The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Deep-Treat: Learning Optimal Personalized Treatments from Observational Data Using Neural Networks

Onur Atan,[1] **James Jordon,**[2] **Mihaela van der Schaar**[2,3,1]
[1]Electrical Engineering Department, University of California Los Angeles, USA
[2]Department of Engineering Science, University of Oxford, UK
[3]Alan Turing Institute, London, UK
oatan@ucla.edu, james.jordon@wolfson.ox.ac.uk, mihaela.vanderschaar@eng.ox.ac.uk

Abstract

We propose a novel approach for constructing effective treatment policies when the observed data is biased and lacks counterfactual information. Learning in settings where the observed data does not contain all possible outcomes for all treatments is difficult since the observed data is typically biased due to existing clinical guidelines. This is an important problem in the medical domain as collecting unbiased data is expensive and so learning from the wealth of existing biased data is a worthwhile task. Our approach separates the problem into two stages: first we reduce the bias by learning a representation map using a novel auto-encoder network – this allows us to control the trade-off between the bias-reduction and the information loss – and then we construct effective treatment policies on the transformed data using a novel feedforward network. Separation of the problem into these two stages creates an algorithm that can be adapted to the problem at hand – the bias-reduction step can be performed as a preprocessing step for other algorithms. We compare our algorithm against state-of-art algorithms on two semi-synthetic datasets and demonstrate that our algorithm achieves a significant improvement in performance.

Introduction

Potential treatment outcomes for the current patient must be learned from the outcomes of the treatments given to previous patients. However, treatment outcomes of the patients are never the same and so the learning process must involve learning how the current patient *is* alike to previous patients. This learning problem is complicated by the fact that the data has only the outcomes of the treatments received – not the potential outcomes of the alternative treatments, the *counterfactuals*. Access to similar patients with different treatments is also made difficult as the received treatments are often chosen by some pre-existing clinical guidelines and so it may be the case that similar patients received the same treatment – the observed data is *biased*.

This paper proposes a novel approach for treatment policy optimization. We construct two independent feedforward networks: the first we call the *bias-removing auto-encoder*, which is used to map the data (using a representation map Φ) to a less biased representation without losing too much informativeness – we control this trade-off with a hyperparameter, λ_1 – and the second we call the *feedforward policy optimization network*, which is used to learn the optimal policy, taking the transformed data (under Φ) as the training set.

Learning is carried out by first learning the auto-encoder and then learning the policy optimizer. The bias-removing auto-encoder contains a block used to learn the propensity function – the function that takes representation-treatment pairs and maps them to the probability of observing the treatment given the representation. This is then used to force the learned representations to be less biased.

We illustrate our algorithm on learning treatment effects from Infant Health and Development (IHDP) dataset and learning chemotherapy regimes for a breast cancer dataset. Our results show that our algorithm achieves significant improvements with respect to strong baselines.

Related Work

From a conceptual point of view, the paper most closely related to ours is (Johansson, Shalit, and Sontag 2016) which discusses a similar problem: learning representations from observational data. The approach taken in (Johansson, Shalit, and Sontag 2016) is somewhat different to ours, though, in that they minimize the predictive loss combined with discrepancy between the factual and counterfactual representation populations. A more important difference from our work is that (Johansson, Shalit, and Sontag 2016) assumes that there are only two actions: treat and don't treat – in most situations there will be many possible actions. (Johansson, Shalit, and Sontag 2016) states explicitly that generalizing their approach to more than two actions is not an obvious extension.

From a technical point of view, our work is most closely related to (Swaminathan and Joachims; 2015a, 2015b) in that we both aim to learn treatment policies from observational data. The approach taken in (Swaminathan and Joachims; 2015a, 2015b) is somewhat similar to ours in the sense that they minimize a "corrected" loss function using Inverse Propensity Scores (IPS) corrected by a regularization term over the linear stochastic policy class. We go much further in our paper by minimizing our loss function over a much more general class of stochastic policies.

