)]\frac{|R_i|}{\pi(A_i,\mathbf{X}_i)} - \frac{\lambda}{\tau_2}\sum_{j=1}^{p}|w_j|I(w_j^{(m)} \geq \tau_2) - \mathbf{w}^{*\prime}V_1^{(m)} - w_{p+1}V_2^{(m)} \\
&= \min_{\mathbf{w}} \quad \frac{\kappa}{2}\mathbf{w}^{*T}\boldsymbol{w}^* + \frac{\lambda}{\tau_2}\sum_{j=1}^{p}|w_j|I(w_j^{(m)} \leq \tau_2) + \frac{1}{n}\sum_{i=1}^{n}[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i > 0) \\
&+ \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i < 0)]\frac{|R_i|}{\pi(A_i,\mathbf{X}_i)} - \mathbf{w}^{*\prime}V_1^{(m)} - w_{p+1}V_2^{(m)},
\end{aligned} \tag{2.12}$$

where $V_1^{(m)}$ and $V_2^{(m)}$ are defined in ((A.1)).

The sub-problem ((2.12)) is convex, which can be solved similarly as in ((2.9)). After solving the subproblem, we update $\hat{\mathbf{w}}^{(m+1)}$ until convergence. The estimated decision function is $\widehat{f}(\mathbf{X}) = \widehat{\boldsymbol{w}^*}'\mathbf{X} + \widehat{w}_{p+1} = \langle \mathbf{w}, \widetilde{\boldsymbol{X}}\rangle$, where $\mathbf{w}$ and $\widetilde{\boldsymbol{X}}$ are defined in Section 2.1, leading to the optimal ITR as $\text{sign}(\widehat{f}(\mathbf{X}))$.

2.2.2 Variable selection for nonlinear classification

(56; 99)pointed out that irrelevant features may effect the performance of kernel methods, and proposed the feature-regularized loss function in the nonlinear kernel methods. The main idea is to penalize the use of features by weighting differently among features. In (2),

features within the kernel are weighted and a lasso penalty is placed on these weights to encourage sparsity, and in (55), it penalized the use of features in the dual formulation of SVM.

In this section, we take the Gaussian kernel as example. The an-isotropic Gaussian kernel is defined as:

$$K_\theta(\mathbf{x}, \boldsymbol{z}) = \exp(-\sum_{j=1}^{p} \theta_j (x_j - z_j)^2)$$

in which the kernel shape is given by $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_p)' \geq 0$. Considering different widths in different dimensions, the importance of feature j is determined by θ_j. For example, if θ_j is very small, the particular variable j loses its importance since its contribution to the kernel function's exponent will be small. On the other hand, if θ_j is very large then the contribution of the variable j to the exponent will be large and proves its importance. After variable selection procedure, if $\theta_j = 0$, it indicates the non significance of covariate j. The original scale parameter σ is absorbed to $\boldsymbol{\theta}$. By incorporating the TLP penalty, we propose the following cost function to minimize:

$$\begin{aligned} \min_{\mathbf{w},\theta} \quad & \lambda_1 \sum_{j=1}^{p} \min(|\theta_j|, \tau_2) + \frac{\kappa}{2} \sum_{i,j=1}^{n} w_i^* w_j^* K_\theta(\mathbf{X}_i, \mathbf{X}_j) \\ & + \frac{1}{n} \sum_{i=1}^{n} \frac{R_i}{\pi(A_i, \mathbf{X}_i)} \psi \left(A_i (\sum_{j=1}^{n} w_j^* K_\theta(\mathbf{X}_i, \mathbf{X}_j) + w_{n+1}) \right) \\ & s.t. \quad \theta \geq 0. \end{aligned} \tag{2.13}$$

where λ_1,κ, and τ_2 are tuning parameters and τ_1 is the tuning parameter in the ψ function. When $\theta = 0$, ((2.13)) can produce sparse solutions for the corresponding feature. Then the

subproblem at the $(m+1)$th iteration is :

$$\begin{aligned}\min_{\mathbf{w},\theta} \quad & \frac{\lambda}{\tau_2}\sum_{j=1}^{p}|w_j| + \frac{\kappa}{2}\sum_{j,j=1}^{n} w_i^* w_j^* K_\theta(\mathbf{X}_i,\mathbf{X}_j) \\ & + \frac{1}{n}\sum_{i=1}^{n}[\psi_1(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i>0) + \psi_2(A_i f(\widetilde{\boldsymbol{X}}_i))I(R_i<0)]\frac{|R_i|}{\pi(A_i,\mathbf{X}_i)} \\ & - \frac{\lambda}{\tau_2}\sum_{j=1}^{p}|w_j| I(w_j^{(m)} \geq \tau_2) - \mathbf{w}^{*'}V_1^{(m)} - w_{n+1}V_2^{(m)},\end{aligned} \tag{2.14}$$

with $f(\widetilde{\mathbf{X}}_i) = \sum_{j=1}^{n} w_j K_\theta(\mathbf{X}_i,\mathbf{X}_j) + w_{n+1}$, where $\mathbf{w} = (\boldsymbol{w}^*, w_{n+1}) = (w_1, w_2, \cdots w_{n+1} \in \mathbb{R}^{n+1})$ is defined in Section 2.1, and $V_1^{(m)}$, $V_2^{(m)}$ are defined in ((A.3)). For this subproblem, is still non-convex. This subproblem is nonconvex, we still apply a similar iterative procedure as in the DC algorithm, i.e. solving a sequence of subproblems using the Majorization-Minimization (MM) algorithm. After solving the subproblem ((2.14)), we update $\hat{\mathbf{w}}^{(m+1)}$. The procedure is repeated until convergence. The estimated decision function is $\widehat{f}(\widetilde{\mathbf{X}}_i) = \sum_{j=1}^{n} \hat{w}_j K_\theta(\mathbf{X}_i,\mathbf{X}_j) + \hat{w}_{n+1}$, where $\mathbf{w}$ and $\widetilde{\boldsymbol{X}}$ are defined in Section 2.1, leading to the optimal ITR as $\text{sign}(\widehat{f}(\widetilde{\boldsymbol{X}}))$.

Chapter 3

Deep Reinforcement Learning(DRL) Based Method

3.1 RL related work

In recent years, RL has been adapted to develop powerful solutions in a variety of healthcare domains characterized as sequential procedures.(17; 101; 114; 74; 25; 51) For example, Dynamic Treatment Regimes (DTR) are tailored for generating new scientific hypotheses and developing optimal treatments across or within groups of patients by utilizing data generated, for instance, from the Sequential Multiple Assignment Randomized Trials (SMART) (65); the design of DTR can be viewed as a sequential decision making problem that fits into the RL framework. The domains of applying RL can be classified into two main categories: chronic diseases and critical care. Due to the long term treatment of a chronic disease, RL has been utilized to automate the discovery and generation of optimal DTRs in a variety of chronic diseases, including cancer, diabetes, anemia, HIV and mental illnesses. For example, (108; 36; 111) applied RL to determine optimal chemotherapy dosages for cancer treatment, and (100; 88) adopted RL in decision making for optimal scheduling of radiation therapy for cancer. In the critical care domain, RL has been applied, for example, to treatment selection for sepsis, (67; 98) medication recommendations in critical care units, (93) and management of invasive mechanical ventilation in intensive care units. (68) Nevertheless, how to tailor the existing RL methods to deal with limited biomedical data and to take into consideration

of risk, safety, robustness and efficiency in healthcare systems remain challenging and to be addressed when applying RL.

3.1.1 Reinforcement Learning: a Brief Review

Recently reinforcement learning (RL) has shown great potential in many challenging applications that require dynamic modeling and long term planning, such as game playing, (61) real-time ads bidding. (11) It has also been introduced into recommender systems. (77; 20; 35; 113; 107)There are two main categories of RL commonly used: model-based RL and model-free RL. (83) In model-based RL, the model always refers to how the environment reacts to certain actions, or how the transition is made in the environment. Model-based RL, such as POMDP, (77) needs to model the dynamic transition, and may not be suitable for complicated recommendation scenarios when the number of candidate drugs is large but with only few cell-lines, as the update of dynamic programming step is too time-consuming. For model-free RL, no dependency on the model of the transition during learning is needed. It can be further divided into two sub-categories: value-based (113) and policy-based. (35; 107) The value-based approaches compute Q-values of all available actions for a given state, and the selection is based on the evaluation of overall actions. Although value-based methods present many advantages such as seamless off-policy learning, they are known to be prone to instability with function approximation. (84)

Despite the practical success of many value-based approaches such as deep Q-learning, (61) policy convergence of these algorithms are not well-studied. These approaches may become inefficient with a large action space. For policy-based approaches, it does not estimate the transition probability and does not store the Q-values; instead, they aim to generate a policy, of which the input is a state, and the output is an action. Policy-based approaches, on the other hand, remain rather stable with respect to function approximations given a sufficiently small learning rate. However, the popular Policy Gradient method only uses one sample each step, and after updating the policy, the sample is removed, thus it disables the reuse of samples. The way that the agent grasps the optimal policy and uses the same to act

is also referred to as "on-policy" RL. The policy that is used for updating and that used for acting is the same. In this way, every time we simulate a cell-line's trajectory, it can only be used once and then is discarded, which may result in high variances and gradient explosion. This is also mentioned in training a vanilla policy gradient algorithm REINFORCE (83) with neural networks, as a model with only 46 weight parameters requires more than 10000 epochs to converge, and we have tried with simulated data even with a sample size larger than 1M, the variance is still large; when the sample size $n = 1000$ in simulations, the model's weights might become "NaN" values during training.

Since this form of gradient has a potentially high variance, a baseline function is typically introduced to reduce the variance whilst not changing the estimated gradient. (83) A natural candidate for this baseline is the state value function. This is the basic framework of Actor-Critic, (83) which can be easily used to learn high dimensional or even continuous actions. Typically, we use an Actor Network (Policy Network) to account for policy improvement, where the Actor Network takes the cell-line features (as well as the drug features if any) as input and generates the scores for each action (i.e. each drug), and outputs an action according to the scores, and we update the policy parameters from the Actor Network. Furthermore, the Actor Network also outputs a state representation and feeds it to a Critic Network. We use the Critic Network to evaluate the current policy, where we fit the Critic Network to estimate the state values. We have tried to use a simple linear function to learn the interactions of cell-lines and drugs, which converges fast and performs well on test data. But it fails to capture non-linear relationships. In order to achieve a stable and convergent algorithm, we leverage deep reinforcement learning (DRL) with adapted artificial neural networks as non-linear approximators to estimate the value function. However, even with the on-policy Actor-Critic framework, the algorithm still cannot get the most out of every sample, namely, it is sample inefficient. In order to improve sample efficiency, we adapt the Proximal Policy Optimization (PPO) algorithm, which enables us to alternate between sampling data from the policy and performing several epochs of optimization on the sampled data. This has been shown to be significantly better in more efficient use of the samples.

More importantly, to account for possible patient (or cell-line) preference/condition changes, we use a GRU layer to memorize the previous preferences (or conditions), and update the state if the cell-line's evaluation signal is positive. The state representation learned by the Actor Network is also the input to the Critic Network, which is used to evaluate the policy. Furthermore, without any approximation to the targeted metric as the loss function, policy-based approaches exactly use the gradient methods for policy optimization, without considering the discretization of the loss function.

3.2 Personalized Ranking

Given N cell lines (or patients), we observe $\mathbf{X} \in \mathbb{R}^{N \times P}$, with each row and column representing one cell line and one feature, e.g., one gene's expression, respectively; cell line-specific responses, denoted as $\mathbf{Y} \in \mathbb{R}^{N \times M}$, are their responses to (maximum) M distinct drugs. Each of the N cell lines $\left\{c^{(i)}\right\}_{i=1}^{N}$ is paired with its drug response vector $\mathbf{Y}^{(i)} = \left\{y_1^{(i)}, \cdots, y_{M_i}^{(i)}\right\}$, where M_i is the number of candidate drugs for cell line $c^{(i)}$. For each cell line $c^{(i)}$, together with $\left\{\left(\mathbf{X}^{(i)}, \mathbf{D}^{(i)}, \mathbf{Y}^{(i)}\right)\right\}$, where $\mathbf{X}^{(i)} \in \mathbb{R}^P$ is the corresponding cell line feature vector, $\mathbf{D}^{(i)} = \left\{\mathbf{d}_1^{(i)}, \cdots, \mathbf{d}_{M_i}^{(i)}\right\}$ is a set of the drugs tested on cell line; each drug $\mathbf{d}_j^{(i)} \in \mathbb{R}^D$ with $j \in \{1, \cdots M_i\}$ denotes a drug feature vector, either a vector representing the drug structure or only an indicator for the drug's index when the drug feature vector is unused/unknown. In the ranking process, we use $\mathbf{D}_t^{(i)}$ as the remaining candidate drugs to be ranked at time step t for cell line i, so $\mathbf{D}_t^{(i)}$ is a subset of $\mathbf{D}^{(i)}$. Table 3.1 lists main mathematical notations used throughout this paper. The main PPORank algorithm and its episode-sampling component are also described as two Algorithms.

We formulate the ranking problem as learning a scoring function that scores each candidate drug at each ranking position $t \in \{1, 2, \cdots, M\}$. The scores usually represent some relevance, inducing an ordering of the drugs by sorting them in descending order of relevance to form a ranked list at each position. When moving to the next position, according to the dynamic change of the candidate drugs, the context environment, and the corresponding feature vectors, the scores for the remaining drugs may change under the scoring function.

Table 3.1: Mathematical notations.

Meaning	Notation
Cell-line	c, or $c^{(i)}$ *
Number of training cell lines	N
Maximum number of drugs for each cell line	M
Feature vector of cell line $c^{(i)}$	$X^{(i)}$ or X
Drug u is more sensitive than drug v	$d_u \succ d_v$
Number of drugs associated with cell line $c^{(i)}$	M_i
Feature vector or binary features of a drug j associated with cell line $c^{(i)}$	$\mathbf{d}_j^{(i)}$ or $\mathbf{d}_j$
Embedding vector of a drug j associated with cell line $c^{(i)}$	$\mathbf{d}_{emb,j}^{(i)}$ or $\mathbf{d}_{emb,j}$
Feature vectors or binary indexes of drugs associated with cell line $c^{(i)}$	$D^{(i)} = \{\mathbf{d}_j^{(i)}\}_{j=1}^{M_i}$
Remaining drugs associated with cell line $c^{(i)}$ at time step t	$D_t^{(i)}$ or D_t
Ground truth response score of a drug j associated with cell line $c^{(i)}$	$y_j^{(i)}$ or y_j
Ground truth permutation(ranking list) for drugs associate with cell line c	π^*
Original drug index of the j-th element in permutation π	$\pi^{-1}(j)$
Ranking position of drug j in permutation π	$\pi(j)$

*the superscript $\{i\}$ indicates the associate cell line $c^{(i)}$, if it is missing, it can be applied to general case that is associated with cell line c

As such, the goal of the ranking is to build a parameterized ranking function $f(\mathbf{X}, \mathbf{D}_t, t)$ that minimizes the empirical loss across all the ranking positions t:

$$\mathcal{L}(f, N) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{t=1}^{M} l(\mathbf{Y}, f(\mathbf{X}^{(i)}, \mathbf{D}_t^{(i)}, t)), \tag{3.1}$$

where $l(\cdot)$ is a specified local loss function. In ranking problems, $l(\cdot)$ is usually derived from the utility of interest such as the discounted cumulative gain (DCG) metric, (39) and its normalized form called normalized discounted cumulative gain (NDCG). Formally, given a ranking list of the drugs for a cell line, say π, the DCG at position k is defined:

$$\text{DCG@}k = \sum_{t=1}^{k} G\left(f, \pi^{-1}(t)\right) \eta(t), \tag{3.2}$$

where $\pi^{-1}(t)$ denotes the drug ranked at position t by the ranking list π, $G(\cdot)$ is the response score of a drug, usually set as $G\left(f, \pi^{-1}(t)\right) = 2^{f_{\pi^{-1}(t)}} - 1$ with $f_{\pi^{-1}(t)}$ being the response

score learned from the ranking function $f(\mathbf{X}, \mathbf{D}_t, t)$ at position t in the ranking list π, and $\eta(t)$ is a position discount factor, usually set as $\eta(t) = 1/\log_2(t+1)$. In the end, DCG@k represents the cumulative gain of information from position one to position k with discounts on the positions. DCG@k gives a higher weight to a drug with a higher sensitivity score, while the discount factor penalizes the incorrect ordering of these k rankings; the gains can be maximized when the most effective k drugs are the top k recommended ones.

As an evaluation criterion, the normalized discounted cumulative gain (NDCG) is more widely used. It is defined as NDCG@$k = \frac{1}{Z_k}\sum_{t=1}^{k} G\left(f, \pi^{-1}(t)\right)\eta(t)$, where Z_k is a normalizing factor calculated at the perfect ranking list π^* based on the true response scores; NDCG@k ranges from 0 to 1, with 1 indicating a perfectly predicted ranking. The final evaluation metric for top k positions is NDCG@$k = \frac{1}{Z_k}\sum_{t=1}^{k} \frac{2^{f_{\pi^{-1}(t)}}-1}{\log_2(t+1)}$. When it comes to rank at all the positions, the criterion is

$$\text{NDCG} = \frac{1}{Z_M}\sum_{t=1}^{M} \frac{2^{f_{\pi^{-1}(t)}}-1}{\log_2(t+1)}, \tag{3.3}$$

where M is the number of all the candidates (drugs) to be ranked (for a cell line), and Z is the normalizing factor.

Given N samples $\left\{\left(c^{(i)}, \mathbf{X}^{(i)}, \mathbf{D}^{(i)}, \mathbf{Y}^{(i)}\right)\right\}_{i=1}^{N}$, (minus) the loss function to be maximized for a given ranking function $f(\mathbf{X}^{(i)}, \mathbf{D}_t^{(i)}, t)$ across all ranking positions is

$$\mathcal{L}_{NDCG}(f, N) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{M_i Z_{i,M_i}} \sum_{t=1}^{M_i} \frac{2^{f_{\pi^{-1}(t)}}-1}{\log_2(t+1)}, \tag{3.4}$$

where M_i is the number of candidate drugs for cell line i. Note that $\mathbf{Y}$ in general contains missing values since a cell line may be given only a (proper) subset of all candidate drugs; that is, M_i may differ across the cell lines.

3.3 Reinforcement Learning for Ranking

We formulate a personalized ranking process as a Markov Decision Process (MDP). (95) For each cell line, the ranking process on the candidate drugs is considered as a sequence of discrete-time steps; each time step corresponds to a ranking position, e.g. ranking position t is considered as the same as the time step t. As the candidate drug number varies across the cell lines, the length for each ranking process is determined by the number of available drugs. At each time step, the agent receives the environment's state and chooses an action based on the state. The state is constructed by the remaining candidate drugs, the cell line, and a list of the previously ranked drugs, including the candidate drug and cell line features, drug-drug, and cell line-drug interactions. The action of drug selection only depends on a policy, which is a function mapping from the current state to a probability distribution of selecting possible drugs. After one time step, the environment transits to a new state according to the current state with the cell line evaluation signal and the chosen action.

The ranking problem modeled as a MDP in the framework of RL can be represented by $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \pi, \gamma \rangle$, composed of states, actions, transition, reward and policy as defined below:

States $\mathcal{S}$ is a set of states, each describing the environment. In the ranking problem, the agent should be aware of the current ranking position as well as the candidate drug set that the agent can choose from. For simplicity, we only describe the notation under one cell line; if it comes to batch computation, we just need to include multiple cell lines. Thus, at time step t, we have observation $\mathbf{O}_t = [\mathbf{X}, \mathbf{D}_t, \mathbf{H}_t]$, and the state $\mathbf{s}_t$ is defined as

$$\mathbf{s}_t = f_s(\mathbf{O}_t) = f_s([\mathbf{X}, \mathbf{D}_t, \mathbf{H}_t]), \tag{3.5}$$

where$f_s(\cdot)$ stands for a state representation module, $\mathbf{X}$ is the cell line, D_t is the remaining drugs to be ranked, and $\mathbf{H}_t$ is a hidden state learned from previous positive evaluation signals; for example, if the cell line provides a positive response (i.e. the immediate reward is positive), then the next hidden state is updated to $\mathbf{H}_{t+1} = \text{GRU}(\mathbf{X}, \mathbf{D}_t, \mathbf{H}_t)$; otherwise,

$\mathbf{H}_{t+1} = \mathbf{H}_t$. The initial state can be written as $\mathbf{s}_1 = f_s(\mathbf{O}_1) = f_s([\mathbf{X}, \mathbf{D}, \mathbf{H}_1])$, where $\mathbf{D}$ is the set of all the candidate drugs, and the hidden state is initialized to be $\mathbf{0}$.

Actions $\mathcal{A}$ is a discrete set of actions that the agent can take. The set of possible actions depends on the state $\mathbf{s}_t$, denoted as $\mathcal{A}(\mathbf{s}_t)$. At time step t, $a_t \in \mathcal{A}(\mathbf{s}_t)$ is the selection of a drug $\mathbf{d}_{m(a_t)} \in D_t$ for the ranking position t, where $m(a_t)$ indexes the drug selected by a_t.

Transition $\mathcal{T}(\mathcal{S}, \mathcal{A})$ is a function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, mapping a state $\mathbf{s}_t$ to a new state according to action a_t. When choosing a drug by action a_t, it means removing the drug $\mathbf{d}_m(a_t)$ from the candidate set, then $\mathbf{s}_{t+1} = \mathcal{T}([\mathbf{X}, \mathbf{D}_t, \mathbf{H}_t]) = f(\left[\mathbf{X}, \mathbf{D}_t \backslash \left\{\mathbf{d}_{m(a_t)}\right\}, \mathbf{H}_{t+1}\right])$.

Reward $r(\mathcal{S}, \mathcal{A})$ is the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. As our target in ranking is to maximize NDCG, the reward should be able to reflect the contribution to the evaluation metric of selecting the drug. Thus we define the reward received corresponding to choosing action a_t as the promotion of the DCG (15):

$$r_{\mathrm{DCG}}(\mathbf{s}_t, a_t) = \begin{cases} 2^{y_{m(a_t)}} - 1 & t = 1 \\ \frac{2^{y_{m(a_t)}} - 1}{\log_2(t)} & t > 1 \end{cases}, \tag{3.6}$$

where $y_{m(a_t)}$ is the drug sensitivity score $\mathrm{y} = -\log(\mathrm{IC}_{50})$ for the selected drug $m(a_t)$ at position t (in our two cell line data examples). Here we see the immediate reward at each time step t is just the promotion of DCG at each ranking position t, which enables the learning process to utilize DCG values at each time step. Furthermore, for each cell line, the cumulative rewards along a trajectory will be $\mathrm{DCG} = \sum_{t=1} r_{\mathrm{DCG}}(\mathbf{s}_t, a_t)$, which we aim to maximize.

Policy $\pi(\mathbf{a}|\mathbf{s}) : \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the behaviors of the agent, which is a probabilistic distribution over the possible actions for a given state. Specially, we use the softmax function to parametrize the policy, outputting the probability of selecting each of the drugs for the current ranking position:

$$\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) = \frac{e^{\phi(\mathbf{s}_t, \mathbf{a}_t)^\top \boldsymbol{\theta}}}{\sum_{\mathbf{a} \in A(\mathbf{s}_t)} e^{\phi(\mathbf{s}_t, \mathbf{a})^\top \boldsymbol{\theta}}}, \tag{3.7}$$

where $\boldsymbol{\theta}$ is the policy parameters in the actor network, $\phi(\mathbf{s}_t, \mathbf{a}_t)$ is the feature vector decided by the current state and action, and $\top$ denotes the transpose. More details can be found in section Section 3.3.8.

Discount factor γ, $\gamma \in [0, 1]$, defines the discount factor when we measure the present value for future reward. In particular, when $\gamma = 0$, the agent only considers the immediate reward; alternatively, when $\gamma = 1$, all future rewards can be counted fully into that of the current action. It is typical to have $\gamma \in (0, 1)$.

Without loss of generality, a list-wise ranking procedure is a T-time step trajectory as $(\mathbf{s}_1, a_1, \cdots, \mathbf{s}_T, a_T)$. In this way, we define one sample as a tuple of $\langle \mathbf{s}_t, a_t, r(\mathbf{s}_t, a_t), \mathbf{s}_{t+1} \rangle$. Although we may only observe one trajectory for one cell line, we can simulate multiple and different trajectories for each cell line.

3.3.1 Some Preliminaries for DRL

The goal of RL is to maximize the expected long-term return from an initial state in terms of τ:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)}[G(\tau)], \tag{3.8}$$

where $\pi_\theta(\tau) = \pi_\theta(\mathbf{s}_1, a_1, \ldots, \mathbf{s}_T, a_T) = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid \mathbf{s}_t) p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t)$ is the distribution over the trajectory, which only relates with the parameters in policy. $G(\tau)$ is the long-term return of the simulated ranking list, which is defined as the discounted sum of the rewards:

$$G(\tau) = \sum_{t=1}^{M} \gamma^{t-1} r_t, \tag{3.9}$$

where $r_t = r(\mathbf{s}_t, a_t)$ is the immediate reward the agent receives at time step t after taking action a_t, and $0 \leq \gamma \leq 1$ is the discount factor. For our task, the maximum time step for each trajectory is the number of the total candidate drugs, M. Note that when the discount factor $\gamma = 1$, $G(\tau)$ is identical to NDCG.

Differentiating $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, (83) the gradient becomes

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left[\left(\sum_{t=1}^{M} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} \left(\mathbf{a}_t | \mathbf{s}_t \right) \right) \left(\sum_{t=1}^{M} \gamma^{t-1} r_t \right) \right], \tag{3.10}$$

which can be approximated by sampling trajectory τ^i:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{M} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} \left(\mathbf{a}_{i,t} | \mathbf{s}_{i,t} \right) \right) \left(\sum_{t=1}^{M} \gamma^{t-1} r_{i,t} \right), \tag{3.11}$$

where $r_{i,t} = r(\mathbf{s}_{i,t}, a_{i,t})$ is the immediate reward of the i-th trajectory at time step t and can be calculated from ((3.6)).

To reduce the variance of the gradients, at time step t, we only consider the cumulative rewards received after t, as the action taken at time t' cannot affect the reward at t for all $t < t'$. Secondly we subtract a baseline from the total rewards. A commonly used baseline is the value function, $V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$, which is the expected return starting in $\mathbf{s}_t$ and following policy $\pi_\theta(a_t|\mathbf{s}_t)$ thereafter. Here we use the superscript π_θ denotes that the agent will follow policy $\pi_\theta(a_t|\mathbf{s}_t)$. The state-action value $Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) = E_{\pi_{\boldsymbol{\theta}}(\tau)}[\sum_{t'=t}^{M} \gamma^{t'-t} r(\mathbf{s}'_t, \mathbf{a}'_t)|(\mathbf{s}_t, \mathbf{a}_t)]$, which is the expected return of taking action a_t at state $\mathbf{s}_t$ following policy π_θ thereafter. And by subtracting this baseline from $Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t)$, we are essentially calculating the advantage, $A^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$. Instead of estimating both the state-action value function $Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t)$ and value function $V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$, we approximate $Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_{t+1})$ based on $\mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t) | \mathbf{s}_t, a_t \right] = A^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t)$. Then we build the critic network (also called the value network) to estimate the non-linear approximation of the value function $V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$. In our DNN, limited by the sample size, the critic network will share the parameters with the policy network, so we will also only use $V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$ for the state value output from the critic network. This gives the actor-critic network's structure, (84) which, with the critic, improves over the policy gradient. Denote by $\delta_t^{\pi_{\boldsymbol{\theta}}} = R(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_{t+1}) - V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$. Then, the gradient of the policy parameters becomes:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{s}_t) \delta_t^{\pi_{\boldsymbol{\theta}}} \right], \tag{3.12}$$

and $\delta_t^{\pi_{\boldsymbol{\theta}}}$ serves as an unbiased estimate of the advantage function.

The actor-critic network combines the advantages of value-based learning and policy gradient to achieve accelerated and stable learning. It consists of two components: an actor to optimize the policy $\pi_{\boldsymbol{\theta}}$ in the direction of gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ using ((3.10)), and a critic to estimate a state-value function $V^{\pi_{\boldsymbol{\theta}}}$ with the state output from the actor network. Finally, we obtain the policy gradient in ((3.12)).

3.3.2 Actor network

We describe the architecture of the actor network, inspired by the deep & cross network. (94) A main motivation is for flexible modeling of possibly complex relationships between high-dimensional cell line and drug features. Previous studies (e.g. (91; 82)) have shown the importance of such flexible modeling. This network starts with an embedding layer and a stacking layer, followed by a cross network and a deep network. In the concatenation layer, we combine the output from the two networks and that from a Gated Recurrent Unit (GRU) layer. (13) The concatenated vector from the combination layer is defined as the state in ((3.5)). Before building the embedding layer, we have pre-trained a matrix factorization (MF) model to initialize the MF layer. The structure is displayed in Figure 3.1. Next, we describe the actor network under the setting for the cancer drug response ranking problem.

3.3.3 Matrix Factorization (MF) layer

Due to the heterogeneous data sources, if we model the interactions between cell lines and drugs, we may first project them into the spaces of the same dimension. The easiest way of incorporating drug features is to consider the input as one-hot encoding. Cell-line features can be projected into the same dense embedding layer with drug embedding vector for further computation. We have tried initializing the weight matrix for projecting cell line features randomly but noted that it converges slowly. Inspired by, (105; 30) we decide to use an MF layer for initialization. The input data include dense features such as the cell line gene expression and binary encoding for drugs. The drug response matrix $\mathbf{Y}$ can be decomposed

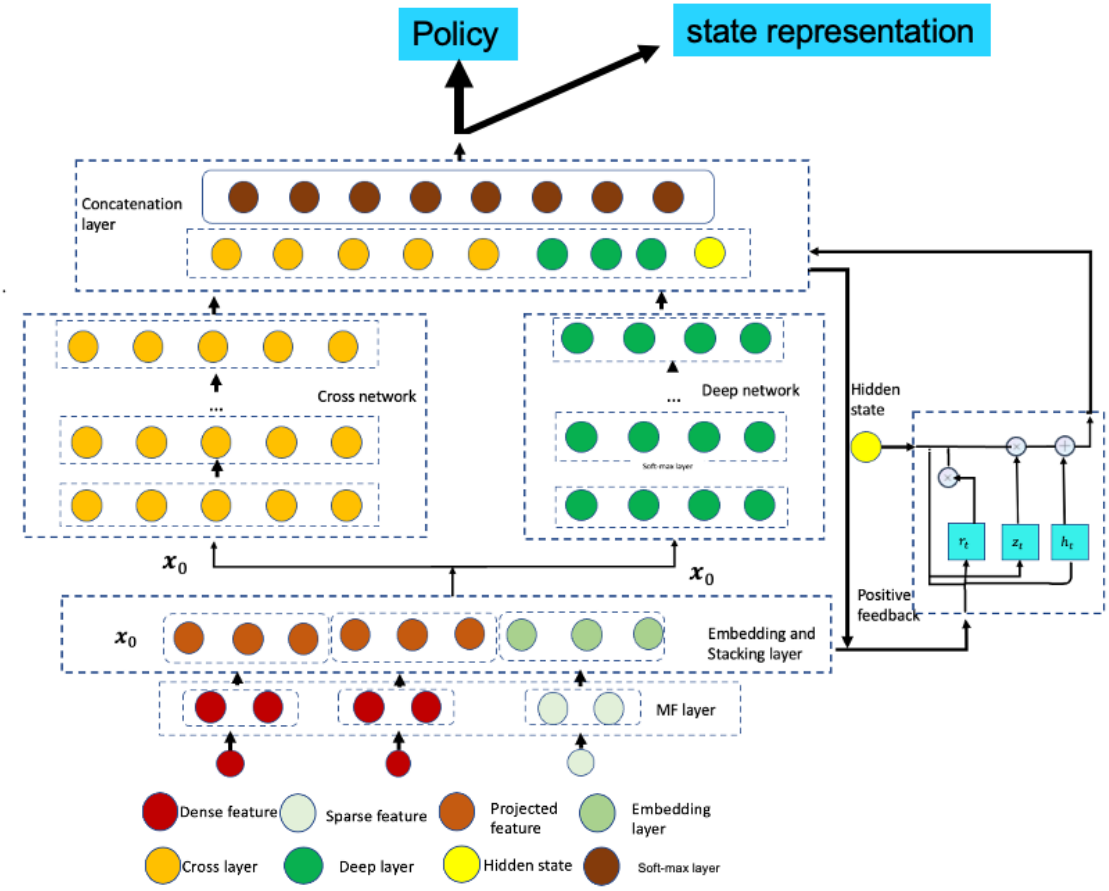

Figure 3.1: The actor network.

into a bias matrix $\mathbf{B}$ and some latent factors for cell lines and drugs:

$$\mathbf{Y} = \mathbf{B} + \mathbf{U}\mathbf{V}^{\mathrm{T}}, \quad \mathbf{U} = \mathbf{X}\mathbf{W}_{\mathbf{c}}, \quad \mathbf{V} = \mathbf{D}\mathbf{W}_{\mathbf{d}}, \tag{3.13}$$

where $\mathbf{B} = (b_1, b_2, \cdots, b_M)^\top \in \mathbb{R}^{N \times M}$, b_i is the bias term for drug i; $i = 1, \cdots, M$, $\mathbf{W_c} \in \mathbb{R}^{P \times f}$ and $\mathbf{W_d} \in \mathbb{R}^{M \times f}$ are the weight matrices projecting the cell line and drug features into the corresponding latent spaces with dimension f respectively, and are both trainable in a neural network, $\mathbf{D} \in \mathbb{R}^{M \times M}$ which is the identity matrix. Thus $\mathbf{U} = (u_1, u_2, \ldots, u_N)^\top \in \mathbb{R}^{N \times f}$ denotes the cell lines preference matrix, $\mathbf{V} = (v_1, v_2, \ldots, v_M)^\top \in \mathbb{R}^{M \times f}$ denotes the drugs' preference matrix. Therefor, the output off the MF layer is the projected dense features $\mathbf{U}$ for each cell line and the embedding matrix $\mathbf{V}$ for the each drug.

3.3.4 Embedding and stacking layer

The input of this layer comes from the output of the MF layer. Then we stack the embedding vectors for each drug along with the dense features of cell lines, together with the cosine similarity score between the cell line's and drugs' latent vectors, into one vector:

$$\mathbf{x}_0 = \left[u^\top, v_1^\top, v_2^\top, \cdots, v_M^\top, \cos\right], \tag{3.14}$$

where u is projected dense features, $v_1, v_2, \cdots, v_M$ is from ((3.13)) and *cos* is the cosine similarity measure between the cell line and the drug. Then we have $\mathbf{x}_0 = (x_1, x_2, \cdots x_d)^\top \in \mathbb{R}^d$, $d = 1 + Mf + f + M$, and x_j is the j-th element of $\mathbf{x}_0$. Compared to the Word2Vec (59) embedding, we explicitly use the response matrix and the cell line features, which not only generates an embedding for the drug and cell line in the same latent space but also compresses the cell line features.

Importantly, we use the stacking of all the drugs' embedding vector, which is used as the initial state $\mathbf{s}_1$. At time step t, if drug j is removed from the candidate drugs, we will use a zero vector $(0, 0, \cdots, 0) \in \mathbb{R}^f$ to replace v_j. It is noted that, although we only use the one-hot encoding vector for a drug as the sparse vector, sometimes it may not be sufficient,

and we can use other side information for drugs. For example, some drug similarity scores from PubChem (41) may be used as additional dense features and be projected onto the lower-dimensional dense layer.

3.3.5 Cross network

For MF-based ranking methods, it is possible to capture the linear relationship between the latent vectors, but not higher-order cell line-drug interactions based on the MF latent factor representation. To capture a higher-order interactions, we add additional layers in a network to represent interactions as cross-term features. We denoted the cross term (monomial) as $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$, with degree $|\boldsymbol{\alpha}| = \sum_{i=1}^{d} \alpha_i > 2$.

To capture higher order interactions, we have the cross network composed of cross layers, where the input is $\mathbf{x}_0$ from the stacking and embedding layer in ((3.14)), and with each layer having the following formula:

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \mathbf{x}_l^\top \mathbf{w}_l + \mathbf{b}_l + \mathbf{x}_l = f\left(\mathbf{x}_l, \mathbf{w}_l, \mathbf{b}_l\right) + \mathbf{x}_l, \tag{3.15}$$

where $\mathbf{x}_0$ is the output from ((3.14)), $\mathbf{x}_l, \mathbf{x}_{l+1}$ are the column vectors as the outputs from the l-th and $(l+1)$-th cross layers, respectively; $\mathbf{w}_l, \mathbf{b}_l$ are the weight and bias parameters in the l-th layer. By (94), the highest degree of cross terms grows with the layer depth.

It is noted that the cross network is a generalization of MF and so-called Factorization Machines (94) by allowing cross terms of any degrees.

In addition, by (94) the number of parameters in the cross network is only $d \times L_c \times 2$, where L_c is the number of cross layers. And the output of the cross network is $\mathbf{x}_{L_c}$. Accordingly, when we integrate multi-omic and other high-dimensional features, the number of parameters in a network model only grows linearly in the number of layers.

3.3.6 Deep network

The deep network is a fully-connected feed-forward neural network, which is good at generalization. Its input is the same as the cross network, which is also $\mathbf{x}_0$ from the stacking and embedding layer in ((3.14)), and each layer is represented as

$$\mathbf{H}_{l+1} = f\left(W_l \mathbf{H}_l + \mathbf{b}_l\right), \tag{3.16}$$

where $\mathbf{H}_l \in \mathbb{R}^{n_l}$ and $\mathbf{H}_{l+1} \in \mathbb{R}^{n_{l+1}}$ are the l-th and $(l+1)$-th hidden layers respectively; $W_l \in \mathbb{R}^{n_{l+1} \times n_l}$ and $\mathbf{b}_l \in \mathbb{R}^{n_{l+1}}$ are the weight matrix and bias vector (as the parameters) for the l-th layer; and $f(\cdot)$is the ReLU function. Suppose we have L_h layers for the deep network, then the output is $\mathbf{H}_{L_h}$ with the initial input $\mathbf{x}_0$. In this fashion, the cross and deep components share the same input, leading to more efficient training, c.f., .(30)

3.3.7 GRU layer

In clinical settings, patients' preference or conditions (not captured by given covariates) may change over time, thus we propose using evaluation signals from the ranking process to learn the dynamic hidden preference. The positive drugs represent key information about patients' preferences, i.e., which drugs the patients prefer (e.g. without minimal side effects). Thus we propose using a GRU layer to memorize the previous evaluation signals, allowing it to remember values over multiple iterations in the sequence. So if the evaluation signal is positive, we can update the GRU layer as the following:

$$\begin{aligned}
\mathbf{z}_t &= \sigma\left(W_z \mathbf{x}_t + U_z \mathbf{H}_{t-1} + b_z\right), \qquad \mathbf{re}_t = \sigma\left(W_r \mathbf{x}_t + U_r \mathbf{H}_{t-1} + b_r\right), \\
\mathbf{H}_t &= GRU\left(\mathbf{H}_{t-1}, \mathbf{x}_t\right) = \mathbf{z}_t \circ \mathbf{H}_{t-1} + (1 - \mathbf{z}_t) \circ \tanh\left(W_h \mathbf{H}_{t-1} + U_h\left(\mathbf{re}_t \circ \mathbf{H}_{t-1}\right) + b_h\right),
\end{aligned}$$

where $\mathbf{z}_t$ is the update vector, $\mathbf{re}_t$ is the reset gate vector, $\mathbf{x}_t$ is the concatenated output from the deep network and cross network at time step t, $\mathbf{H}_t$ is the hidden state at time step t, $\circ$ is the Hadamard product and the matrices W_z, W_h, W_r, and vectors b's are the unknown weights to be optimized, σ is the ReLU activation function. The initial hidden state is set

as **0**. Then if the current evaluation signal is positive, the output of the hidden state is incorporated into the next combination output layer. We tried to use a pooling layer to represent the dynamic evaluation signal, which gives equal weight to the cell line evaluation signals in the previous time steps; but with the large action space, we noticed that the top 1 drug would dominate the historical evaluation signals, and if the top 1 recommendation was far away from the truth, the actor network would have no chance to adjust the learning process.