More broadly, our work is related to unsupervised auto encoder neural networks, counterfactual estimation and es-

timating individual treatment effects. The standard auto encoder neural networks aim to learn a representation of the data (possibly in a lower dimensional space) using encoder and reconstruction layers (Vincent et al. 2010; Bengio, Courville, and Vincent 2013; Kingma and Welling 2013) for dimensionality reduction. However, in our work we use auto encoders to remove the bias created by the logging policy. Our novel auto encoder contains both encoder and reconstruction layers but jointly minimizes both the mean-squared loss between the reconstruction and actual features and the distance between the different distributions of the various representation populations.

There are many works in estimating Individualized Treatment Effects (ITE) (Athey and Imbens 2015; Hill 2011). For example, one of the most recent one (Wager and Athey 2015) proposes a variant of random forests, called causal forests. Their approach to remove bias is somewhat different to ours in that they use tree-based algorithms on propensity scores and estimate treatment effect on each leaf. Another recent work on ITE estimation is (Alaa and van der Schaar 2017) which proposes multi task Gaussian processes to estimate treatment effects from observational data. However, Gaussian processes are known to have poor scaling, $\mathcal{O}(N^3)$, with the number of samples, which may result in this approach being computationally infeasible. (Shalit, Johansson, and Sontag 2017) recently extends (Johansson, Shalit, and Sontag 2016).

We can categorize the counterfactual estimation techniques into three groups: direct estimation, inverse propensity score (IPS) estimation and doubly robust estimation. The direct estimation approach computes the counterfactuals by learning a mapping from feature-treatment pair to their outcomes (Prentice 1976; Wager and Athey 2015). The IPS estimates learn the counterfactuals by re-weighting each instance with their inverse propensity scores (Swaminathan and Joachims 2015a; Joachims and Swaminathan 2016). Self normalizing estimator, a variant of IPS estimator, is also used to learn counterfactuals (Swaminathan and Joachims 2015b). The doubly robust estimation techniques combine the direct and IPS methods and generate more robust counterfactual estimates (Dudík, Langford, and Li 2011; Jiang and Li 2016). We believe our method lies within the realm of doubly robust techniques.

Our paper builds on (Atan et al. 2016) where the bias is addressed by feature selection step. This work extends their approach by simply mapping the features into a representation step by trading off between the information loss and the bias.

Problem Setup

Let $\mathcal{T}$ be the set of k treatments, $\mathcal{X}$ be the feature space and $\mathcal{Y}$ be the set of possible outcomes. For each instance $x \in \mathcal{X} \subset \mathbb{R}^d$, there is a treatment assignment $T \in \mathcal{T}$ and there are k *potential* outcomes: $Y^{(0)}, Y^{(1)}, \ldots, Y^{(k-1)} \in \mathcal{Y}$, corresponding to each of the k treatments. The *observed* outcome is denoted $Y \equiv Y^{(T)}$. We assume that the potential outcomes, $(Y^{(0)}, \ldots, Y^{(k-1)})$, are independent of the treatment assignment T given the feature vector x, that is, $(Y^{(0)}, \ldots, Y^{(k-1)}) \perp\!\!\!\perp T|x$.

A treatment policy, $\pi : \mathcal{X} \to \Delta(\mathcal{T})$, is a map taking feature vectors to probability distributions over the k possible treatments. Let $\Pr(\mathcal{X})$ be a probability distribution over $\mathcal{X}$. Our goal is to find a policy, π^* that maximizes the expectation of the outcome $Y^T|X = x$ when $x \sim \Pr(\mathcal{X})$ and $T \sim \pi$. Formally, let π^* maximize

$$V(\pi) = \mathbb{E}_{x\sim\Pr(\mathcal{X})}\left[\mathbb{E}_{T\sim\pi(x)}\left[\mathbb{E}\left[Y^{(T)}|x\right]\right]\right]$$