3.3.8 Concatenation layer

The concatenation layer concatenates the outputs from the two networks and the GRU layer. We firstly output the concatenated vector as a state representation to input the critic network, and secondly apply a standard softmax activation function, yielding the output for policy:

$$\pi\left(\mathbf{a}_t|\mathbf{s}_t\right) = \frac{\exp\left\{\boldsymbol{\theta}_{logits}^{\top}\mathbf{z}_{m(a_t)}\right\}}{\sum_{a\in A(s_t)}\exp\left\{\boldsymbol{\theta}_{logits}^{\top}\mathbf{z}_{m(a_t)}\right\}}, \tag{3.17}$$

where $\mathbf{z}_{m(a_t)} = \left[\mathbf{x}_{L_c}^{\top}, \mathbf{H}_{L_h}^{\top}, \mathbf{H}_t\right]$ is the concatenated vector, $\mathbf{x}_{L_c}^{\top}, \mathbf{H}_{L_h}^{\top}$ are the outputs from the cross network and deep network, and $\mathbf{H}_t$ is the evaluation signal output from the GRU layer. $m(a(t))$ is the drug index, $\boldsymbol{\theta}_{logits}$ is the weight vector for the output layer.

During the training phase of PPORank (as many RL algorithms), the action is often selected according to the current policy estimate; however, to explore (and learn) other potentially more rewarding new actions, it selects an action randomly from the estimated policy distribution. On the other hand, in the testing phase, we just rank the drugs by their predicted scores and select the best one.

3.3.9 Critic network

The critic network is designed to reduce the variance of the gradients in the policy gradient method, which leverages a deep neural network to approximate the true state value function

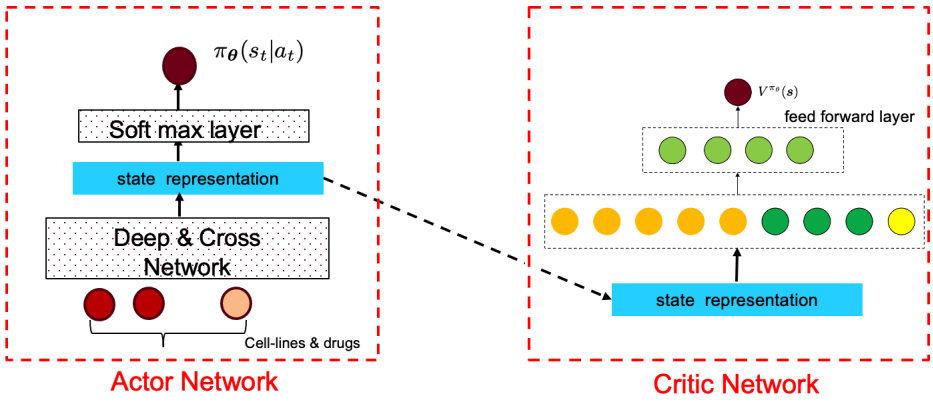

Figure 3.2: The actor and critic networks.

$V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$. The input of the critic network is just the state generated from the actor network in ((3.17)), and its output is the state value function $V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$. The critic network is jointly learned with the actor network, with the shared state representation as the input from the output of the actor network. We have tried a separate network design, resulting in the divergence of the policy with too many parameters to learn; it is a good way to stabilize training with the shared parameters. The critic is then used to evaluate the policy generated by the actor network and guides the policy gradient. The structure can be found in Figure Figure 3.2.

3.4 Learning to Rank with Proximal Policy Optimization (PPO)

Actor-Critic combines the advantage of value-based methods and policy gradient to achieve accelerated and stable learning, but it still needs further modifications for the dynamic ranking problem. On the other hand, considering the sample complexity, as the on-policy algorithm requires collecting new samples whenever the policy changes, the old sample is not reusable. A good way to make use of old samples is to use importance sampling (IS), in which we can use samples from the old policy to calculate the policy gradient. But if

we sample under the old policy to calculate the expected long-term rewards under the new policy, and if the ratio of $\frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}_{old}}}$ is high, the variance of the estimate may still explode. Thus we still need to re-sample trajectories frequently using the current policy.

We also notice that with the increasing training time, the scores for sensitive drugs will continue increasing, resulting in a final failure. One of the reasons is that the gradient depends on the policy parameterization, instead of the actual policy, as in our case with a softmax policy, the policy can often be reparameterized without changing action probabilities. The steepest ascent in the parameter space does not guarantee the steepest ascent in the policy space. In particular, when the sample size is small, it may not be able to recover from the previous bad policy and thus will collect data under the bad policy.

Hence in our policy optimization algorithm, we want an update step that uses rollouts collected from the most recent policy as efficiently as possible, and takes steps that respect the distance in the policy space as opposed to in the parameter space, which aims to control the changes in policy. In order to stabilize the learning process and improve the sample efficiency, we use the ideas from the trust region policy optimization (TRPO), (73) which bounds the distribution to use the KL divergence and proximal policy optimization (PPO) (74) that clips on the probability ratio. To implement simply, we mainly design the algorithm with PPO.

By bounding the policy distribution change, we are able to optimize the expected advantage under the old policy, which aims to get an improved policy from optimization on the sampled data from the old policy $\pi_{\boldsymbol{\theta}_{old}}$. Here, we use the clip ratios as the constraint for the policy probability change, then the clipped "surrogate" objective is:

$$L_t^{CLIP}(\boldsymbol{\theta} = \hat{\mathbb{E}}_t \left[\min \left(r_t(\boldsymbol{\theta} \hat{A}_t^{\pi_{\boldsymbol{\theta}_{old}}}, \text{clip} \left(r_t(\boldsymbol{\theta}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t^{\pi_{\boldsymbol{\theta}_{old}}} \right) \right], \tag{3.18}$$

where $\hat{\mathbb{E}}_t[\ldots]$ indicates the empirical average over a finite batch of samples at time step t, and $\hat{A}^{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{s}_t, \mathbf{a}_t)}$ is an estimate of the advantage function at time step t from samples following $\pi_{\boldsymbol{\theta}_{old}}$, and $r_t(\boldsymbol{\theta} = \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}_{\boldsymbol{\theta}_{old}}}(\mathbf{a}_t|\mathbf{s}_t)}$ is the probability ratio, ϵ is the clipping parameter by limiting $r_t(\boldsymbol{\theta}$ within the interval $[1-\epsilon, 1+\epsilon]$.

As we can see, the clipping serves as a regularization by removing incentives for the policy to change dramatically, and the hyperparameter ϵ corresponds to how far away the new policy can go from the old one while still increasing the objective.

As our state is constructed from the cell-line features (and candidate drug features), so we choose to share the parameters between the policy and value networks, denoted as $\boldsymbol{\theta}$. Similar to that in Actor-Critic, we use the supervised value loss function to evaluate the policy, and notice that the value function also explodes as the last layer of the Critic Network is a linear layer, which is not updated with the previous parameters, so we also add a clip penalty on it, then the clipped value loss is:

$$L_t^{VF}(\boldsymbol{\theta} = \min\left[\hat{\mathbb{E}}_t\left(V^{\pi_{\boldsymbol{\theta}}}\left(\mathbf{s}_t\right) - V_t^{\text{targ}}\right)^2, \hat{\mathbb{E}}_t\left(\text{clip}\left(V^{\pi_{\boldsymbol{\theta}}}, V^{\pi_{\boldsymbol{\theta}_{old}}} - \varepsilon, V^{\pi_{\boldsymbol{\theta}_{old}}} + \varepsilon\right) - V_t^{\text{targ}}\right)^2\right]. \tag{3.19}$$

Here V_t^{targ} is the target value at time step t; as used in supervised learning, the easiest case is $V_t^{\text{targ}} = r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma V^{\pi_{\boldsymbol{\theta}_{old}}}\left(\mathbf{s}_{t+1}\right)$. In our ranking problem, even starting from the same initial state [†]; according to the randomness of policy, we may sample different trajectories, and with the increase of time step t, the variance among these different trajectories will become larger. Furthermore, as the maximum time step is $M = |D|$, we can directly apply the generalized advantage estimation (GAE), (60) which will cut the trajectory before the variance becomes too large. The advantage estimator at time step t is

$$\begin{aligned} \hat{A}_t^{GAE} &= \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{M-1}, \\ &\text{where} \quad \delta_t = r_t + \gamma V\left(\mathbf{s}_{t+1}\right) - V\left(\mathbf{s}_t\right), \end{aligned} \tag{3.20}$$

and M is the maximum steps for each trajectory, which may be different across cell-lines. When applying this advantage function into (3.18), it is calculated based on the sampled batch data from the old policy $\pi_{\boldsymbol{\theta}_{old}}$.

PPO is prone to suffering from lack of exploration, especially with bad initialization,

[†]for every cell-line from which we sample episodes, the initial state is always $f([\mathbf{X}, \mathbf{D}, \mathbf{H}_1])$

which may lead to the failure of training or being trapped in bad local optima. Thus we introduce a supervised signal, the entropy accounting for efficient exploration similar to that used in the original paper, (74) for regularization.

The entropy regularization is

$$S\left(\pi_{\boldsymbol{\theta}}, \mathbf{s}_t\right) = -\left(1 - \hat{I}_t\right) \log\left(1 - \pi_{\boldsymbol{\theta}}(a_t|\mathbf{s}_t)\right). \tag{3.21}$$

At time step t, $\hat{I}_t \in \{0, 1\}$ denotes whether the tth drug is selected. The idea is similar to Soft Actor-Critic (SAC). (32)

In this way, the surrogate loss at time step t is

$$L_t(\boldsymbol{\theta} = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\boldsymbol{\theta} - c_1 L_t^{VF}(\boldsymbol{\theta} + c_2 S\left(\pi_{\boldsymbol{\theta}}, \mathbf{s}_t\right)\right], \tag{3.22}$$

where c_1, c_2 are coefficients. With P parallel actors each collecting at most M-time step data, while the sampling procedure for each actor is the same as in 1, we can optimize the surrogate loss on these $P \times M$-time step data with minibatch SGD for K epochs; here K is called the ppo epochs. By optimizing with the proximal policy, we call the resulting ranking algorithm PPORank.

The sampling of one episode is equivalent to ranking the drug list for each cell-line under the current policy. The sampling process for one episode (ranking list) can be found in Algorithm 1.

The list-wise ranking process is constructed as follows: given a cell-line c, with feature vector X, the set of relevant drugs with feature set $D = \{\mathbf{d}_j\}_{j=1}^{M}$, and the true labels Y. At each time step t, the agent receives state $\mathbf{s}_t$, then according to the current policy $\pi_{\boldsymbol{\theta}}$, choose an action a_t of selecting drug $\mathbf{d}_{m(a_t)}$, and place it at rank position t. Then the environment moves to the next step $t + 1$, and transits to the next state $\mathbf{s}_{t+1}$; at the same time, the environment receives reward $r_t = r\left(\mathbf{s}_t, \mathbf{a}_t\right)$ as well as the evaluation signal to decide whether or not to update the hidden state. The process is continued until all the drugs are selected. And the algorithm can be found in Algorithm 2.

Algorithm 1 Sample An Episode

Input : $\boldsymbol{\theta}, \mathbf{X}, \mathbf{D}, \mathbf{Y}$.
Output : An Episode E.

Initialize $\mathbf{O}_0 \leftarrow [\mathbf{X}, \mathbf{D}, \mathbf{H}_0]$, $M \leftarrow |D|$, and episode $E \leftarrow \emptyset$
for t=0 to t=M-1 **do**
    Sample an action $a_{t+1} \in \mathcal{A}(\mathbf{s}_t) \sim \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$ by ((3.17))
    $r_t \leftarrow \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t)$ by ((3.6))
    Append $\mathbf{s}_t, a_t, r_t$ to the end of E
    Observe the evaluation signal from current ranking position
    if the feedback is positive **then** $H_{t+1} = GRU(\mathbf{X}, \mathbf{D}_t, \mathbf{H}_t)$
    else $\mathbf{H}_{t+1} = \mathbf{H}_t$
    end if
    Move to the next state $\mathbf{s}_{t+1} \leftarrow f([\mathbf{X}, \mathbf{D} \backslash \{\mathbf{d}_{m(a_t)}\}, \mathbf{H}_{t+1}])$
end for
return $E = (\mathbf{s}_0, a_0, r_1, \cdots, \mathbf{s}_{M-1}, a_{M-1}, r_M)$.

Algorithm 2 PPORank

Input : Labeled training dataset $\mathbf{T} = \left\{\left(c^{(i)}, \mathbf{X}^{(i)}, \mathbf{D}^{(i)}, \mathbf{Y}^{(i)}\right)\right\}_{i=1}^{N}$,
learning rate η, discount factor γ, ppo epoch K, GAE discount factor λ,
clip range ϵ, coefficients c_1, c_2, parallel actors P, mini-batch size m.
Output : model parameters $\boldsymbol{\theta}$.

Initialize the Actor $\pi_{\boldsymbol{\theta}}$ and Critic $V_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$.
Initialize $\boldsymbol{\theta}_{old} \leftarrow \boldsymbol{\theta}$
Initialize the rollout storage $\mathcal{B}$
for iteration=1,2... **do**
    for actors=1 to P **do**
        Run policy $\pi_{\boldsymbol{\theta}_{old}}$ and collect $E_k \leftarrow$ SampleEpisode $(\boldsymbol{\theta}_{old}, \mathbf{X}, \mathbf{D}, \mathbf{Y})$ {Algorithm 1}
        Compute advantage estimates $\hat{A}_1^{GAE} \cdots \hat{A}_M^{GAE}$ in Supplement eq ((3.20))
        Compute target value estimates $V_1^{\text{targ}} \cdots V_M^{\text{targ}}$
        Compute entropy $S(\pi_{\boldsymbol{\theta}}, \mathbf{s}_1) \cdots S(\pi_{\boldsymbol{\theta}}, \mathbf{s}_M)$
    end for
    Update rollout storage $\mathcal{B}$ with the $P \times M$ time steps data
    for epoch =1 to K **do**
        Optimize surrogate loss w.r.t. $\boldsymbol{\theta}$ with mini-batch size m using Adam in Supplement eq ((3.22))
    end for
    $\boldsymbol{\theta}_{old} \leftarrow \boldsymbol{\theta}$
end for
return $\boldsymbol{\theta}$.

3.5 Top k ranking

In drug recommendation, often only top one or few recommendations are needed (as opposed to predicting the exact values of the response to each of all the drugs). In drug recommendation, often only the top one or few matters (as opposed to predicting the exact value of a patient's responses to all drugs). Thus we may be more interested in ranking the top k drugs, instead of ranking all drugs,(82) though the latter may provide more information about the relationships between the drugs and cell lines. When using $NDCG@k$ as the evaluation metric for ranking, researchers have used different convex upper bounds for non-convex optimization with different k (89; 96); these upper bounds are functions of k. The optimization process requires pre-defined parameter k. We can adjust the tuning parameters from two aspects. (1) Cutting the length of each trajectory. Recall that in Algorithm 1, for each cell-line we run the ranking process to the end. Here in order to make more precise prediction on the top k drugs, we want to generate short trajectories for different actions given the initial state. (2) Modifying the discount parameter so that the agent will take into account only the most recent actions. Consider the Advantage function in (3.20), which is controlled by γ, λ. There are two specific cases with $\lambda = 0$ and $\lambda = 1$ as $\text{GAE}(\gamma, 0) : \hat{A}_t^{GAE} := \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ and $\text{GAE}(\gamma, 1) : \hat{A}_t^{GAE} := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t)$. When $\gamma = 0$, the agent acts to maximize the immediate reward. So for different choices of k, we choose different discounting factors.

In addition to the tuning process, we also include another supervised loss. As we are ranking the top k drugs, we can decompose this process into two parts: (1) selecting a list of k drugs; (2) re-ranking the k drugs. For selecting a list of k drugs , we have the entropy loss as

$$-(1 - \hat{\mathbf{a}}_{t,k}) \log(1 - \pi_\theta(a_t|\mathbf{s}_t)). \tag{3.23}$$

At time step t, $\hat{\mathbf{a}}_{t,k} \in \{0, 1\}$ denotes whether the selected drug is among the top k drugs based on the ground truth.

Chapter 4

Numerical Results in Ψ-learning Method

4.1 Simulation Studies

This section conducts simulations to investigate operating characteristics of the proposed method and compare with competing methods.

Our simulations concerns a variety of situations. Specifically, samples are generated: A covariate vector $X = (X_1, X_2, ..., X_{50})^T$ is sampled from an independent uniform distribution $U[-1, 1]$. Moreover, a binary treatment variable $A \in \{-1, 1\}$ is randomly sampled from an independent Bernoulli distribution with $P(A = 1) = \frac{1}{2}$. Then the response R is sampled from $N(Q_0, 1)$ with $Q_0 = T(\mathbf{X}) + T_0(\mathbf{X})A$, where $T(X)$ is a common effect from the clinical covariates and $T_0(\mathbf{X})A$ represents an interaction between the treatment and the clinical covariates.

1. $T(\mathbf{X}) = 1 + X_1 + X_2 + 2X_3 + 0.5X_4$ and $T_0(\mathbf{X}) = 1.8(0.3 - X_1 - X_2)$
2. $T(\mathbf{X}) = 1 + X_1 + X_2 + 2X_3 + 0.5X_4$ and $T_0(\mathbf{X}) = 0.4(X_2 - 0.25X_1^2 - 1)$;
3. $T(\mathbf{X}) = 1 + X_1^2 + X_2^2 + 2X_3^2 + 0.5X_4^2$ and $T_0(\mathbf{X}) = 3.8(0.8 - X_1^2 - X_2^2)$

Scenarios one and two are similar to those in (109), where the response function R is linear in the first four components for the main effect while the optimal treatment rule assigns treatment -1 for negative values of $T_0(\mathbf{X})$. In Scenario three, the response function R is

a quadratic function. Note that Scenarios two and three involve a nonlinear relationship in covariates whereas Scenario three misspecifies the model. Eight methods are compared, namely, (1) Linear-OWL (109), (2) Gaussian-OWL (109), (3) the L_1 penalized least squares L_1-PLS (54), (4) Linear-RWL (115), (5) Gaussian-RWL (115), (6) Linear-ψ-learning, (7) Gaussian-ψ-learning, (8) Linear-ψ-learning with variable selection (var-sel).

As a remark, we note that the OWL method can only treat non-negative outcomes (109). To make a fair comparison, we treat $R-\min(R)$ as an outcome for the OWL method. Moreover, the L_1-PLS approximates $E(R|\mathbf{X}, A)$ using a basis $(1, \mathbf{X}, A, \mathbf{X}A)$ followed by variable selection by Lasso (87) and estimates the optimal ITR by $\text{sign}(E(R|\mathbf{X}, A=1) - E(R|\mathbf{X}, A=-1))$.

With respect to tuning, the ψ-learning method has three or four tuning parameters. In particular, two tuning parameters (κ, τ) are estimated for Linear-ψ, three parameters (κ, σ, τ) are required for Gaussian-ψ, and four tuning parameters $(\kappa, \tau_1, \tau_2, \lambda)$ are for linear ψ with variable selection. To ease computation, for linear ψ-learning and Gaussian kernel ψ-learning, we set $\tau = \tau_1 = \tau_2 = 0.1$ and σ to be the median distance between the positive and negative classes (52), thus we only tune κ. For simplicity, we fix the tuning parameters of τ_1 and τ_2 as they don't impact our final result in Table 4.1. For linear ψ-learning with variable selection, we fix $\tau_2 = 1$ and $\tau_1 = 0.1$ to tune (λ, κ). In our simulations, we apply a 5-fold cross validation procedure, where we randomly partition the prognostic matrix $\boldsymbol{X}$, the reward $\boldsymbol{R}$ and the treatments $\boldsymbol{A}$ into training and tuning sets: $(\boldsymbol{X}^{tr}, \boldsymbol{A}^{tr}, \boldsymbol{R}^{tr})$ and $(\boldsymbol{X}^{tu}, \boldsymbol{A}^{tu}, \boldsymbol{R}^{tu})$, and on the tuning set we calculate the tuning error $TE(\mathbf{w}^{tr}) = \frac{1}{n^{tu}} \sum_{i=1}^{n^{tu}} \frac{I\left(1 \neq A_i^{tu} \text{sign}\left(f\left(\widetilde{\boldsymbol{X}}_i^{tu}\right)\right)\right)}{\pi(A_i, \mathbf{X}_i^{(tu)})} R_i^{tu}$.

For tuning (λ, κ) we minimize the tuning error $TE(\mathbf{w}^{tr}(\lambda, \kappa))$ with respect to (κ, λ), and we search over 20×20 grids points of (κ, λ) equally spaced on the log-scale over an rectangle of $[10^{-2}, 1) \times [10^{-4}, 10^{-2})$. In the presence of multiple minimizers,

we choose the one $(\hat{\lambda}, \hat{\kappa})$ giving the least number of model parameters to yield a more parsimonious model to conduct variable selection. For Linear-ψ and Gaussian-ψ, similarly, we minimize the tuning error $TE(\mathbf{w}^{(tr)}(\kappa))$ with respect to κ, where an evaluation of 50 equally spaced grids is used on the log-scale over $[10^{-4}, 1]$.

Two performance metrics are used to evaluate a method's performance. The first is the misclassification error rate, denoted by $\mathbf{MR} = \boldsymbol{P}_n^*[I(\hat{\mathcal{D}}(X) \neq \mathcal{D}^*(X))]$, measuring disagreement between two rules, where $\hat{\mathcal{D}}(X)$ and $\mathcal{D}^*$ are estimated ITR and the true optimal ITR, and $\boldsymbol{P}_n^*$ denotes the empirical average over a test dataset. The second evaluates the value function using the estimated optimal ITR on an independent test set of size 10000 based on an unbiased estimator of value function (**VF**) $\mathcal{V}(\mathcal{D})$:

$$\mathcal{V}(\hat{\mathcal{D}}) = \boldsymbol{P}_n^*[I(A = \mathcal{D}(X))R/\pi(A, \mathbf{X})]/\boldsymbol{P}_n^*[I(A = \mathcal{D}(X))/\pi(A, \mathbf{X})],$$

where $\boldsymbol{P}_n^*$ is the empirical distribution based on the test set and $\pi(A, \mathbf{X})$ is the probability of X being assigned to treatment A. Note that unbiasedness follows from the fact that $E[(R - \mathcal{V}(\mathcal{D}))I_{A=\mathcal{D}(\mathbf{X})}/\pi(A, \mathbf{X})] = 0$ for any ITR $\mathcal{D}$ (54).

For each scenario, we consider two sample sizes 100 and 400 for training based on 200 simulations. With respect to tuning the proposed ψ-learning methods, require two to four tuning parameters. For example, there are two tuning parameters (κ, τ) in ψ-linear, three tuning parameters (κ, σ, τ) for ψ-Gaussian, four tuning parameters $(\kappa, \tau_1, \tau_2, \lambda)$ for ψ-linear-VS. For ψ-linear and ψ-Gaussian, we fix the tuning parameter τ at 0.1, and the width of Gaussian kernel σ to be the median distance between the positive and negative classes from (52). As a result, we only need to tune κ in ψ-linear and ψ-Gaussian. For ψ-linear-VS we fix $\tau_2 = 1$ and $\tau_1 = 0.1$, thus we only need to tune over (λ, κ). In the simulation, we apply the five-fold cross-validation to tune parameters, and the training data were split to training set and tuning set. In the case of ψ-linear-VS, we perform a grid search over a predefined rectangle, specified as $[10^{-2}, 1) \times [10^{-4}, 10^{-2})$. When there are two tuning parameters, the tuning procedure can be summarized as:

1 Initialization :Supply an initial estimate $\mathbf{w}^{(0)}$ and a predefined finite set (λ, κ),

2 Partition: Randomly partition the prognostic matrix $\boldsymbol{X}$, the reward $\boldsymbol{R}$ and the treatments $\boldsymbol{A}$ into training and tuning sets: $(\boldsymbol{X}^{tr}, \boldsymbol{A}^{tr}, \boldsymbol{R}^{tr})$ and $(\boldsymbol{X}^{tu}, \boldsymbol{A}^{tu}, \boldsymbol{R}^{tu})$

Table 4.1: Mean misclassification error rates (MMR) and mean value functions (MVF) as well as standard errors (in parenthesis) over test sets for three scenarios based on 200 simulations. Best performers are bold-faced. Optimal values refer to those under the true optimal individualized treatment rule. Note that small values of MMR indicate good performance whereas those of MVF are just opposite.

	n=100		n=400	
	MMR	MVF	MMR	MVF
Scenario 1 (optimal value 2.25)				
L_1-PLS	**0.09(0.05)**	**2.20(0.08)**	**0.04(0.03)**	**2.23(0.02)**
Linear-OWL	0.36(0.05)	1.52(0.11)	0.32(0.03)	1.68(0.12
G-kernel OWL	0.38(0.04)	1.49(0.13)	0.33(0.04)	1.67(0.11)
Linear-RWL	0.28(0.05)	1.79(0.13)	0.14(0.02)	2.12(0.03)
Gaussian-RWL	0.28(0.04)	1.79(0.12)	0.14(0.02)	2.12(0.03)
Linear-ψ	0.22(0.06)	1.89(0.13)	0.14(0.02)	2.12(0.03)
Gaussian-ψ	0.25(0.04)	1.89(0.10)	0.12(0.02)	2.15(0.03)
Linear-ψ-var-sel	0.18(0.06)	2.02(0.12)	**0.04(0.04)**	**2.23(0.04)**
Scenario 2 (optimal value 1.43)				
L_1-PLS	0.11(0.07)	1.34(0.16)	0.03(0.07)	1.42(0.05)
Linear-OWL	0.14(0.12)	1.31(0.15)	0.03(0.03)	1.42(0.04)
G-kernel OWL	0.10(0.14)	1.35(0.13)	0.01(0.04)	1.42(0.05)
Linear-RWL	0.07(0.13)	1.38(0.11)	0.03(0.06)	1.41(0.03)
Gaussian-RWL	0.11(0.15)	1.34(0.13)	0.04(0.07)	1.41(0.05)
Linear-ψ	**0.04(0.08)**	**1.40(0.10)**	0.01(0.03)	1.42(0.03)
G-kernel ψ	**0.04(0.05)**	**1.40(0.04)**	**0.01(0.03)**	**1.43(0.02)**
Linear-ψ-var-sel	0.09(0.09)	1.35(0.12)	0.01(0.03)	1.42(0.04)
Scenario 3 (optimal value 3.88)				
L_1-PLS	0.40(0.07)	2.90(0.20)	0.38(0.07)	3.00(0.05)
Linear-OWL	0.39(0.05)	2.96(0.15)	0.38(0.03)	3.00(0.04)
G-kernel OWL	0.39(0.04)	2.97(0.12)	0.37(0.03)	3.03(0.04)
Linear-RWL	0.40(0.03)	2.90(0.19)	0.38(0.01)	3.00(0.05)
Gaussian-RWL	0.40(0.05)	2.89(0.19)	0.38(0.01)	3.00(0.09)
Linear-ψ	0.38(0.06)	2.95(0.20)	0.37(0.03)	3.03(0.03)
Gaussian-ψ	0.41(0.06)	2.89(0.20)	0.38(0.03)	3.00(0.10)
Linear-ψ-var-sel	**0.34(0.10)**	**3.07(0.26)**	**0.32(0.03)**	**3.17(0.14)**

3 Training: For each $\lambda \in \Lambda$ and each $\kappa \in R$, we apply Algorithm 1 to the training set with the initial estimate $\mathbf{w}^{(0)}$ to obtain the corresponding estimate $\mathbf{w}^{(tr)}(\lambda, \kappa)$.

4 Tuning: Given the tuning set $(\boldsymbol{X}^{tu}, \boldsymbol{A}^{tu}, \boldsymbol{R}^{tu})$, we minimize the tuning error

$$TE(\mathbf{w}^{tr}(\lambda, \kappa), (\boldsymbol{X}^{tu}, \boldsymbol{A}^{tu}, \boldsymbol{R}^{tu})) = \frac{1}{n^{tu}} \sum_{i=1}^{n^{tu}} \frac{I\left(1 \neq A_i^{tu} \text{sign}\left(f\left(\widetilde{\boldsymbol{X}}_i^{tu}\right)\right)\right)}{\pi(A_i, \mathbf{X}_i^{(tu)})} R_i^{tu},$$

where n^{tu} denotes the number of rows of $\boldsymbol{X}^{tu}$, that is the number of observations in the tuning set. Then

$$(\hat{\lambda}, \hat{\kappa}) = \underset{(\lambda,\kappa)\in(\Lambda,R)}{\arg\min}\ TE(\hat{\boldsymbol{\beta}}^{tu}(\lambda, \kappa), (\boldsymbol{X}^{tu}, \boldsymbol{A}^{tu}, \boldsymbol{R}^{tu}))$$

.

Given $\lambda = \hat{\lambda}$ and $\kappa = \hat{\kappa}$, we apply Algorithm Section 2.1.1 to the whole data $(\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{R})$ to find $\hat{\mathbf{w}}$ that minimizes $S(\boldsymbol{\beta})$ in ((2.5)).

For ψ-linear-VS, we minimize the tuning error $TE(\mathbf{w}^{tr}(\lambda, \kappa))$ with respect to (κ, λ), and we search over 20×20 grids points of (κ, λ) equally spaced on the log-scale over an rectangle of $[10^{-2}, 1) \times [10^{-4}, 10^{-2})$. Since the tuning parameters (κ, λ) are initially set as grids from rectangle, the value of an ITR corresponding to the different combinations of (κ, λ) may be the same, so in the cross validation process we always find the optimal value may have correspond to multiple selection of $(\hat{\lambda}, \hat{\kappa})$. We choose the $(\hat{\lambda}, \hat{\kappa})$ that has the corresponding least number of parameters to yield a more parsimonious model to conduct variable selection. For tuning of ψ-linear and ψ-Gaussian, the tuning process is similar. We minimize the tuning error $TE(\mathbf{w}^{(tu)}(\kappa))$ with respect to κ, where for a grid of 50 values of κ equally spaced on the log-scale over $[10^{-4}, 1]$.

As indicated in Table 4.1, the proposed weighted ψ-learning method substantially outperforms the penalized least squares and the OWL methods in Scenarios two and three, in terms of misclassification error and the mean value function, except in Scenario one, where its performance is quite close to them, especially when the sample size becomes large. In

each scenario, the amount of improvement of the proposed method over OWL ranges from 18% to 50% for classification and 3.7% to 20% for the value function. Yet when compared with RWL methods, we note that in Scenario three the results are comparable without variable selection. Moreover, variable selection improves the accuracy of estimation, particularly when the true model depends on only a small number of variables, as in Scenarios one and three.

4.2 Real Data Studies

We now analyze a dataset from a randomized intervention trial called the Lung Health Study (LHS) (23; 16). In the LHS, it was of interest to determine if a smoking intervention program with the possible prescription of an inhaled bronchodilator could reduce the annual rate of decline in FEV_1 (forced expiratory volume in one second) in smokers aged from 35 to 60 over a follow-up period of five years. Each participant of LHS was randomized into one of three intervention groups, usual care (**UC**) with no strong interventions (control), a smoking intervention program with the inhaled bronchodilator ipratropium bromide **(SIA)**, and a smoking intervention program with the inhaled placebo (**SIP**). The effect of the smoking intervention with or without inhaled bronchodilator can be assessed by differences of declining FEV_1-rates between the UC and SIA/SIP groups. Similarly, the effect of inhaled bronchodilator can be examined by comparing the difference of declining FEV_1-rates between the SIA and SIP groups.

Here the genotype attributes are used to identify novel genetic factors contributing to lung function. After excluding subjects with missing values, we had 1390, 1335 and 1361 participants in the UC, SIA and SIP groups, respectively.

In this analysis, we use the outcome as the decline of FEV_1 from the baseline to the final/fifth annual visit (4). To estimate the optimal ITR, we perform pairwise comparisons between all combinations of two intervention groups, and apply the eight competing methods to each two-arm comparison, that is, (1) Linear-OWL, (2) Gaussian-OWL, (3) the L_1 penalized least squares L_1-PLS, (4) Linear-RWL, (5) Gaussian-RWL, (6) Linear-ψ-learning,

(7) Gaussian-ψ-learning, (8) Linear-ψ-learning with variable selection (var-sel). Reward used here is the decline of post-BD FEV_1, so the smaller the better. The results from the ψ-methods are compared to the results from Linear-OWL, and L_1 partial least squares (L_1-PLS) (54), which used the basis function set $(1, \mathbf{X}, A, \mathbf{X}A)$ in the regression model. To evaluate the performance of the estimated rules, we use 5-fold cross-validation. The data is randomly partitioned into 5 equal-sized folds. We estimate the ITR based on four folds of the data with parameter tuning for predicted optimal treatments on the left-out fold. The procedure is repeated four more times with the other four combinations of the four folds for training and the other fold for tuning, finally obtaining the predicted optimal treatment for each patient. The first evaluation criterion was the value function, which is given by

$$\boldsymbol{P}_n^*[I(A = \mathcal{D}(X)R/\pi(A, \mathbf{X})]/\boldsymbol{P}_n^*[I(A = \mathcal{D}(X)/\pi(A, \mathbf{X})],$$

As used in (54), where $\boldsymbol{P}_n^*$ denotes the empirical average and $\pi(A, \mathbf{X})$ is the probability of being assigned treatment A, and a larger value function is preferred. The second criterion is based on significance testing. After predicting the optimal treatment for each patient, in each pair-wise comparison, we divide the patients into two groups: those who were actually assigned to the predicted optimal treatment groups and those who were not. Then we test the difference between the two groups using the two-sample t-test to see whether the group of the participants assigned to the estimated optimal treatment was better than the other group at the significance level $\alpha = 0.05$. Also we will provide the value functions for these two groups. The whole procedure was repeated 100 times with different partitions (for 5-fold cross-validation).

4.2.1 Treatment with only baseline characteristics

First, without the genotype data, we only consider using some baseline covariates. In the presence of an interaction between the treatment and the baseline covariates, the outcome

for FEV_1 is in a form of

$$R = T(\mathbf{X}) + T_0(\mathbf{X})A + \epsilon, \tag{4.1}$$

where R is the reversed cumulative decline of FEV_1, $\mathbf{X}$ represents the baseline covariates, $T(\mathbf{X})$ is for the main effects, and $T_0(\mathbf{X}) \cdot A$ is for the interaction, and ϵ is a mean-zero random error. The main effects here is linear, but the interaction can be either linear or nonlinear. The decision boundary is determined by the covariates $\mathbf{X}$. We first regress the outcome on the covariates, then use the residuals to estimate the parameters for the decision function. Here we estimate the main effects by minimizing the sum of weighted squares (115): $\sum_{i=1}^{n} \frac{1}{2\pi(A_i,\mathbf{X}_i)}(R_i - \beta_0 - \mathbf{X}_i^T\boldsymbol{\beta})^2$, where $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_p)^T$. Then the estimate of the main effects is $\widehat{T}(\mathbf{X}) = \widehat{\beta}_0 + \mathbf{X}^T\widehat{\boldsymbol{\beta}}$. And the interaction effect is estimated by $R - \widehat{T}(\mathbf{X})$.

We considered 16 possibly important baseline covariates based on previous research (10), including age, gender, year of education(YEAREDUC), body mass index(BMI), pack years (ACKYEAR), accepted screen2 Post-BD maximum forced vital capacity (FVCAC112), accepted screen2 Post-BD FEV_1 predicted (FEVPAC12), accepted screen3 Pre-BD maximum FEV_1 (FEVAC121), accepted screen3 Pre-BD maximum forced vital capacity FVC (FC121), accepted screen3 Pre-BD FEV_1 predicted (FEVPAC21), cigarettes per day (f31cigs). In the literature (86; 72), the methacholine reactivity was expressed as a logarithmic function of the two-point slope of percent decline in FEV_1 over the concentration of methacholine, so we also consider the methacholine related baseline covariates, including methacholine two-point dose-response slope from the baseline to the end of study (S3PAC2), (baseline) screen 3 methacholine absolute change in FEV_1 versus cumulative dose (S3SLOP2A), screen 3 methacholine absolute change in FEV_1 versus $log10(concentration + 1)$ (S3SLOP3A), screen 3 methacholine absolute change in FEV_1 versus concentration (S3SLOP4A).

We report the comparisons between SIA and SIP, SIA and UC, and SIP and UC respectively in Table 4.2 , where we list the value function based on the rewards as cumulative decline of post-BD FEV_1, so a smaller value is better; the standard errors of the value function are listed in parentheses. The Mean value for SIA (or SIP or UC) is the value function

for the participants whose estimated optimal treatment is SIA (or SIP or UC) as actually assigned. We also report the predicted treatments and the proportion of significant p-values in the table.

From Table 4.2, we can observe that in the comparison of SIA versus SIP groups, Linear-ψ method gives the smallest decline of FEV_1 , while Linear-OWL just estimates for every participant to receive SIA. By all four methods, the participants who actually received SIA as predicted by the optimal ITR have a smaller decline of FEV_1 than those in SIP group. This result is consistent with previous study that the aggressive intervention group can significantly reduce the decline of FEV_1 (4). In the comparisons of the two other group-pairs, SIA versus UC and SIP versus UC, we notice that in overall Linear-ψ does better than other methods. And the patients who have received the optimal treatment as the intervention (SIA or SIP) shows smaller decline of FEV_1 than those of UC group. The results confirm what was discovered in the previous study that the two smoking intervention groups showed significantly smaller declines in FEV_1 than the UC group (4). Interestingly, Linear-OWL assigns all the participants to one of the two intervention groups by its estimated optimal ITR, failing to detect possible heterogeneity among the participants. On the other hand, with only 16 covariates, it may be hard for Linear-ψ-var-sel to outperform Linear-ψ.

Moreover, across all three pairwise comparisons, the Linear-ψ-var-sel method will always select SEX, AGE, FEVPAC12, FEVAC121, YEAREDUC, FEVAC21, PACKYEAR, BMI, f31cigs and S3PAC2, though other different covariates may be selected.

4.2.2 Treatment with baseline covariates and genotype

In order to elucidate genetic variants associated with optimal interventions, we consider genotype data, in addition to the covariates considered above. Based on the literature search, we selected 38 genes possibly related to smoking in previous studies; each gene contains multiple SNPs. We firstly conduct a simple linear regression gene by gene to identify those more likely to be associated with the decline of FEV_1, leading to selecting two genes, DBH

Table 4.2: Mean Value functions(MVF) for the whole data and each subgroup with standard errors (in parenthesis) as well as the proportion of significant p-values for each pairwise comparison and predicted treatments for each subgroup over LHS phenotype data based on 100 repetition, where the reward is cumulative decline of FEV_1, of which small values indicates high performance. Best performers are bold-faced.

	Overall MVF	MVF of SIA	MVF of SIP	Prop p-value	Prediction SIA vs SIP
L_1-PLS	.163(0.04)	.162(0.04)	.165(0.04)	0.85	2171:174
Linear-OWL	.163(0.00)	.163(0.00)	.000(0.00)	0.95	2345:0
Gaussian-OWL	.163(0.00)	.163(0.00)	.000(0.00)	0.95	2345:0
Linear-RWL	**.153(0.05)**	.144(0.02)	.181(0.05)	0.95	1769:579
Gaussian-RWL	.156(0.07)	.150(0.02)	.162(0.06)	0.95	1625:720
Linear-ψ	**.153(0.03)**	.143(0.02)	.182(0.05)	0.90	1738:607
Gaussian-ψ	**.153(0.03)**	.143(0.02)	.182(0.05)	0.90	1738:607
Linear-ψ-var-sel	.155(0.04)	.153(0.04)	.170(0.04)	0.95	2069:276
	Overall MVF	MVF of SIA	MVF of UC	Prop p-value	Prediction SIA vs UC
L_1-PLS	.163(0.00)	.163(0.00)	.000(0.00)	1	2333:0
Linear-OWL	.163(0.00)	.163(0.00)	.000(0.00)	1	2333:0
Gaussian-OWL	.160(0.00)	.159(0.00)	.189(0.00)	0.95	2215:120
Linear-RWL	.160(0.06)	.159(0.05)	.190(0.05)	0.95	2207:126
Gaussian-RWL	.160(0.05)	.159(0.05)	.190(0.05)	0.95	2210:123
Linear-ψ	**.159(0.05)**	.157(0.05)	.193(0.06)	1	2200:133
Gaussian-ψ	.160(0.03)	.159(0.02)	.192(0.05)	0.95	2221:112
Linear-ψ-var-sel	**.159(0.04)**	.158(0.04)	.192(0.06)	1	2174:159
	Overall MVF	MVF of SIP	MVF of UC	Prop p-value	Prediction SIP vs UC
L_1-PLS	.193(0.02)	.194(0.02)	.198(0.02)	0.90	2303:9
Linear-OWL	.194(0.00)	.194(0.00)	.000(0.00)	0.97	2312:0
Gaussian-OWL	.193(0.01)	.193(0.01)	.200(0.00)	0.95	2300:12
Linear-RWL	**.190(0.02)**	.188(0.03)	.205(0.02)	1	1995:308
Gaussian-RWL	.193(0.00)	.191(0.00)	.203(0.00)	0.95	2201:99
Linear-ψ	.191(0.02)	.190(0.02)	.203(0.02)	1	2022:290
Gaussian-ψ	.191(0.03)	.190(0.02)	.204(0.05)	0.95	2112:188
Linear-ψ-var-sel	.192(0.01)	.192(0.01)	.194(0.02)	0.87	2199:113

and SGCD, including 15 and 67 SNPs respectively. Combining these 82 SNPs with the 16 baseline covariates, we have a total 98 covariates/predictors. We estimate the optimal ITRs and compare the groups in a similar procedure as in the previous subsection.