A simple argument shows that $\pi^*(x)$ assigns all probability mass to the treatment(s) with the greatest expected outcome, for each x. Now suppose π is a policy such that $\pi(x)$ assigns the treatment with the greatest expected outcome the highest probability mass (but it may not be the case that *all* probability mass is assigned to the 'best' treatments). Then we can recover an optimal π^* from π by simply sending the maximal probability to 1 and all other probabilities to 0, that is $\pi^* = h(\pi)$. Therefore, it suffices to find a policy that assigns higher probabilities to better outcomes, and we broaden our definition of an optimal policy to include all such policies, noting that by applying h to such a policy, we will in fact obtain a *truly* optimal policy.

In the dataset $\mathcal{D}^N = \{(x_1, t_1, y_1), \ldots, (x_N, t_N, y_N)\}$, we observe the patient's features, x_n, the treatment received, t_n, and the outcome of the treatment received, $y_n \equiv Y^{(t_n)}$ – importantly, we do not observe any of the other potential outcomes (i.e. $Y^{(t)}(x_n)$ for $t \neq t_n$). In the medical setup, the feature vector x could represent a patient's demographic information, family history, previous treatments, etc., t could represent the surgery, intervention or drug received by the patient, and y represents the effect of the treatment. We assume that the data is generated by the following process:

- A patient with feature vector x_n arrives according to the fixed but unknown distribution $\Pr(\mathcal{X})$, i.e., $x_n \sim \Pr(\mathcal{X})$.
- Given the feature vector x_n, a treatment t_n is decided upon by a *logging policy* $\pi_0(x_n)$. That is $t_n \sim \pi_0(x_n)$.
- The potential outcomes are drawn according to $y_n^t \sim \Pr(\cdot|x_n, t)$ for all t, and $y_n \equiv y_n^{t_n}$ is observed.

The difficulty in this problem arises because of (i) missing outcomes of the treatments that are not received by the patient – the counterfactuals, (ii) the logging policy π_0, which for example represents some pre-existing clinical guidelines. Even if π_0 did not depend on x, that is, $\pi_0(x) = \pi_0(x')$ for all $x, x' \in \mathcal{X}$, the problem of learning an optimal policy is not trivial as the data does not contain the counterfactual treatment outcomes. Our problem is further complicated by π_0 being dependent on x, that is, $\pi_0(x) \neq \pi_0(x')$ in general for $x \neq x'$. Then the distributions under different treatment regimens, $\Pr(\mathcal{X}|t)$, are no longer the same as the general distribution $\Pr(\mathcal{X})$, so the functions trained on the treatment regimens will not generalize well to the whole population.

It is worth noting at this point that it *may* be the case that π_0 varies with some unmeasured features; for example, it may be the case that poorer patients cannot afford a certain drug and so are rarely given it and that the wealth of a patient is not measured in the dataset. This is known as hidden *confounding* (Rosenbaum and Rubin 1983;

Pearl 2009) . We make the common assumption that we do, in fact, measure all possible influences on the logging policy π_0.

We emphasize here that the dependence of π_0 on x actually creates two nested classes of problems: (i) if π_0 does not vary with x then we are in a *randomized* setting, an example of which would be a clinical trial, (ii) if π_0 does vary with x then we are in an *observational* setting, an example of which is simply collecting records of patients that have seen a doctor and received a treatment (*based on the doctor's belief about the available treatments*). In this paper we tackle (ii) by first trying to make the problem as close to (i) as possible.