In the presence of an interaction between the treatment and genotype as well as the baseline covariates, the outcome for FEV_1 is in a form of

$$R = T(\mathbf{X}, \boldsymbol{Z}) + T_0(\mathbf{X}, \boldsymbol{Z})A + \epsilon, \tag{4.2}$$

where $T(\mathbf{X}, \boldsymbol{Z})$ is the main effect, $T_0(\mathbf{X}, \boldsymbol{Z}) \cdot A$ is the interaction, and ϵ is a mean 0 random error. We model the main effects as linear, but the interaction can be linear or nonlinear. The decision boundary is determined by both the genotypes $\boldsymbol{Z}$ and the baseline covariates $\mathbf{X}$. As done before, we regress the outcome on the baseline covariates and genotype data, then use the residuals to estimate the parameters for the decision function. The estimated decline of FEV_1 is calculated as before using 5-fold cross-validation with the same two criteria to compare the performance of the four competing methods. Similarly, we report the pairwise comparisons between any two groups in Table 4.3.

In the comparison of SIA versus SIP groups in Table 4.3, both Linear-ψ and Linear-ψ-var-sel outperform the other methods; in particular, with relatively high-dimensional genotype data, Linear-ψ-var-sel, by conducting variable selection, would avoid the over-fitting problem, and helps to find out important genetic variants. And compared with the results from the comparison in SIA verse SIP groups in Table 4.2, we can find an improvement in preventing the decline of FEV_1, which indicates that there is some effect of important genetic variants related with the decline of FEV_1. Although L_1-PLS also can conduct variable selection, for real data it is hard to decide the right basis functions; in this sense, ψ-method is more robust than L_1-PLS. For Linear-OWL, with increasing dimensions, it cannot select important variables; it tends to assign the participants to the treatment that the patients really receives. In the comparisons of SIA versus UC and SIP verse UC groups in Table 4.3, Linear-ψ and

Table 4.3: Mean Value functions(MVF) for the whole data and each subgroup with standard errors(in parenthesis) as well as the proportion of significant p-values for each pairwise comparison and predicted treatments for each subgroup over LHS phenotype and genotype data based on 100 repetition, where the reward is cumulative decline of FEV_1, of which small values indicate high performance. Best performers are bold-faced.

	Overall MVF	MVF of SIA	MVF of SIP	Prop p-value	Prediction SIA vs SIP
L_1-PLS	.170(0.03)	.171(0.03)	.166(0.02)	0.82	2175:170
Linear-OWL	.151(0.02)	.134(0.02)	.170(0.02)	0.97	1279:1066
Gaussian-OWL	.150(0.02)	.136(0.02)	.169(0.02)	0.97	1699:646
Linear-RWL	.143(0.03)	.140(0.02)	.155(0.03)	0.90	2099:246
Gaussian-RWL	.139(0.03)	.138(0.02)	.150(0.03)	0.90	1821:512
Linear-ψ	.142(0.03)	.140(0.02)	.152(0.03)	0.90	2025:320
Gaussian-ψ	.140(0.03)	.139(0.02)	.151(0.03)	0.90	1900:445
Linear-ψ-var-sel	**.136(0.02)**	.131(0.02)	.144(0.04)	0.89	1420:925
	Overall MVF	MVF of SIA	MVF of UC	Prop p-value	Prediction SIA vs UC
L_1-PLS	.177(0.03)	.177(0.03)	.184(0.04)	0.75	2214:19
Linear-OWL	.177(0.01)	.176(0.01)	.198(0.01)	1	2290:43
Gaussian-OWL	.174(0.02)	.170(0.02)	.191(0.02)	1	2200:145
Linear-RWL	**.159(0.02)**	.158(0.02)	.166(0.03)	0.85	2073:282
Gaussian-RWL	.160(0.04)	.159(0.03)	.169(0.04)	0.85	2139:206
Linear-ψ	**.159(0.02)**	.158(0.02)	.166(0.02)	0.85	2047:286
Gaussian-ψ	**.159(0.03)**	.158(0.03)	.166(0.02)	0.85	2089:256
Linear-ψ-var-sel	**.159(0.02)**	.158(0.02)	.164(0.02)	0.85	2114:219
	Overall MVF	MVF of SIP	MVF of UC	Prop p-value	Prediction SIP vs UC
L_1-PLS	.199(0.01)	.199(0.01)	.200(0.01)	0.81	2228:84
Linear-OWL	.191(0.02)	.176(0.01)	.219(0.02)	1	1858:454
Gaussian-OWL	.185(0.04)	.173(0.03)	.201(0.03)	0.95	1858:454
Linear-RWL	.166(0.02)	.152(0.02)	.193(0.02)	1	1655:657
Gaussian-RWL	.166(0.03)	.151(0.04)	.200(0.03)	0.95	1601:711
Linear-ψ	.166(0.02)	.152(0.02)	.193(0.02)	1	1608:704
Gaussian-ψ	.166(0.03)	.152(0.04)	.193(0.03)	0.95	1700:612
Linear-ψ-var-sel	**.164(0.02)**	.162(0.02)	.201(0.02)	1	2169:143

Linear-ψ-var-sel perform similarly, both outperforming the other methods. Also L_1-PLS fails to detect the difference between the SIA and UC groups when the participants actually received the estimated optimal treatments. And when compared the MVF of the intervention groups (SIA and SIP) with that of UC group, ψ-methods will recommend the intervention treatment with significantly lower decline of FEV_1. Especially when comparing the SIP verse UC groups, by including the genetic variants, the decline of FEV_1 has improved from 0.192 to 0.165 by Linear-ψ-var-sel, which indicates the information in genotype to help with FEV_1.

Among all the three pairwise comparisons with the genotype data, ψ-methods outperform others; in particular, perhaps Linear-OWL fails to select important variables while L_1-PLS misses right basis functions. Across the three comparisons,Linear-ψ-var-sel always selects the following covariates: SEX, AGE, FEVAC112, FEVPAC12, FVCAC121, FEVPAC21, PACKYEAR, S3PAC2 and S3OCONNR, suggesting their importance; in addition, it also selects some important genetic variants.

In summary, the proposed weighted ψ-learning method can identify potentially useful ITRs with the findings consistent with the previous studies (4). Furthermore, with high-dimensional data such as when genotype data are included,Linear-ψ-var-sel seems to perform better with its variable selection capability.

Chapter 5

Numerical Studies in PPORank

We first conducted our experiments on the cancer screening data sets of GDSC and CCLE. Then we demonstrated the effectiveness of PPORank using simulation data.

5.1 Cancer Drug Screening Data

5.1.1 Datasets

The drug-screening data for cancer cell lines are obtained from Genomics of Drug Sensitivity in Cancer (GDSC) and Cancer Cell Line Encyclopedia (CCLE). The GDSC database provides a large-scale drug screening dataset with each $\log(\mathrm{IC}_{50})$ value for a drug-cancer cell line pair. The CCLE database provides genomic, transcriptomic, and epigenomic profiles for more than a thousand cancer cell lines. The GDSC has released a panel of 1001 cancer cell lines (covering 31 cancer types) as well as 265 pharmacological compounds/drugs. (37) The cell lines were mostly characterized using gene expression, whole-exome sequencing, copy number variation, and DNA methylation. As pointed by, (19) gene expression data is most informative. Then, we will put more emphasis on our analysis based on gene expression data. To this end, we encode each of the 983 profiled cell lines with the RMA-normalized (robust multi-array average) basal expression of 17737 genes. To extract cell line features from the gene expression profiles, we adopt the method of , (82) where we normalize the baseline gene expression values for each gene by computing the fold-changes compared to

the median value across cell lines. According to , (91) 1856 essential genes are selected for dimension reduction for each cell line. Then we calculate Pearson's correlation for every pair of cell lines using the expression fold-changes of these essential genes, among which we select 983 cell line features for the full GDSC dataset.

For the CCLE dataset, 24 drugs and a panel of 1036 human cancer cell lines are available, and we select 491 cancer cell lines for which both drug sensitivity measures and gene expression profile data are available.

Both GDSC and CCLE report drug sensitivity as the logarithm of half maximal inhibitory concentration of a drug on a cell line, $\log(\mathrm{IC}_{50})$, which denotes the concentration of the drug/compound required to inhibit the cell growth at 50%; the smaller $\log(\mathrm{IC}_{50})$, the more sensitive the cell line to the drug. But the measured IC_{50} values are not comparable across different compounds. (37) calculated drug-specific sensitivity (binarization) thresholds for each of the 265 tested drugs. These thresholds were determined using a heuristic outlier detection procedure (44) following a previous observation that the majority of cell lines are typically resistant to a given drug. (26) Thus, we use these drug-specific sensitivity thresholds to normalize the data set. Furthermore, as the ranking metric NDCG is only bounded with non-negative scores, we normalize the sensitivity scores as $-\log(\mathrm{IC}_{50}/thr_d) + a$, where thr_d is the sensitivity threshold of the given drug and a is the maximum normalized $\log(IC_{50})$ score across all drugs. After deleting toxic drugs, we have 223 drugs for GDSC and 19 drugs for CCLE.

For drug features, chemical structural information for each drug is available in PubChem. (41) We exclude drug samples without PubChem ID in the GDSC database, and we also exclude 15 drugs with the same PubChem IDs, and finally 223 drugs remained. But at the end, for fair comparisons with other baseline methods, drug features are not used.

We employ The Cancer Genome Atlas (TCGA) breast cancer (BRCA) sub-cohort data to further evaluate whether/how the cell line-based GDSC results of PPORank generalize to cancer patients. We download the Firehose data run 2016_01_28 from. (34) We use the immunohistochemistry annotations to identify HER2 over-expressed ($\mathrm{HER2}^{+}$) patients

($n = 163$) and triple-negative breast cancers (TNBCs) patients ($n = 116$). Furthermore, we define the BRCA1/2-mutant (mBRCA) patients as the 37 patients carrying germline loss-of-function BRCA1/2 mutations. (57) Finally, we use the pipeline proposed by (27) to harmonize the Level 3 Illumina HiSeq RNA-seq v2 data (1080 patients) and the GDSC gene expression data.

5.2 Baseline methods and evaluations

Our method (PPORank) is motivated by more efficient healthcare applications: (i) in clinical settings, it is more important to recommend the most sensitive drugs in ranked order, instead of numerically predicting the sensitivity score of a given drug; (ii) a treatment regime is usually characterized after a prolonged and sequential procedure; (iii) currently, many cancer-related resources covering genotypes, phenotypes, and drug information are available, calling for integrating theses multiple data sources for model building; (iv) often only one or few *top* recommendations are needed (as opposed to all the drugs).

To address the above issues, we will conduct experiments from the following aspects:

(1) for completeness, we evaluate our method on the full GDSC and CCLE data sets; for a fair comparison, we use the same data sources including only cell line features and drug response matrix and evaluate with NDCG to show long-term effects, addressing the above issues (i) and (ii);

(2) to integrate multiple data resources, we also include drug similarity data into the training dataset along with other molecular data: whole-exome sequencing (WES), copy number variation (CNV), and DNA methylation (MET) data, without any changes to the model structure, addressing issue (iii);

(3) to address the top k ranking problem of issue (iv), we also evaluate the methods with the NDCG@k ranking metrics for different values of k.

We compare the methods with 5-fold cross-validation, where each fold of the data is used once for testing while the other four folds are used for training. For each fold, we test

completely on unseen cell-lines. We compare our method, PPORank, against several state-of-the-art and representative ones: a method based on the elastic net regression model ElasticNet (EN), (7; 37) kernel ridge regression (KRR), (63) similarity-regularized matrix factorization(SRMF), (91)) Cancer Drug Response prediction using a Recommender System (CaDRRes), (82)) and Kernelized Ranking Learning(KRL). (33) For EN, the model is trained for each drug as described previously (7; 37) using the Elastic Net library in scikit-learn (l_1 ratio =0.5). KRR has been recommended as one of the best-performing algorithms in a systematic survey (38) and was also implemented using the machine learning library- scikit-learn in Python. For SRMF, it goes beyond matrix factorization by also considering drug similarity obtained from chemical structural data and cell line similarity based on their gene expression profiles as regularization terms to avoid over-fitting; but as described in the paper, drug similarity did not improve the prediction performance, so we also ignore drug similarity here. Furthermore, SRMF cannot make predictions on unseen cell lines or patients. For CaDRReS, we set the latent space dimension to be 10. For KRL, we adopt the Radial Basis Function (RBF) kernel, which performs better than the linear kernel, and instead of using cell line similarity features, according to the original paper, we use the RMA-normalized (robust multi-array average) basal expression of 17737 genes as the input cell line features.

5.3 Hyper-parameter tunning

For the baseline methods, the tunning parameters are tuned over grids with 5-fold cross-validation. For PPORank, we have the following hyper-parameters, some of which are fixed while others are tuned through the same cross-validation procedure as for the baseline methods. The cross-part of the deep- and cross network has two cross layers; the deep part of the Deep & Cross network has three layers with 128, 64, and 32 nodes respectively. For the Deep & Cross network, we tried some deeper or shallower structures but found that with more layers, the performance did not improve much while a higher dropout rate was needed; to reduce the computational complexity, we ended with the above network structure for our agent policy. For the PPO part, there are two main issues: how many experience

epidodes (T) should the agent gather before updating the policy and how to update an old policy to a new policy. For each trajectory, the cell lines have at most 24 and 265 drugs for the two datasets respectively, so T is set to be 24 for CCLE and 265 for GDSC; for a fair comparison, we do not use the drug fingerprint features here; the numbers of the paralleled actors are 8 for CCLE and 16 for GDSC, mini-batch size is 16 for both, PPO epoch number K for the surrogate loss (in the Supplementary eq ((3.22))) is 8 for both. For other hyper-parameters in the clipping loss (in the Supplementary eq ((3.18)), which controls the new policy from moving into collapse), the clipping range ϵ is tunned among $\{0.1, 0.2, 0.3\}$, and the discounting factors λ, γ used in the advantage function (in the Supplementary eq ((3.20))) are tuned in grids of $\{0.1, 0.2, 0.3, \cdots, 0.9, 0.99\}$, but when evaluated on the whole trajectory's ranking, they are all fixed at the value of 0.95; we only need to tune their values when the evaluation metrics are NDCG@k. The coefficients with the value function and entropy(in the Supplementary eq ((3.22))) are fixed at $c_1 = 0.5$ and $c_2 = 0.001$.

5.4 Prediction with the full dataset

We use the full GDSC and CCLE data sets with only cell line features to compare the performance of different methods, then we report the average performance based on NDCG through five-fold cross-validation. The results are shown in Figure Figure 5.3. For the CCLE dataset, we can see that our method PPORank performs marginally better than other methods: it gives a mean NDCG of 0.7611, slightly higher than 0.7432 and 0.7468 from linear methods CaDRRes and EN. Note that the total number of the drugs is only 24, so even with unseen cell lines, the candidate drugs have appeared frequently enough in the training set, leading to good performance by both regression methods and static ranking methods. However, for the GDSC dataset, as shown in Figure Figure 5.3(b), the improvement of our PPORank over other methods is much higher. PPORank shows a consistent improvement over all the other methods with the smallest standard deviation (SD) (0.003); as a comparison, the second-best method, CaDRRes, has a much larger SD of 0.011, highlighting the robustness of our proposed method. Note that CaDRRes also uses

the NDCG as a loss function, which may explain why it performs second best. To further compare the performance, we will use NDCG@k to show the top $k \geq 1$ selected drugs by each method as to be discussed in Section 3.3 (where it is shown that PPORank demonstrates larger performance gains over CaDRRes and other methods).

To compare the performance between our proposed RL and DNN-based supervised learning, we also developed and trained a DNN using an approximate NDCG as the loss function (69) based on the GDSC data. Figure Figure 5.4 shows that after 600 epochs, PPORank approaches the performance of the DNN. It suggests that the improvement of PPORank over other methods perhaps mainly comes from two sources: its highly non-linear model and its direct use of NDCG as its target loss function.

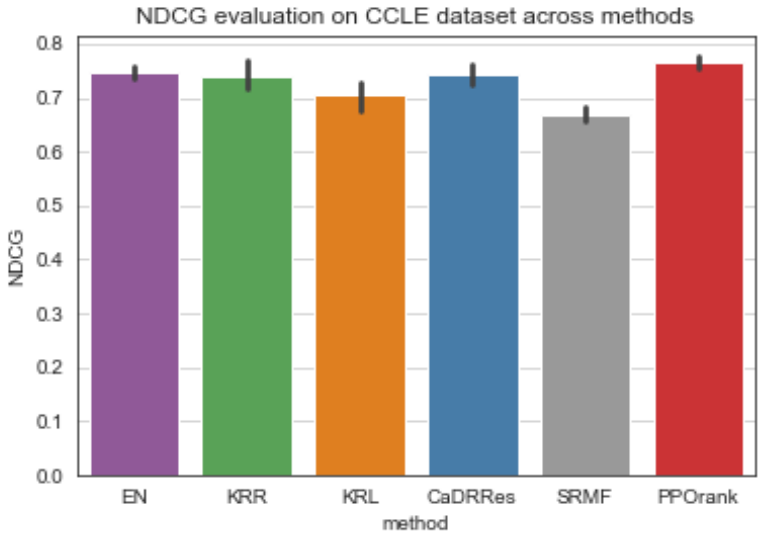

Figure 5.1: NDCG values using full CCLE data set.

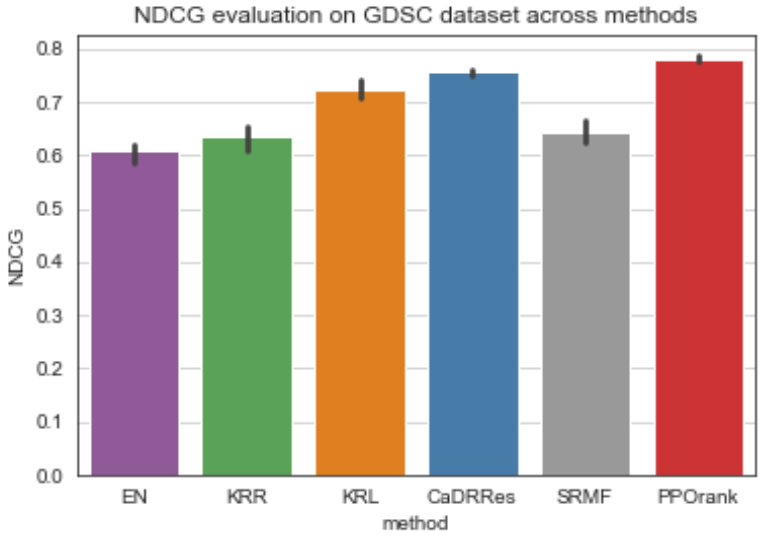

Figure 5.2: NDCG values using full GDSC data set.

Figure 5.3: Mean NDCG values by 5-fold cross-validation (with 1 SD as error bars).

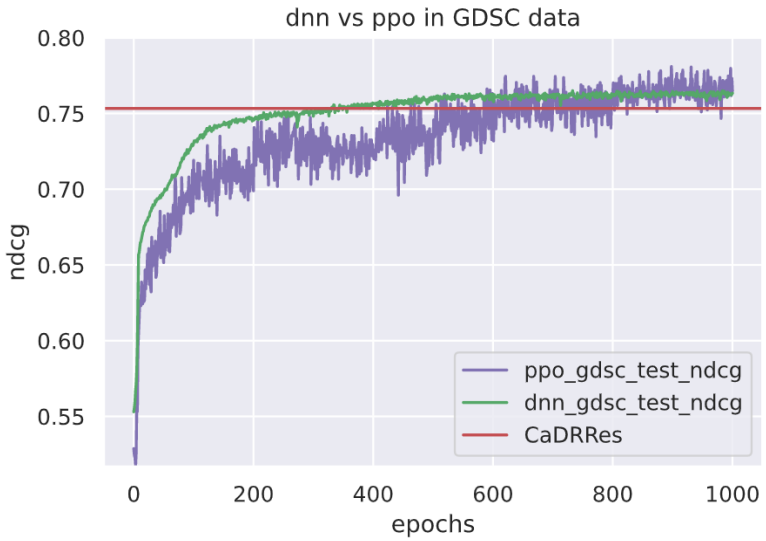

Figure 5.4: DNN and PPORank 's performance using $NDCG$ on the GDSC data.

5.5 Prediction with other types of cell line features

We use the Gaussian kernel (14) to summarize side information from other omic data sources: for each cell line c, $\mathbf{K}(\mathbf{x_c}, \mathbf{x_{c'}}) = \exp\left(-\|\mathbf{x_c} - \mathbf{x_{c'}}\|^2 / 2\sigma^2\right)$, where $\mathbf{x_c}$ and $\mathbf{x_{c'}}$ denote the feature vectors of two cell lines in the form of (i) whole-exome sequencing (WES), (ii) copy number variation (CNV), or (iii) DNA methylation (MET) data, and σ is the kernel width of the Gaussian kernel. For simplicity, we heuristically determine the kernel width using the 'median trick', setting the width as the reciprocal of the median squared Euclidean distance among all training sample pairs. In this fashion, we generate an $N \times N$ kernel matrix $\mathbf{K}$ for each additional data type, where N is the number of cell lines. We treat each row of $\mathbf{K}$ as a feature vector of length N for the corresponding omic data and cell-line, which can be combined with the original cell-line features to incorporate the use of other omic data.

Since our method is based on DNNs, it is quite flexible to accommodate other data sources: we add an (MLP) subnetwork to project other high-dimensional features to a low-dimension space before feeding its output to the stacking layer. As gene expression (GEX) data is known to be informative, we use the following cell line features (1) GEX+WES,

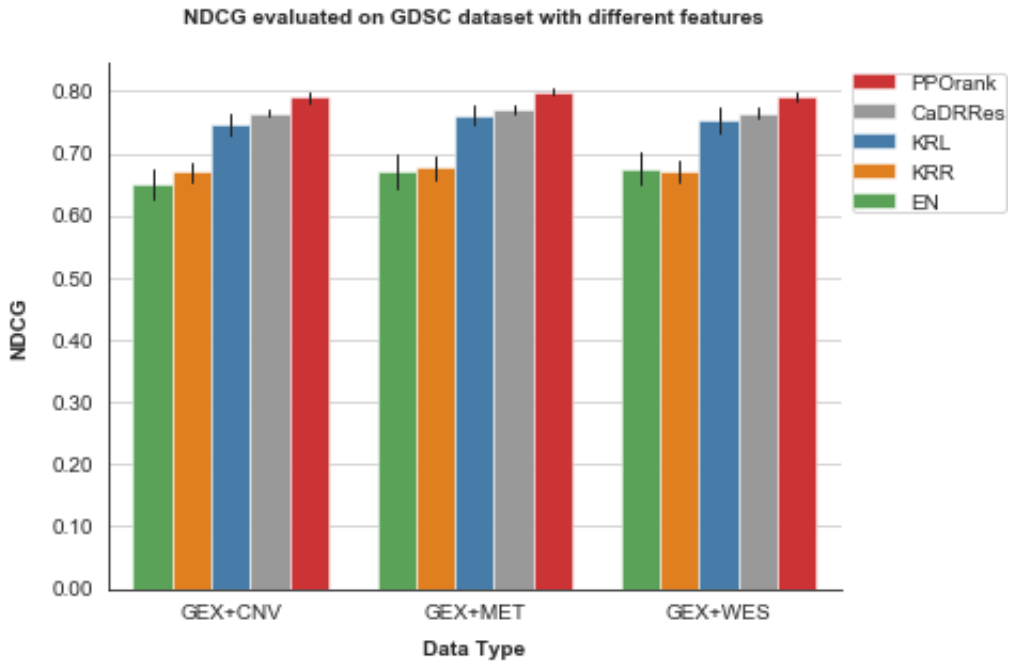

Figure 5.5: Mean NDCG values by 5-fold cross-validation on the GDSC data with different types of omic data features.

(2) GEX+CNV, (3) GEX+MET respectively to show how performance may change; more details about these features can be found in Appendix Table B.1. Figure Figure 5.5 compares the performance in NDCG across using these three additional types of features. In agreement with the findings from a drug sensitivity collaborative competition, (19) gene expression is confirmed to be the most predictive data type regardless of the method being employed, since it shows almost no improvement with the additional CNV or WES data and only slight improvement with the MET data; these results are also consistent with some previous studies supporting that MET data are informative to cancer. (43)

5.6 Sequential Learning and Prediction

One distinct feature of RL is its sequential learning. Although all the data from GDSC are available at the time of training, we mimic a realistic situation by starting RL with a subset of the available data. We begin with 50% of all the cell lines in the GDSC data at start time t_0. As the model uses all the drugs for embedding learning in our deep & cross net,

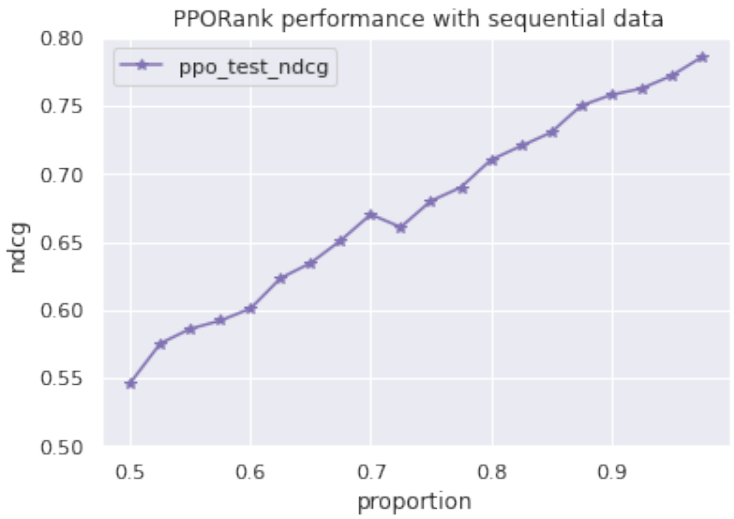

Figure 5.6: PPORank performance on GDSC with sequential data

we consider all the drugs at time t_0; we evaluate the method at each following time point using the independent test data. Specifically, at time point $t_0 + 1$, we add 20 cell lines as new data and run the RL algorithm based on the feedback of the newly added data; after the convergence, we evaluate its performance based on the (unseen) test data. Then we add another 20 cell lines as new data, and repeat the process. To save time, at each time point we start RL with the estimated model (i.e. its estimated parameters) obtained from the previous run (or time point), then update the model with the new incoming cell lines. Figure Figure 5.6 displays the results of applying RL on the GDSC test data, starting at t_0 with only 50% of the whole training data and subsequently adding 20 more cell lines at each time point until we use up all the GDSC training data. As expected, as more data are added, overall the performance of RL improves. It is confirmed that RL performs well in the realistic set-up for sequential learning.

5.7 Prediction with Top k Ranking

To show the ability of the PPORank model for ranking the top k most sensitive drugs across cell lines, we use the full CCLE and GDSC data sets respectively to compared the methods

in terms of NDCG@k with $k \in \{1, 5, 10\}$; with $k = 1$, one is to select the most sensitive drug. As shown in Figure Figure 5.9, PPORank performs best consistently; its improvements over other methods are more obvious for the GDSC data. From Section Section 3.2,

$$\text{NDCG@}\{k+1\} - \text{NDCG@}k = \frac{[\frac{2^{f_{\pi^{-1}(k+1)}}-1}{\log_2(k+2)} \times Z_k - DCG@k \times \frac{2^{y_{k+1}}-1}{\log_{k+2}}]}{Z_k Z_{k+1}},$$

where Z_k is a normalized factor that $Z_k \geq \text{DCG@}k$. So, when $\frac{2^{f_{\pi^{-1}(k+1)}}-1}{\log_2(k+2)} \times Z_k$ is larger than $\text{DCG@}k \times \frac{2^{y_{k+1}}-1}{\log_{k+2}}$, NDCG@$k$ increases in k. In practice, depending on data, an increasing trend may not be seen in all circumstances for a top k ranking problem.

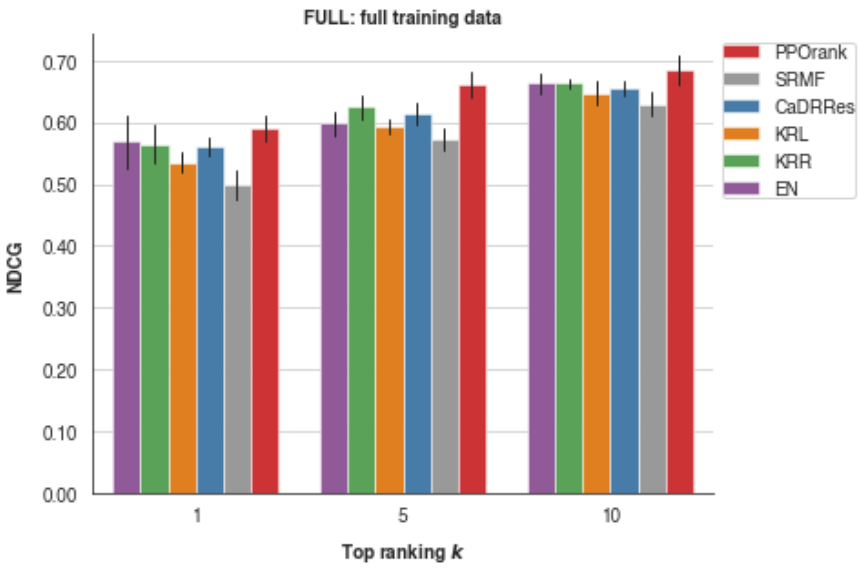

Figure 5.7: NDCG@k with the CCLE data.

Figure 5.8: NDCG@k with the GDSC data.

Figure 5.9: NDCG@k values with the full CCLE or GDSC data for different k; the error bars show one SD based on 3-fold cross-validation.

*

5.8 External Validation with TCGA Breast Cancer Cohort

To show the generalization ability of PPORank, we show that drug ranks predicted with PPORank trained on cell lines from GDSC are correlate with patient response to treatment in a TCGA breast cancer cohort. (66) We follow the data preprocessing of (27) to harmonize the gene expression profiles measured by RNA-seq in TCGA and by micro-arrays in GDSC. In this way, they were able to train ridge regression models on the GDSC cell lines and use these models to 'impute' drug response for the TCGA patients. They also showed that $HER2^+$ TCGA-BRCA patients were predicted to be more sensitive to lapatinib, a first-line therapy for $HER2^+$ patients, (28) than other patients. Here we follow the idea of using molecular subtypes to evaluate PPORank's recommendations.

We compare lapatinib and four PARP 1/2 inhibitors (PARPi), veliparib, olaparib, talazoparib, and rucaparib, some of which have shown promising therapeutic response in BRCA1/2-mutant (mBRCA) tumors. (90) In our experiments, lapatinib is ranked higher than all four PARPi for 150 (92%) of the 163 $HER2^+$ patients; in contrast, for mBRCA TNBC , lapatinib is ranked higher than PARPi only 22%. Table 5.1 shows how lapatinib is recommended as compared to the overall and individual PARPi treatments. To show the robustness of our proposed method, we retrain the model 10 times with a subset of 100 drugs (but with lapatinib and PARPi always included) randomly selected from the 200+ available drugs in the GDSC data. It is confirmed that PPORank's recommendations are indeed statistically different from that of a random recommendation (the unpaired t-test with mean p-value $= 10^{-11}$ for $HER2^+$ and p-value $= 10^{-27}$ for mBRCA TNBC patients). Furthermore, when we apply the unpaired t-test to compare random recommendations with each of the four individual PARPi treatments, the results were statistically significant (p-value ≤ 0.005) except for rucaparib (p-value $= 0.13$). This suggests no recommendation of rucaparib to mBRCA TNBC patients, which is in agreement with a recent clinical trial's (22) conclusion that the efficacy of rucaparib was not established and it re-

Table 5.1: PPORank's recommendation rates of lapatinib for the TCGA patients with two different breast cancer subtypes.

Recommendation	HER2$^+$ ($n = 163$)	mBRCA ($n = 9$)
lapatinib > PARPi	0.92	0.22
lapatinib > veliparib	0.95	0.22
lapatinib > olaparib	0.94	0.44
lapatinib > talazoparib	0.92	0.22
lapatinib > rucaparib	0.95	0.44

quired further investigation.

5.9 Simulation Studies

5.9.1 Primary simulations

To compare our method with other baseline methods, we first construct the response matrix with information from the cell lines. Now consider 100 drugs and 1000 cell lines with 2000 features for cell lines thus $\mathbf{X} \in \mathbb{R}^{1000\times 2000}$. As pointed out by, (91; 76) drug pairs within the same cluster of chemical fingerprints can show similar inhibitory effects on the same cell line. So we generate $\mathbf{X}$ with 2000 features from 10 clusters with a cluster size of $\{480, 380, 300, 240, 190, 150, 120, 90, 40, 10\}$; the features belong to the same cluster are correlated. Each feature is generated from a uniform distribution. The correlation matrix of $\mathbf{X}$ can be seen from Figure Figure 5.10. We design a weight matrix $\mathbf{W} \in \mathbb{R}^{2000\times 100}$ with 10 clusters each with an equal size of 10; for each cluster, the weight is generated from a standard normal distribution as shown in Figure Figure 5.11. For each simulated cell line, a maximum number of 100 drugs need to be ranked.

We generate the response matrix $\mathbf{Y} \in \mathbb{R}^{1000\times 100}$ from one of the three models:

1. $\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\epsilon} = 0.2 \times \mathbf{XW} + \boldsymbol{\epsilon}$,

2. $\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\epsilon} = 0.15 \times \mathbf{X}^3\mathbf{W} + 0.15 \times \mathbf{XW} + \boldsymbol{\epsilon}$,

3. $\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\epsilon} = 0.1 \times \exp(\mathbf{X})\mathbf{W} + 0.1 \times \mathbf{X}^3\mathbf{W} + \boldsymbol{\epsilon}$,

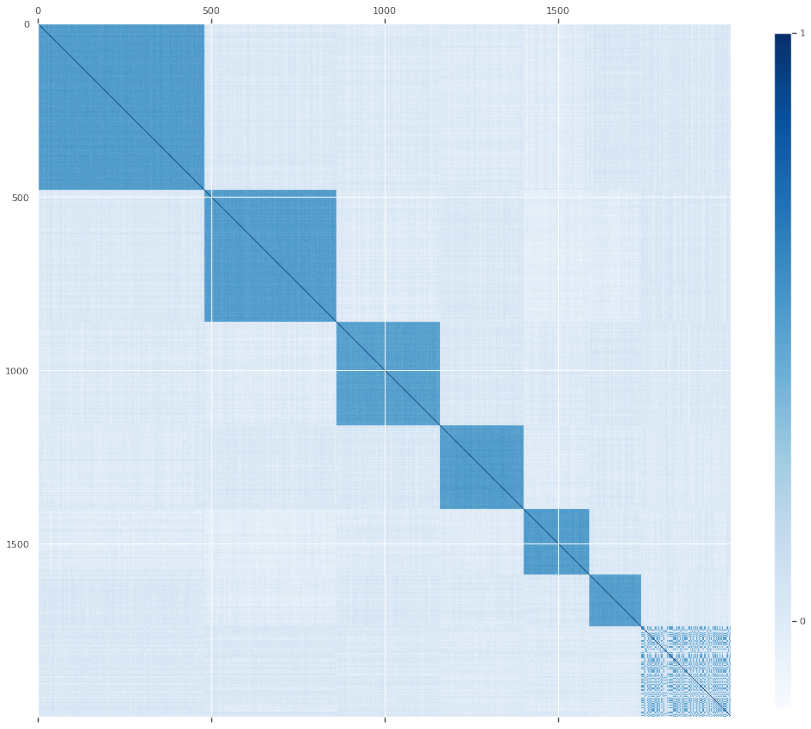

Figure 5.10

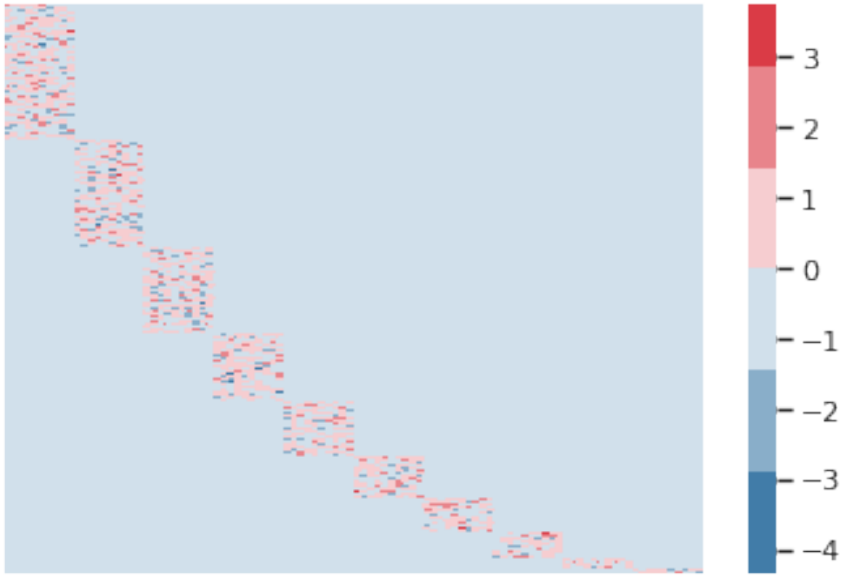

Figure 5.11

Figure 5.12: Simulation setups: (a) the correlation matrix of the simulated cell lines features with 10 clusters; (b) the weight matrix **W** with 10 clusters.

representing linear, non-linear (cubic) and highly non-linear (exponential) scenarios, and $\boldsymbol{\epsilon}$ is a Gaussian noise. Here, instead of using the original Pearson correlation matrix as in the CaDRRes paper, (82) we use the original cell line features from $\mathbf{X}$. We define the (i, j)-th element of $\mathbf{Y}_{true}$ as the true mean response for the i-th cell line given the j-th drug. As NDCG is commutable only when the relevance label is non-negative, we set a response value $\mathbf{Y}$ as missing if it is negative. We also set 50% of the remaining response values as missing. To distinguish between sensitive and nonsensitive drugs, we use the median response of all the cell lines to a specific drug as the threshold. If a response value is larger than the threshold, we take it as sensitive/positive; otherwise, it is nonsensitive/negative. During training, if we have a nonsensitive drug in a ranking position while there are other sensitive candidate drugs to be selected, we take it as a negative evaluation signal and otherwise as a positive evaluation signal.

We also compare DNN with PPORank in the linear and exponential scenarios with a sample size of 1000. From Figure Figure 5.13, we can see that in the linear scenario, DNN reaches the stable and perhaps optimal performance sooner, while PPORank fluctuates but reaches a relatively stable plateau after 1000 epochs. As our sample size is only 1000, while the number of the parameters in PRORank is almost 10 times the sample size, it needs to sample more trajectories in a long run to reach and even slightly exceeds the DNN's performance. As expected, both DNN and PPORank perform worse than the linear method of "CaDDRes". On the other hand, for the exponential scenario, Figure 5.14 shows faster convergence by both DNN and PPORank than that in the linear scenario, presumably because we are using nonlinear networks to describe the true nonlinear relationship.

Then we compare our PPORank with and without positive evaluation signals (PPO versus PPO-w/o), respectively, and other baseline methods, using simulated data. The sample size for each scenario is 1000 or 10000. We do not include SRMF as it requires drug similarity, though the GDSC data application shows that drug similarity does not improve performance much. From the NDCG values in scenario (1), where the cell lines' features have a linear effect on drug response, MF-based methods can capture the linear relationship

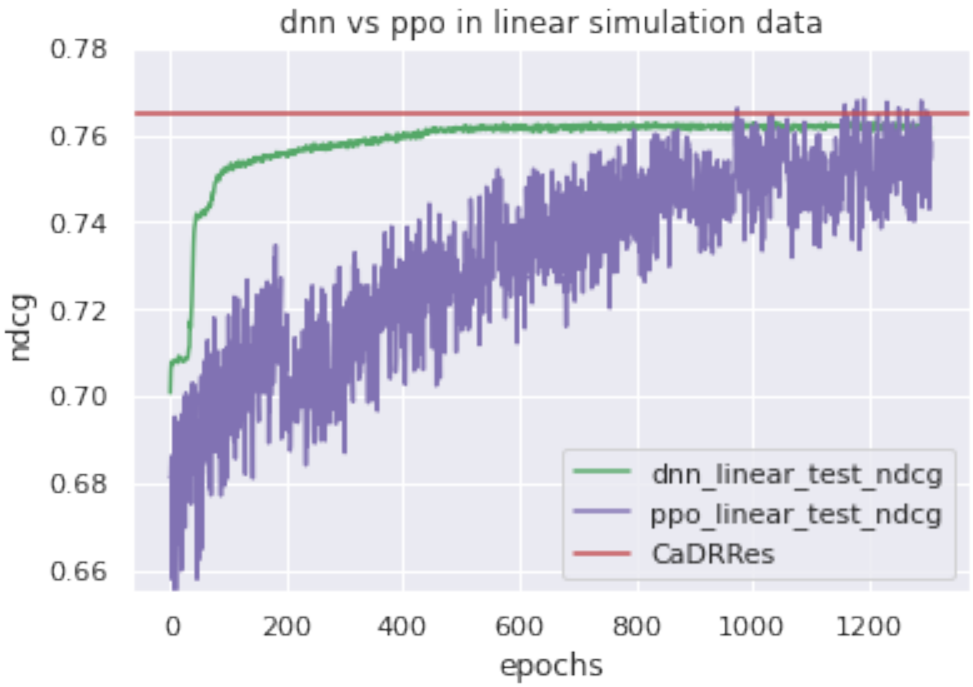

Figure 5.13: Simulation scenario (1).