To help illustrate our approach, consider the following toy example: suppose a doctor assigns patients of a certain disease a treatment based on their age which might be *young*, *middle-aged* or *old*. Suppose, that the doctor assigns treatment 1 to all young patients and treatment 2 to all old patients, and to all middle-aged patients he assigns the treatment completely at random. Then certainly any dataset collected for this procedure will be biased (there are *no* young patients who receive treatment 2). However, if instead we classify patients as *young-or-old* and *middle-aged* (and there is an equal proportion of young and old patients) then now the treatment assignments to *these* categories are completely random, and so by performing this transformation on the features, we have removed the bias in the dataset. Note, however, that there is a cost to this; we have lost information that was held in the feature vector, x, that may have been useful for learning the optimal policy. In practice it will be very difficult to remove all the bias and at the very least very costly in terms of the information loss – so even after performing such a transformation we will still need to learn an optimal treatment policy from *incomplete* (and *slightly* biased) data that does not contain the counterfactual outcomes.

We build on the above example by breaking the problem down into two parts. We introduce a representation space, $\mathcal{R}$, and seek to find a map $\Phi : \mathcal{X} \to \mathcal{R}$ such that the induced distribution of the treatments over $\mathcal{R}$, which we denote $\Pr(T|\Phi)$, is less biased (as in the toy example above) but also such that a representation, $\Phi(x)$, retains as much information about the input feature, x, as possible. We then look to define an optimal policy $\hat{\pi} : \mathcal{R} \to \Delta(\mathcal{T})$ which leads to a policy $\bar{\pi} : \mathcal{X} \to \Delta(\mathcal{T})$ given by $\bar{\pi}(x) = \hat{\pi}(\Phi(x))$.

We measure the bias of a distribution, q, over $\mathcal{T}$ with respect to a target distribution, p, using the *cross-entropy loss* defined by:

$$\ell(p, q) = -\sum_{t \in \mathcal{T}} p(t) \log q(t)$$

noting that, for p fixed, the above loss is minimized when $q \equiv p$. In our case, the target distribution, p, is the distribution of treatments over the whole population (note that this distribution is induced by $\Pr(\mathcal{X})$ and π_0), which we write $\Pr(T)$, since then we will have that the distributions $\{\Pr(\mathcal{X}|t) : t \in \mathcal{T}\}$ are all equal to $\Pr(\mathcal{X})$. Therefore the point-wise loss we look to minimize over maps Φ is:

$$\ell_{ce}(\Phi; x) = \ell(\Pr(T), \Pr(T|\Phi(x))) \tag{1}$$

To measure the information loss due to a map $\Phi : \mathcal{X} \to \mathcal{R}$ we define the map $\hat{\Psi} : \mathcal{R} \to \mathcal{X}$ as the function that minimizes:

$$\ell_{inf}(\Phi, \Psi) = \int ||x - \Psi(\Phi(x))||_2^2 \, \mathrm{d}\Pr(\mathcal{X}) \tag{2}$$

where $||\cdot||_2$ denotes the usual ℓ_2-norm and then the information loss is defined as $\ell_{inf}(\Phi, \hat{\Psi})$. Note that if Φ is injective (which would correspond to no information being lost due to the map Φ), then $\hat{\Psi} = \Phi^{-1}$ and so $\ell_{inf}(\Phi, \hat{\Psi}) = 0$. Also note that since $\hat{\Psi}$ minimizes (2) we can bound (2) using *any* function $\Psi : \mathcal{R} \to \mathcal{X}$, and so we can minimize (2) over all function *pairs* (Φ, Ψ).

Therefore, our first goal is to find a pair (Φ, Ψ) that minimizes (1) and (2). We control the trade-off between these two (adversarial) losses using a hyperparameter, λ_1. The loss we look to minimize in the first stage is therefore:

$$\begin{aligned}&\mathcal{L}_1(\Phi, \Psi)\\ &= \int ||x - \Psi(\Phi(x))||_2^2 + \lambda_1 \ell_{ce}(\Phi; x) \, \mathrm{d}\Pr(\mathcal{X})\end{aligned} \tag{3}$$

Now, given a suitable representation map, Φ, we look to find an optimal policy, $\hat{\pi}$, on the *representation* space, $\mathcal{R}$. We emphasize again that this is not trivial, even after having performed the first stage since (i) not all bias will have been removed and (ii) the dataset is incomplete – the counterfactuals are still inaccessible. We define the target policy π^* by[1]:

$$\pi^*(t|r) = \frac{\mathbb{E}(Y^{(t)}|r)}{\sum_{s \in \mathcal{T}} \mathbb{E}(Y^{(s)}|r)} \tag{4}$$

noting that this policy assigns higher probabilities to treatments with greater expected outcomes, and is therefore optimal in the sense described at the beginning of this section.

We use the cross-entropy loss to measure the loss of a policy against π^* at a point $r \in \mathcal{R}$, i.e.

$$\ell_{pol}(\pi; r) = -\sum_{t \in \mathcal{T}} \pi^*(t|r) \log(\pi(t|r))$$

again noting that such a cross-entropy loss is minimized when $\pi(r) \equiv \pi^*(r)$. In fact, since the denominator in (3) is the same for each term in (4) so we can multiply ℓ_{pol} by the denominator. The total loss we therefore aim to minimize is

$$\mathcal{L}_2(\pi) = -\int \sum_{t \in \mathcal{T}} \mathbb{E}(Y^{(t)}|r) \log(\pi(t|r)) \, \mathrm{d}\Pr(\mathcal{R})$$

Algorithm

Having defined the problem, we now describe our algorithm, *Deep-Treat*. We model the representation-decoder pair, (Φ, Ψ), using our novel *bias-removing auto-encoder* and model the optimal policy (over $\mathcal{R}$) using our novel *feed-forward policy optimization network*.

[1]We abuse notation slightly – writing $\pi(t|r)$ to denote the probability of taking action t according to $\pi(r)$.

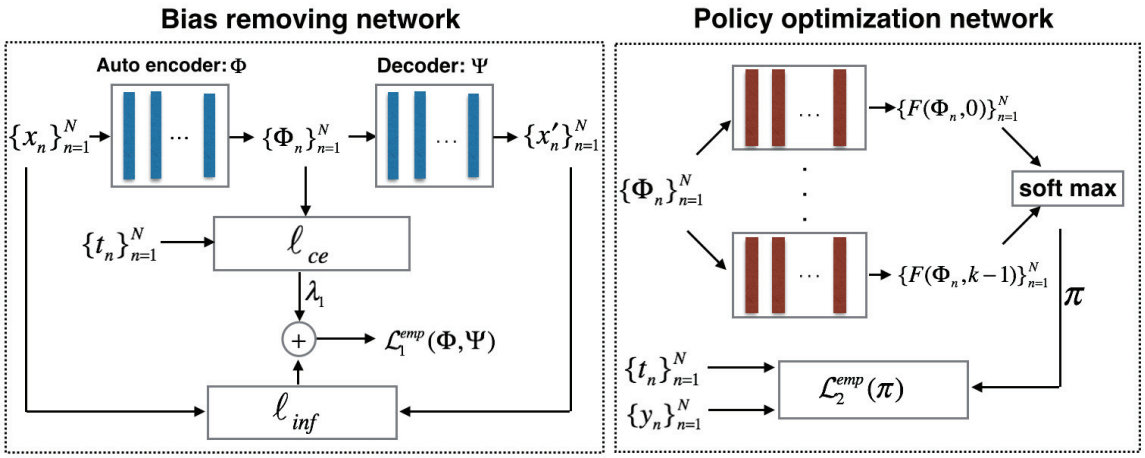

Figure 1: Neural network Model

Bias removing auto-encoder

We model the pair (Φ, Ψ) as the encoder and decoder blocks, respectively, of an auto-encoder network. Formally,

$$\begin{aligned}\Phi(x) &= W_e^{L_e-1}\sigma(W_e^{L_e-2}\dots\sigma(W_e^0 x)), x \in \mathcal{X} \\ \Psi(r) &= W_d^{L_d-1}\sigma(W_d^{L_d-2}\dots\sigma(W_d^0 r)), r \in \mathcal{R}\end{aligned}$$

where σ denotes an activation function (rectified linear unit in our case) and W_e^l and W_d^l are the weight matrices of the l^{th} layer of the encoder and decoder blocks, respectively. Let $\Theta_r = (W_e^0, \dots, W_e^{L_e-1}, W_d^0, \dots, W_d^{L_d-1})$ be the parameters of the auto-encoder. Note that L_e and L_d are the number of layers in the encoder and decoder blocks.