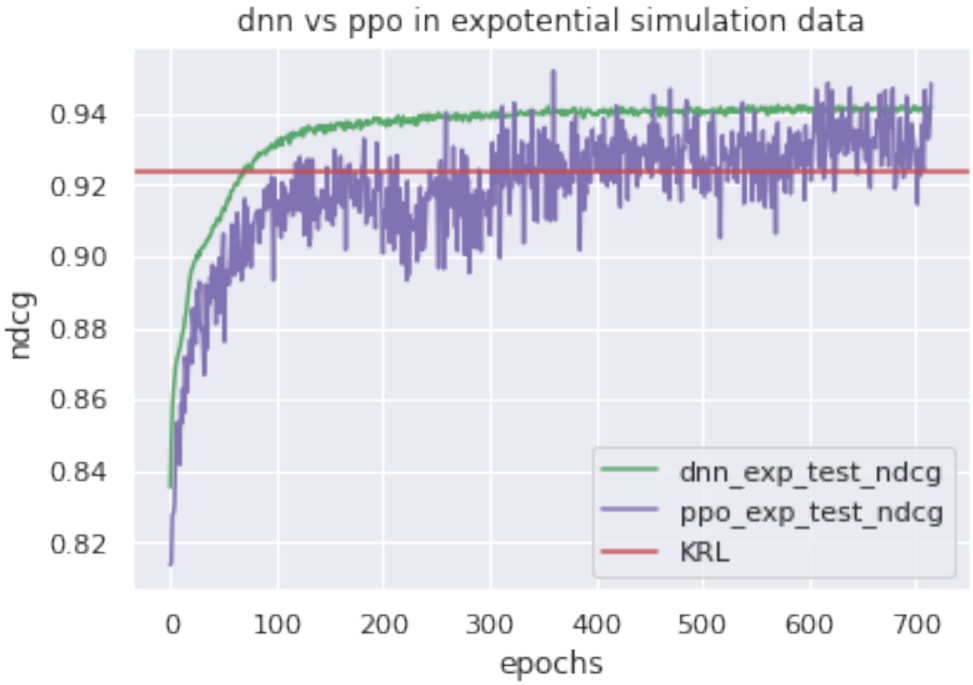

Figure 5.14: Simulation scenario (3).

Figure 5.15: Performance of DNN and PPORank in terms of $NDCG$ in two simulation scenarios.

and perform well. After increasing the sample size to 10000, PPORank performs best, presumably because it is the only one targeting NDCG, the evaluation criterion being used, as its loss function. For scenario (2), we reach the same conclusion as for scenario (1). In contrast, for scenario (3) with a highly non-linear true model, PPORank performs best with either sample size. The Simulation results are listed in Table 5.2.

Table 5.2: Mean NDCG (SD) in the primary simulations.

	Scenario 1		Scenario 2		Scenario 3	
	n=1000	n=10000	n=1000	n=10000	n=1000	n=10000
EN	0.793(0.03)	0.801(0.02)	0.593(0.05)	0.621(0.04)	0.890(0.04)	0.905(0.03)
KRR	0.774(0.04)	0.769(0.07)	0.532(0.05)	0.582(0.05)	0.882(0.06)	0.901(0.04)
KRL	0.783(0.02)	0.793(0.01)	0.625(0.02)	0.647(0.03)	0.920(0.03)	0.927(0.02)
CaDRRes	**0.809(0.02)**	0.824(0.02)	0.632(0.03)	0.682(0.02)	0.922(0.02)	0.931(0.02)
ppo-w/o	0.798(0.07)	0.811(0.10)	**0.670**(0.04)	0.693(0.01)	0.940(0.03)	0.948(0.04)
PPORank	0.790(0.06)	**0.832(0.04)**	0.651(0.02)	**0.705(0.05)**	**0.941(0.05)**	**0.959(0.03)**

5.9.2 Secondary simulations

To consider more complex relationships between cell lines and drugs, we design the following simulation setup. Suppose we have 40 drugs and 1000 cell lines, the cell line feature matrix $\mathbf{X} \in \mathbb{R}^{1000\times 2000}$ is the same as in the previous static simulation setup. The weight matrix $\mathbf{W} \in \mathbb{R}^{2000\times 100}$ is also the same as before. However, the drug response varies with the drug. For the first 20, next 10 and final 10 drugs, we have $\mathbf{Y} = 0.2 \times \mathbf{XW} + \boldsymbol{\epsilon}$, $\mathbf{Y} = 0.15 \times \mathbf{X}^3\mathbf{W} + 0.15 \times \mathbf{XW} + \boldsymbol{\epsilon}$, and $\mathbf{Y} = 0.2 \times \exp(\mathbf{X})\mathbf{W} - 0.1 \times \mathbf{X}^3\mathbf{W} + \boldsymbol{\epsilon}$, respectively.

We evaluate the performances with the sample size $n = 1000$ and $n = 10000$. From Table Table 5.3 we can see that PPORank performs best.

Table 5.3: Mean NDCG (SD) for the secondary simulations.

	$n = 1000$	$n = 10000$
EN	0.645(0.08)	0.679(0.07)
KRR	0.641(0.11)	0.650(0.10)
KRL	0.689(0.09)	0.707(0.08)
CaDRRes	0.647(0.11)	0.689(0.09)
ppo-w/o	0.701(0.11)	0.713(0.12)
PPORank	**0.721(0.11)**	**0.732(0.09)**

Chapter 6

Conclusions and Future Work

We have talked two scenarios in precision medicine recommendation, (1) with only one time point and (2) with sequentially data.

For the first scenario, only one treatment is always conducted. We have proposed a new method called ψ-learning to estimate optimal ITRs. In ψ-learning, a newly non-convex surrogate was introduced, and the computational tools was developed in the article to solve the non-convex optimization problem. In our proposed ψ-learning, whose weight for each subject was determined by the outcome, it adds an correction term to the SVM cost function which was used in OWL method in (109), so that the result was closer to the true generation error (52).

The newly proposed method appears to achieve better performance in both simulation and real data analysis compared to other existing methods listed in the article. The aim of ψ-learning is to achieve higher accuracy when classification in nature is non-convex and should be solved through non-convex cost function, and also it should take into consideration of the negative outcome in real life, while OWL only consider the positive outcome. So it is not hard to notice that the ψ-learning method has achieved better performance than OWL in simulation and real data analysis. Especially OWL tends to assign patients with the treatment that they actually received, while ψ-learning method is more likely to assign the treatment according to their outcome. While the two-step procedure like L_1-PLS is likely to overfit the regression model and may misspecify the model, as they strongly depend on the the true model of conditional mean outcome. That is why in the simulation when the

model was correctly specified, L_1-PLS performed well. However, in real life, the relationship between clinical covariates and treatment is hard to decide, and when considering the heterogeneous response to treatment, the decision boundary may be hard to define. And ψ-learning targets to model the decision boundary directly instead of modeling the conditional outcome first which is used in L_1-PLS. However, the model for decision boundary is critical in both methods, and this explains why RBF kernel can approximate the decision boundary with large sample size. But in ψ-learning even with misspecification of the decision boundary it can still perform better than the other methods. In our simulation we used the linear regression to model the outcome, but if the conditional mean is nonlinear it can still be solved in ψ-learning, while in the two-step methods L_1-PLS the decision boundary and the conditional mean outcome will be both linear.

Variable selection is an advantage of ψ-learning compared with OWL method, especially in the case of complicated decision boundary. As was pointed by (29) that the main effects factor tend to explain more of the variability in outcome instead of the treatment interactions, which may discard some prescriptive variables' effects. So the variable selection in ψ-learning aiming to select the covariates that effect the treatment interaction effects is necessary.

We also introduce the kernel method used in ψ-learning, and we adopted the Gaussian Kernel which may be free of model misspecification, but the problem is that it is hard to explain as it can not specify the shape of the decision boundary, while linear kernel is easy to explain but may be easy to misspecify the model.

Several extensions may be taken into considerations. We have only considered the binary treatment classes, although in the real data analysis when we have three classes of treatments, we did the pairwise comparisons, but this way may not be best to find the optimal ITR considering all the treatments simultaneously. So it worthwhile to extend the ψ-learning to multiple-arm trials. Two directions are commonly used in multi-category classification. One it to consider all classes like the multi-category SVMs (9; 92) and the other direction is to use one-verse-all, one-verse-one methods (104). So we may extend the ψ-learning in multi-category classification in these directions.

For the second scenario, as data accrue, it will be more efficient to learn by adapting to the dynamic change of the environment. We have proposed a novel personalized ranking system called Proximal Policy Optimization Ranking (PPORank), which ranks drugs/treatments based on their predicted effects per cell line (or patient). It makes recommendations directly based on the ranking of the drugs, instead of on predicting some drug-specific responses per se, e.g. their $\log(\mathrm{IC}_{50})$ values, as adopted by other existing methods. Hence this policy-based learning framework directly optimizes the target evaluation metric using the policy gradient algorithm. Furthermore, the evaluation metric NDCG is optimized by leveraging information from all the ranks. In the implementation, we have adopted some state-of-the-art techniques in DRL as briefly discussed below. By using a Deep&Cross network to parametrize the policy, we can learn the interactions between cell lines and drugs efficiently, and it can be extended to integrate other data sources. In particular, we use a GRU layer to incorporate the dynamic evaluation signal from cell lines and reinforce the policy with evaluation signals. Variance reduction is realized by using the actor-critic framework, where the actor network learns the states from the input and feeds it to the critic network to evaluate the current policy. To handle the sample inefficiency problem of policy gradient methods, we use the PPO algorithm with multiple epochs of stochastic gradient ascent to update the policy. As proofs-of-concept, we have applied our method to two large-scale cancer drug screening datasets for personalized drug discovery. With the limited data and high dimensions of the states and actions, our method has shown superior performance over other methods. In particular, our method demonstrates its promising and clinically meaningful performance when the learning algorithm trained on the cell line drug screening data is applied to a TCGA breast cancer cohort. Most importantly, our method, as any RL method, can be more efficiently applied in practice: it learns sequentially and continuously as the data accrue; in contrast, existing supervised learning and many DTR methods would require finishing collecting data (from either a randomized experiment or an observational study) first before learning personalized drug ranking, which would be more resource- and time-consuming. This feature of our method would be most useful for more efficient drug discovery as for

our real data applications, and possibly for behavioral interventions in mHealth or selecting experimental drugs for patients with incurable or terminal diseases. In addition, through transfer learning, our proposed RL method can take advantage of existing data and be trained a priori before being applied to and fine-tuned by future clinical data. Finally, an important aspect of our proposed RL method is its active learning and dynamic interaction with individuals: the learning algorithm would suggest the best treatment/intervention based on its current ranking to an individual, then learn from the outcome, e.g. in mHealth applications. Due to lack of data, we have not explored this aspect. This is a distinct and attractive feature unique to RL, but not to supervised learning.

On the other hand, there are some challenges remaining before DRL being applied more generally for health care in practice. First, due to the "data-hungry" nature of both DL and RL (and thus DRL), more data efficient algorithms are urgently needed. Second, the interpretability of a DL model as a black-box needs to be improved. Both topics have generated extensive interests and intensive research in the DL community, which will hopefully lead to some breakthroughs in a near future. In summary, our proposed and possibly other DRL-based methods are promising for precision medicine, and worth further investigation.

References

[1] Aben, N., Vis, D. J., Michaut, M., and Wessels, L. F. (2016). Tandem: a two-stage approach to maximize interpretability of drug response models based on multiple molecular data types. *Bioinformatics*, 32(17):i413–i420.

[2] Allen, G. I. (2013). Automatic feature selection via weighted kernels and regularization. *Journal of Computational and Graphical Statistics*, 22(2):284–299.

[3] Amado, R., Wolf, M., Peeters, M., Van Cutsem, E., Siena, S., Freeman, D., Juan, T., Sikorski, R., Suggs, S., Radinsky, R., et al. (2008). Wild-type kras is required for panitumumab efficacy in patients with metastatic colorectal cancer. *Journal of clinical oncology*, 26(10):1626–1634.

[4] Anthonisen, N. R., Connett, J. E., Kiley, J. P., Altose, M. D., Bailey, W. C., Buist, A. S., Conway, W. A., Enright, P. L., Kanner, R. E., O'hara, P., et al. (1994). Effects of smoking intervention and the use of an inhaled anticholinergic bronchodilator on the rate of decline of fev1: the lung health study. *Jama*, 272(19):1497–1505.

[5] Azuaje, F. (2017). Computational models for predicting drug responses in cancer research. *Briefings in bioinformatics*, 18(5):820–829.

[6] Baehner, F. L., Lee, M., Demeure, M. J., Bussey, K. J., Kiefer, J. A., and Barrett, M. T. (2011). Genomic signatures of cancer: basis for individualized risk assessment, selective staging and therapy. *Journal of surgical oncology*, 103(6):563–573.

[7] Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A. A., Kim, S., Wilson, C. J., Lehár, J., Kryukov, G. V., Sonkin, D., et al. (2012). The cancer cell

line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607.

[8] Boyd, S., Xiao, L., and Mutapcic, A. (2003). Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005.

[9] Bredensteiner, E. J. and Bennett, K. P. (1999). Multicategory classification by support vector machines. In *Computational Optimization*, pages 53–79. Springer.

[10] Buist, A. S., Connett, J. E., Miller, R. D., Kanner, R. E., Owens, G. R., and Voelker, H. T. (1993). Chronic obstructive pulmonary disease early intervention trial (lung health study): baseline characteristics of randomized participants. *Chest*, 103(6):1863–1872.

[11] Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., and Guo, D. (2017). Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 661–670, Cambridge, United Kingdom. Association for Computing Machinery.

[12] Chang, Y., Park, H., Yang, H.-J., Lee, S., Lee, K.-Y., Kim, T. S., Jung, J., and Shin, J.-M. (2018). Cancer drug response profile scan (cdrscan): A deep learning model that predicts drug effectiveness from cancer genomic signature. *Scientific Reports*, 8(1):8857.

[13] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[14] Cichonska, A., Pahikkala, T., Szedmak, S., Julkunen, H., Airola, A., Heinonen, M., Aittokallio, T., and Rousu, J. (2018). Learning with multiple pairwise kernels for drug bioactivity prediction. *Bioinformatics*, 34(13):i509–i518.

[15] Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S., and MacKinnon, I. (2008). Novelty and diversity in information retrieval evaluation.

Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages 659–666. ACM.

[16] Connett, J. E., Kusek, J. W., Bailey, W. C., O'Hara, P., Wu, M., Group, L. H. S. R., et al. (1993). Design of the lung health study: a randomized clinical trial of early intervention for chronic obstructive pulmonary disease. *Controlled clinical trials*, 14(2):3–19.

[17] Coronato, A., Naeem, M., De Pietro, G., and Paragliola, G. (2020). Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in Medicine*, 109:101964.

[18] Cortés-Ciriano, I., van Westen, G. J., Bouvier, G., Nilges, M., Overington, J. P., Bender, A., and Malliavin, T. E. (2016). Improved large-scale prediction of growth inhibition patterns using the nci60 cancer cell line panel. *Bioinformatics*, 32(1):85–95.

[19] Costello, J. C., Heiser, L. M., Georgii, E., Gönen, M., Menden, M. P., Wang, N. J., Bansal, M., Hintsanen, P., Khan, S. A., Mpindi, J.-P., et al. (2014). A community effort to assess and improve drug sensitivity prediction algorithms. *Nature biotechnology*, 32(12):1202–1212.

[20] Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.

[21] Ding, M. Q., Chen, L., Cooper, G. F., Young, J. D., and Lu, X. (2018). Precision oncology beyond targeted therapy: Combining omics data with machine learning matches the majority of cancer cells to effective therapeutics. *Molecular Cancer Research*, 16(2):269–278.

[22] Drew, Y., Ledermann, J., Hall, G., Rea, D., Glasspool, R., Highley, M., Jayson, G., Sludden, J., Murray, J., Jamieson, D., et al. (2016). Phase 2 multicentre trial investigating intermittent and continuous dosing schedules of the poly (adp-ribose) polymerase inhibitor

rucaparib in germline brca mutation carriers with advanced ovarian and breast cancer. *British journal of cancer*, 114(7):723–730.

[23] Enright, P. L., Johnson, L. R., Connett, J. E., Voelker, H., and Buist, A. S. (1991). Spirometry in the lung health study. *Am Rev Respir Dis*, 143:1215–1223.

[24] Ertefaie, A. and Strawderman, R. L. (2018). Constructing dynamic treatment regimes over indefinite time horizons. *Biometrika*, 105(4):963–977.

[25] Figueiredo Prudencio, R., Maximo, M. R., and Luna Colombini, E. (2022). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv e-prints*, pages arXiv–2203.

[26] Garnett, M. J., Edelman, E. J., Heidorn, S. J., Greenman, C. D., Dastur, A., Lau, K. W., Greninger, P., Thompson, I. R., Luo, X., Soares, J., et al. (2012). Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature*, 483(7391):570–575.

[27] Geeleher, P., Zhang, Z., Wang, F., Gruener, R. F., Nath, A., Morrison, G., Bhutra, S., Grossman, R. L., and Huang, R. S. (2017). Discovering novel pharmacogenomic biomarkers by imputing drug response in cancer patients from large genomics studies. *Genome research*, 27(10):1743–1751.

[28] Gomez, H. L., Doval, D. C., Chavez, M. A., Ang, P. C.-S., Aziz, Z., Nag, S., Ng, C., Franco, S. X., Chow, L. W., Arbushites, M. C., et al. (2008). Efficacy and safety of lapatinib as first-line therapy for erbb2-amplified locally advanced or metastatic breast cancer. *Journal of Clinical Oncology*, 26(18):2999–3005.

[29] Gunter, L., Zhu, J., and Murphy, S. (2011). Variable selection for qualitative interactions. *Statistical methodology*, 8(1):42–55.

[30] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

[31] Gupta, S., Chaudhary, K., Kumar, R., Gautam, A., Nanda, J. S., Dhanda, S. K., Brahmachari, S. K., and Raghava, G. P. (2016). Prioritization of anticancer drugs against a cancer using genomic features of cancer cells: A step towards personalized medicine. *Scientific reports*, 6:23857.

[32] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

[33] He, X., Folkman, L., and Borgwardt, K. (2018). Kernelized rank learning for personalized drug recommendation. *Bioinformatics*, 34(16):2808–2816.

[34] Head, T. and Carcinoma, N. S. C. (2016). Broad institute tcga genome data analysis center. *Firehose Stddata__2016_01_28 run*.

[35] Hu, Y., Da, Q., Zeng, A., Yu, Y., and Xu, Y. (2018). Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. *arXiv:1803.00710 [cs]*.

[36] Humphrey, K. (2017). Using reinforcement learning to personalize dosing strategies in a simulated cancer trial with high dimensional data. *Master's Theses, The University of Arizona.*

[37] Iorio, F., Knijnenburg, T. A., Vis, D. J., Bignell, G. R., Menden, M. P., Schubert, M., Aben, N., Gonçalves, E., Barthorpe, S., Lightfoot, H., et al. (2016). A landscape of pharmacogenomic interactions in cancer. *Cell*, 166(3):740–754.

[38] Jang, I. S., Neto, E. C., Guinney, J., Friend, S. H., and Margolin, A. A. (2014). Systematic assessment of analytical methods for drug sensitivity prediction from cancer cell line data. In *Biocomputing 2014*, pages 63–74. World Scientific.