To train this network, we need to evaluate the empirical form of $\mathcal{L}_1$, which is given by

$$\mathcal{L}_1^{emp}(\Phi, \Psi) = \sum_{n=1}^{N} ||x_n - x'_n||_2^2 + \lambda_1 \ell_{ce}(\Pr(T), \Pr(T|\Phi_n))$$

where $\Phi_n = \Phi(x_n)$ and $x'_n = \Psi(\Phi(x_n))$. Unfortunately, this objective cannot be calculated directly because both $\{\Pr(T|r) : r \in \mathcal{R}\}$ and $\Pr(T)$ are not known by the learner initially. The Deep-Treat algorithm estimates $\Pr(T)$ and the functional form of $\Pr(T|r)$ from the dataset, and replaces the actual quantities with the estimated ones. The estimate of $\Pr(T)$ is given by

$$\widehat{\Pr}(T = t) = \frac{\sum_{n=1}^{N} \mathbb{I}(T_n = t)}{N}$$

where $\mathbb{I}(\cdot)$ is an indicator function. The estimate of $\Pr(T|r)$ can be learned using a standard parametrized supervised learning algorithm. We use the logistic regression algorithm which estimates $\Pr(T|r)$ by assuming the following form:

$$\widehat{\Pr}(T = t|r) = \frac{\exp(\theta_t \cdot r)}{\sum_{t \in \mathcal{T}} \exp(\theta_t \cdot r)}$$

where $\cdot$ denotes the dot product and $\{\theta_t\}_{t \in \mathcal{T}}$ are the parameters of the logistic regression. Note that $\Pr(T|r)$ depends on Φ and so we must learn new $\{\theta_t\}_{t \in \mathcal{T}}$ for each update of Φ. In this case, the cross-entropy loss between $\widehat{\Pr}(T)$ and $\widehat{\Pr}(T|\Phi_n)$ reduces to

$$\ell_{ce}(\widehat{\Pr}(T), \widehat{\Pr}(T|\Phi_n)) = - \sum_{t \in \mathcal{T}} \widehat{\Pr}(t) \left(\theta_t \cdot \Phi_n - \log \left(\sum_{t \in \mathcal{T}} \exp(\theta_t \cdot \Phi_n) \right) \right)$$

We train the bias-removing network by starting with a random Θ_r and update using the Adam optimizer (Kingma and Ba 2014). We particularly chose the Adam optimizer because it is computationally efficient and requires very little memory but any other optimization algorithm can also be used.

Policy Optimization Network

We model the optimal policy, $\hat{\pi}$, on the representation space using a softmax layer. Formally,

$$\hat{\pi}(t|r) = \frac{\exp(F(r,t))}{\sum_{t \in \mathcal{T}} \exp(F(r,t))}$$

where

$$F(r,t) = W_p^{t,L_p-1}\sigma(W_p^{t,L_p-2}\dots\sigma(W_p^{t,0} r))$$

and $\Theta_p = \left(\{W_p^{t,0} : t \in \mathcal{T}\}, \dots, \{W_p^{t,L_p-1} : t \in \mathcal{T}\} \right)$ be the parameters of the policy optimization network where $\{W_p^{t,l} : t \in \mathcal{T}\}$ are the weight matrices of the lth layer and L_p is the number of layers in policy optimization network.