[39] Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

[40] Jimeno, A., Messersmith, W. A., Hirsch, F. R., Franklin, W. A., and Eckhardt, S. G. (2009). Kras mutations and sensitivity to epidermal growth factor receptor inhibitors in colorectal cancer: practical application of patient selection. *Journal of Clinical Oncology*, 27(7):1130–1136.

[41] Kim, S., Thiessen, P. A., Bolton, E. E., Chen, J., Fu, G., Gindulyte, A., Han, L., He, J., He, S., Shoemaker, B. A., et al. (2016). Pubchem substance and compound databases. *Nucleic acids research*, 44(D1):D1202–D1213.

[42] Kimeldorf, G. and Wahba, G. (1971). Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95.

[43] Klutstein, M., Nejman, D., Greenfield, R., and Cedar, H. (2016). Dna methylation in cancer and aging. *Cancer research*, 76(12):3446–3450.

[44] Knijnenburg, T. A., Klau, G. W., Iorio, F., Garnett, M. J., McDermott, U., Shmulevich, I., and Wessels, L. F. (2016). Logic models to predict continuous outputs based on binary inputs with an application to personalized cancer therapy. *Scientific reports*, 6(1):1–14.

[45] Kosorok, M. R. and Laber, E. B. (2019). Precision medicine. *Annual review of statistics and its application*, 6:263–286.

[46] Le Thi Hoai, A. and Tao, P. D. (1997). Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. *Journal of global optimization*, 11(3):253–285.

[47] Lei, H., Nahum-Shani, I., Lynch, K., Oslin, D., and Murphy, S. A. (2012). A" smart" design for building individualized treatment sequences. *Annual review of clinical psychology*, 8:21–48.

[48] Lievre, A., Bachet, J.-B., Boige, V., Cayre, A., Le Corre, D., Buc, E., Ychou, M., Bouché, O., Landi, B., Louvet, C., et al. (2008). Kras mutations as an independent prognostic factor in patients with advanced colorectal cancer treated with cetuximab. *Journal of clinical oncology*, 26(3):374–379.

[Liu et al.] Liu, M., Shen, X., and Pan, W. Outcome weighted ψ-learning for individualized treatment rules. *Stat*, page e343.

[50] Liu, M., Shen, X., and Pan, W. (2022). Deep reinforcement learning for personalized treatment recommendation. *Statistics in medicine*, 41(20):4034–4056.

[51] Liu, S., See, K. C., Ngiam, K. Y., Celi, L. A., Sun, X., Feng, M., et al. (2020). Reinforcement learning for clinical decision support in critical care: comprehensive review. *Journal of medical Internet research*, 22(7):e18477.

[52] Liu, S., Shen, X., and Wong, W. H. (2005). Computational developments of ψ-learning. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 1–11. SIAM.

[53] Luckett, D. J., Laber, E. B., Kahkoska, A. R., Maahs, D. M., Mayer-Davis, E., and Kosorok, M. R. (2019). Estimating dynamic treatment regimes in mobile health using v-learning. *Journal of the American Statistical Association*.

[54] M., Q. and S.A.Murphy (2011). Performance guarantees for individualized treatment rules. *Annals of statistics*, 39(2):1180.

[55] Maldonado, S., Weber, R., and Basak, J. (2011). Simultaneous feature selection and classification using kernel-penalized support vector machines. *Information Sciences*, 181(1):115–128.

[56] Mangasarian, O. L. and Kou, G. (2007). Feature selection for nonlinear kernel support vector machines. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 231–236. IEEE.

[57] Maxwell, K. N., Wubbenhorst, B., Wenz, B. M., De Sloover, D., Pluta, J., Emery, L., Barrett, A., Kraya, A. A., Anastopoulos, I. N., Yu, S., et al. (2017). Brca locus-specific loss of heterozygosity in germline brca1 and brca2 carriers. *Nature communications*, 8(1):1–11.

[58] Menden, M. P., Iorio, F., Garnett, M., McDermott, U., Benes, C. H., Ballester, P. J., and Saez-Rodriguez, J. (2013). Machine learning prediction of cancer cell sensitivity to drugs based on genomic and chemical properties. *PLoS one*, 8(4).

[59] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[60] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

[61] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[62] Moodie, E. E., Chakraborty, B., and Kramer, M. S. (2012). Q-learning for estimating optimal dynamic treatment rules from observational data. *Canadian Journal of Statistics*, 40(4):629–645.

[63] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[64] Murphy, S. A. (2003). Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(2):331–355.

[65] Murphy, S. A. (2005). An experimental design for the development of adaptive treatment strategies. *Statistics in medicine*, 24(10):1455–1481.

[66] Network, C. G. A. et al. (2012). Comprehensive molecular portraits of human breast tumours. *Nature*, 490(7418):61.

[67] Peng, X., Ding, Y., Wihl, D., Gottesman, O., Komorowski, M., Lehman, L.-w. H., Ross, A., Faisal, A., and Doshi-Velez, F. (2018). Improving sepsis treatment strategies by combining deep and kernel-based reinforcement learning. In *AMIA Annual Symposium Proceedings*, volume 2018, page 887. American Medical Informatics Association.

[68] Prasad, N., Cheng, L.-F., Chivers, C., Draugelis, M., and Engelhardt, B. E. (2017). A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *arXiv preprint arXiv:1704.06300*.

[69] Qin, T., Liu, T.-Y., and Li, H. (2010). A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13(4):375–397.

[70] Robins, J. M. (1994). Correcting for non-compliance in randomized trials using structural nested mean models. *Communications in Statistics-Theory and methods*, 23(8):2379–2412.

[71] Robins, J. M. (1997). Causal inference from complex longitudinal data. In *Latent variable modeling and applications to causality*, pages 69–117. Springer.

[72] Scanlon, P. D., Connett, J. E., Waller, L. A., Altose, M. D., Bailey, W. C., Sonia Buist, A., and e Lung Health Study Research Group, D. P. T. f. t. (2000). Smoking cessation and lung function in mild-to-moderate chronic obstructive pulmonary disease: the lung health study. *American journal of respiratory and critical care medicine*, 161(2):381–390.

[73] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

[74] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[75] Schulte, P. J., Tsiatis, A. A., Laber, E. B., and Davidian, M. (2014). Q-and a-learning methods for estimating optimal dynamic treatment regimes. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 29(4):640.

[76] Seashore-Ludlow, B., Rees, M. G., Cheah, J. H., Cokol, M., Price, E. V., Coletti, M. E., Jones, V., Bodycombe, N. E., Soule, C. K., Gould, J., et al. (2015). Harnessing connectivity in a large-scale small-molecule sensitivity dataset. *Cancer discovery*, 5(11):1210–1223.

[77] Shani, G., Heckerman, D., and Brafman, R. I. (2005). An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295.

[78] Shen, X., Pan, W., Zhu, Y., and Zhou, H. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5):807–832.

[79] Shen, X. and Pan, Wand Zhu, Y. (2012). Likelihood-based selection and sharp parameter estimation. *Journal of the American Statistical Association*, 107(497):223–232.

[80] Shen, X., Tseng, G. C., Zhang, X., and Wong, W. H. (2003). On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734.

[81] Shor, N. Z. (2012). *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media.

[82] Suphavilai, C., Bertrand, D., and Nagarajan, N. (2018). Predicting cancer drug response using a recommender system. *Bioinformatics*, 34(22):3907–3914.

[83] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[84] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems, pages 1057–1063. MIT Press.

[85] Tao, P. D. and An, L. T. H. (1997). Convex analysis approach to dc programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355.

[86] Tashkin, D. P., Altose, M. D., Connett, J. E., Kanner, R. E., Lee, W. W., and Wise, R. A. (1996). Methacholine reactivity predicts changes in lung function over time in smokers with early chronic obstructive pulmonary disease. the lung health study research group. *American journal of respiratory and critical care medicine*, 153(6):1802–1811.

[87] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

[88] Tseng, H.-H., Luo, Y., Cui, S., Chien, J.-T., Ten Haken, R. K., and Naqa, I. E. (2017). Deep reinforcement learning for automated radiation adaptation in lung cancer. *Medical physics*, 44(12):6690–6705.

[89] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(Sep):1453–1484.

[90] Tutt, A., Robson, M., Garber, J. E., Domchek, S. M., Audeh, M. W., Weitzel, J. N., Friedlander, M., Arun, B., Loman, N., Schmutzler, R. K., et al. (2010). Oral poly (adp-ribose) polymerase inhibitor olaparib in patients with brca1 or brca2 mutations and advanced breast cancer: a proof-of-concept trial. *The Lancet*, 376(9737):235–244.

[91] Wang, L., Li, X., Zhang, L., and Gao, Q. (2017a). Improved anticancer drug response prediction in cell lines using matrix factorization with similarity regularization. *BMC cancer*, 17(1):513.

[92] Wang, L. and Shen, X. (2006). Multi-category support vector machines, feature selection and solution path. *Statistica Sinica*, pages 617–633.

[93] Wang, L., Zhang, W., He, X., and Zha, H. (2018). Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2447–2456. ACM.

[94] Wang, R., Fu, B., Fu, G., and Wang, M. (2017b). Deep & cross network for ad click predictions. Proceedings of the ADKDD'17, pages 1–7. ACM.

[95] Wei, Z., Xu, J., Lan, Y., Guo, J., and Cheng, X. (2017). Reinforcement learning to rank with markov decision process. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 945–948. ACM.

[96] Weimer, M., Karatzoglou, A., Le, Q. V., and Smola, A. J. (2008). Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*, pages 1593–1600.

[97] Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., and Stuart, J. M. (2013). The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113–1120.

[98] Weng, W.-H., Gao, M., He, Z., Yan, S., and Szolovits, P. (2017). Representation and reinforcement learning for personalized glycemic control in septic patients. *arXiv preprint arXiv:1712.00654*.

[99] Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V. (2001). Feature selection for svms. In *Advances in neural information processing systems*, pages 668–674.

[100] Yauney, G. and Shah, P. (2018). Reinforcement learning with action-derived rewards for chemotherapy and clinical trial dosing regimen selection. In *Machine Learning for Healthcare Conference*, pages 161–226. PMLR.

[101] Yu, C., Liu, J., Nemati, S., and Yin, G. (2021). Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36.

[102] Yuan, H., Paskov, I., Paskov, H., González, A. J., and Leslie, C. S. (2016). Multitask learning improves prediction of cancer drug sensitivity. *Scientific reports*, 6(1):1–11.

[103] Zhang, B., Tsiatis, A. A., Laber, E. B., and Davidian, M. (2013). Robust estimation of optimal dynamic treatment regimes for sequential treatment decisions. *Biometrika*, 100(3):681–694.

[104] Zhang, T. (2004). Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5(Oct):1225–1251.

[105] Zhang, W., Du, T., and Wang, J. (2016). Deep learning over multi-field categorical data. European conference on information retrieval, pages 45–57. Springer.

[106] Zhang, Y., Laber, E. B., Davidian, M., and Tsiatis, A. A. (2018). Interpretable dynamic treatment regimes. *Journal of the American Statistical Association*, 113(524):1541–1549.

[107] Zhao, X., Zhang, L., Xia, L., Ding, Z., Yin, D., and Tang, J. (2019). Deep reinforcement learning for list-wise recommendations. *arXiv:1801.00209 [cs, stat]*.

[108] Zhao, Y., Kosorok, M. R., and Zeng, D. (2009). Reinforcement learning design for cancer clinical trials. *Statistics in medicine*, 28(26):3294–3315.

[109] Zhao, Y., Zeng, D., Rush, A. J., and Kosorok, M. R. (2012). Estimating individualized treatment rules using outcome weighted learning. *Journal of the American Statistical Association*, 107(499):1106–1118.

[110] Zhao, Y., Zeng, D., Socinski, M. A., and Kosorok, M. R. (2011a). Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics*, 67(4):1422–1433.

[111] Zhao, Y., Zeng, D., Socinski, M. A., and Kosorok, M. R. (2011b). Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics*, 67(4):1422–1433.

[112] Zhao, Y.-Q., Zeng, D., Laber, E. B., and Kosorok, M. R. (2015). New statistical learning methods for estimating optimal dynamic treatment regimes. *Journal of the American Statistical Association*, 110(510):583–598.

[113] Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., and Li, Z. (2018). Drn: A deep reinforcement learning framework for news recommendation. Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18, pages 167–176, Lyon, France. ACM Press.

[114] Zhou, S. K., Le, H. N., Luu, K., Nguyen, H. V., and Ayache, N. (2021). Deep reinforcement learning in medical imaging: A literature review. *Medical image analysis*, 73:102193.

[115] Zhou, X., Mayer-Hamblett, N., Khan, U., and Kosorok, M. R. (2017). Residual weighted learning for estimating individualized treatment rules. *Journal of the American Statistical Association*, 112(517):169–187.

Appendix A

Supporting Information for Chapter 2

A.1 Solution to DC decomposition

In the convex subproblem in ((2.6)), we derive a subgradient of S_2 at $\mathbf{w}^{(m)}$ at the m_{th} step. For simplify, we denote $\triangledown S_2(\mathbf{w}^{(m)}) = (V_1^{(m)}, V_2^{(m)})$, where

$$\begin{aligned} V_1^{(m)} &= \frac{1}{n}\sum_{i=1}^{n}\frac{|R_i|}{\pi(A_i,\mathbf{X}_i)}A_i\mathbf{X}_i[\triangledown\psi_2(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i))I(R_i>0)+\triangledown\psi_1(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i))I(R_i<0)], \\ V_2^{(m)} &= \frac{1}{n}\sum_{i=1}^{n}\frac{|R_i|}{\pi(A_i,\mathbf{X}_i)}A_i[\triangledown\psi_2(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i))I(R_i>0)+\triangledown\psi_1(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i))I(R_i<0)], \end{aligned} \tag{A.1}$$

where $\triangledown\psi_1(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i)) = -\frac{1}{\tau}I(1-A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i)\geq 0)$ and $\triangledown\psi_2(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i)) = -\frac{1}{\tau}I(A_i f^{(m)}(\widetilde{\boldsymbol{X}}_i)\leq 0)$.

Then our algorithm solves a sequence of subproblems, at iteration m, we solve the minimization of

$$\begin{aligned} S^{(m+1)}(\mathbf{w}) &= S_1(\mathbf{w}) - S_2(\hat{\mathbf{w}}^{(m)} - \langle(\mathbf{w}-\hat{\mathbf{w}}^{(m)}, \triangledown S_2(\hat{\mathbf{w}}^{(m)}\rangle \\ &= S_1(\mathbf{w}) - \langle\mathbf{w}, \triangledown S_2(\mathbf{w}^{(m)}\rangle) + Const, \end{aligned} \tag{A.2}$$

In the kernel case,

$$
\begin{aligned}
V_1^{(m)} &= -\frac{1}{n\tau}\sum_{i=1}^{n} A_i K(\mathbf{x}_i,\cdot)\{I(A_i(\sum_{j=1}^{n} w_j^{(m)} K(\mathbf{x}_i,\mathbf{x}_j) + w_{n+1}^{(m)} \le 0) I(R_i \ge 0) \\
&\quad - I(1 - A_i(\sum_{j=1}^{n} w_j^{(m)} K(\mathbf{x}_i,\mathbf{x}_j) + w_{n+1}^{(m)} \ge 0) I(R_i \le 0)\} \frac{|R_i|}{\pi(A_i,\mathbf{X}_i)}, \\
V_2^{(m)} &= -\frac{1}{n\tau}\sum_{i=1}^{n} A_i\{I(A_i(\sum_{j=1}^{n} w_j^{(m)} K(\mathbf{x}_i,\mathbf{x}_j) + w_{n+1}^{(m)} \le 0) I(R_i \ge 0) \\
&\quad - I(1 - A_i(\sum_{j=1}^{n} w_j^{(m)} K(\mathbf{x}_i,\mathbf{x}_j) + w_{n+1}^{(m)} \ge 0) I(R_i \le 0)\} \frac{|R_i|}{\pi(A_i,\mathbf{X}_i)}.
\end{aligned}
\tag{A.3}
$$

A.2 Subgradient method

For the ψ-linear method, we denote

$$
\widehat{\mathbf{w}}^{(m+1,1)} = (\widehat{w}_1^{(m+1,1)} \cdots, \widehat{w}_p^{(m+1,1)}, \widehat{w}_{p+1}^{(m+1,1)}) = \widehat{\mathbf{w}}^{(m)},
$$

and use the following gradient of the cost function in ((2.9)) at $\widehat{\mathbf{w}}^{(m+1,t-1)}$:

$$
\begin{aligned}
\nabla S^{(m+1)}(\mathbf{w}^{*(m+1,t)}) &= \kappa \mathbf{w}^{*(m+1,t)} + \frac{1}{(n\tau)}\sum_{i=1}^{n}\{-I(1 - A_i f^{(m+1,t)}(\widetilde{\boldsymbol{X}}_i)) \ge 0) I(R_i \ge 0) \\
&\quad - I(-A_i f^{(m+1,t)}(\widetilde{\boldsymbol{X}}_i)) \ge 0) I(R_i \le 0)\} \frac{|R_i|}{\pi(A_i,\mathbf{X}_i)} A_i \mathbf{X}_i' - V_1^{(m)}, \\
\nabla S^{(m+1)}(w_{p+1}^{(m+1,t)} &= \frac{1}{(n\tau)}\sum_{i=1}^{n}\{-I(1 - A_i f^{(m+1,t)}(\widetilde{\boldsymbol{X}}_i)) \ge 0) I(R_i \ge 0) \\
&\quad - I(-A_i f^{(m+1,t)}(\widetilde{\boldsymbol{X}}_i)) \ge 0) I(R_i \le 0)\} \frac{|R_i|}{\pi(A_i,\mathbf{X}_i)} A_i - V_2^{(m)},
\end{aligned}
\tag{A.4}
$$

where $V_1^{(m)}$ and $V_2^{(m)}$ are as defined in ((A.1)), $f^{(m+1,t)}(\widetilde{\boldsymbol{X}}_i) = \mathbf{X}\mathbf{w}^{*(m+1,t)} + w_{p+1}^{(m+1,t)}$ until convergence to obtain $\mathbf{w}^{(m+1)}$.

With the non-summable diminishing step size of $\frac{1}{\sqrt{npt}}$, then we update

$$\mathbf{w}^{(m+1,t+1)} = \mathbf{w}^{(m+1,t)} - \frac{1}{\sqrt{npt}} \nabla S^{(m+1)}(\mathbf{w}^{(m+1,t)})$$

Main effects estimate

In our method, we only consider the case that the outcome R is continuous which has the following expression :

$$R = T(\mathbf{X}) + T_0(\mathbf{X}) \cdot A + \epsilon,$$

where ϵ is the random error with mean zero, $T(\mathbf{X})$ is the common effects of $\mathbf{X}$ for both treatments, and $T_0(\mathbf{X}) \cdot A$ is the interaction effects between the treatment and the clinical covariates. It can be derived that $T(\mathbf{X}) = (E(R|\mathbf{X}, A = 1) + E(R|\mathbf{X}, A = -1))/2 = E(\frac{R}{2\pi(A,\mathbf{X})}|\mathbf{X})$ (115), so we can use $R - T(\mathbf{X})$ to directly estimate the treatment effects. It is proposed in (115) that $\mathcal{D}^*$ remains unchanged if R is replaced by $R - g(\mathbf{X})$ for any function g, as long as g is not related to $\mathcal{D}$. And when with finite sample size, a reasonable choice of g is $g(\mathbf{X}) = E(\frac{R}{2\pi(A,\mathbf{X})}|\mathbf{X})$ which can be estimated by minimizing the sum of weighted squares of $\sum_{i=1}^{n} \frac{1}{2\pi(A_i,\mathbf{X}_i)}(R_i - \mathbf{X}_i^T\boldsymbol{\beta})^2$, where $\boldsymbol{\beta}$ is the coefficients in the common effects.

Appendix B

Supporting Information for Chapter 3

B.1 Multi-omic data types in GDSC

Table B.1: Four types of the omic data in GDSC.[1]

Data type	#cell-lines	#features	missing rate
gene expression(GEX)	962	17737	18%
whole-exome sequencing (WES)	953	300	19%
copy number variation(CNV)	985	425	19%
DNA methylation (MET)	785	378	19%

[1] Note:WES are binary features encoding if the given cell-line carries variants in recurrently mutated sites of one of the 300 candidate cancer genes (CGs) identified in 6,815 patient tumors; CNV are binary features encoding if the given cell-line carries one of the 425 recurrently aberrant copy number segments (RACSs) identified in 8,014 patient tumors; MET are binary features encoding if the given cell-line carries one of the 378 hyper-methylated informative CpG islands (iCpGs) located in the gene promoters identified in 6,166 patient tumors.