To train this network we need to evaluate the empirical form of $\mathcal{L}_2$, given by

$$\mathcal{L}_2^{emp}(\pi) = - \sum_{n=1}^{N} \sum_{t \in \mathcal{T}} Y^{(t)}(x_n) \log(\pi(t|\Phi_n))$$

However, we cannot evaluate this for two reasons: (i) we don't have access to the counterfactual outcomes, (ii) we

Algorithm 1 Deep-Treat

Input: L_e, L_d, L_p, λ_1, λ_2
Set $\widehat{\Pr}(t) := \frac{1}{n}\sum_{n=1}^{N} \mathbb{I}(T_n = t)$ for all t.
// Bias Removing Network
for until convergence **do**
  Fit logistic regression for the data $\mathcal{S}^N = (\Phi_n, t_n)_{n=1}^N$.
  Set $\Theta_r \leftarrow \mathrm{Adam}(\mathcal{S}^N, \Theta_r)$
  Compute new representations Φ based on Θ_r
end for
Output : Representations $\{\Phi_n\}_{n=1}^N$
// Policy Optimization Network
Generate new dataset $\mathcal{V}^N = \{\Phi_n, t_n, y_n\}$
for until convergence **do**
  Set $\Theta_p \leftarrow \mathrm{Adam}(\mathcal{V}^N, \Theta_p)$
end for
Output of training phase : π^*
// Execution Phase
Compute the distribution over treatments $\pi^*(\cdot|x)$
Draw a treatment $\hat{t}(x) = \max_{t\in\mathcal{T}} \pi^*(t|x)$
Output of execution phase: Recommendation $\hat{t}(x)$

need to correct the bias of the representations by weighting the instances by their inverse propensities (Rosenbaum and Rubin 1983) since not all bias will be removed during the auto-encoder stage. We therefore use the following modified cross-entropy loss:

$$\widehat{\mathcal{L}}_2^{emp}(\pi) = -\sum_{n=1}^{N} \frac{y_n \log(\pi(t|\Phi_n))}{\widehat{\Pr}(t_n|\Phi_n)} \tag{5}$$

The corrected cross entropy loss in (5) is more robust than the IPS estimate for policy loss (Swaminathan and Joachims 2015a) which has an exponential term that could explode the gradients.

We introduce a regularization term, $\mathrm{R}(\pi)$, to avoid overfitting and train our neural network to minimize the regularized loss using the Adam optimizer:

$$\hat{\pi} = \arg\min_{\pi} \left(\widehat{\mathcal{L}}_2^{emp}(\pi) + \lambda_2 \mathrm{R}(\pi) \right)$$

where $\lambda_2 > 0$ is the hyperparameter used to trade-off between the loss and regularization.

Once these functions have been learned, we set the final policy to $\pi(x) = h(\hat{\pi}(\Phi(x)))$.

Extension: Treatment Effect Estimation

In this case, we consider a binary treatment set $\mathcal{T} = \{0, 1\}$ where 1 is often known as *treated* and 0 is the *control*. The quantity of interest in this case is the Individualized Treatment Effect (ITE) for feature vector x, which is defined as the difference between the expected treated outcome and expected control outcome, that is,

$$\mathrm{ITE}(x) = \mathbb{E}\left[Y^{(1)} - Y^{(0)} | x \right].$$

We break this problem into two parts as well. In the first part, we use the bias removing auto-encoder to find a map $\Phi : \mathcal{X} \to \mathcal{R}$ and use the representations to estimate $\mathrm{ITE}(x) = \widehat{\mathrm{ITE}}(\Phi(x))$. We model $\widehat{\mathrm{ITE}}(\cdot)$ on the representation space as

$$\widehat{\mathrm{ITE}}(r) = G(r, 1) - G(r, 0)$$

where $G(r, t) = W_i^{t,L_i-1}\sigma(W_i^{t,L_i-2} \ldots \sigma(W_i^{t,0} r))$, and Θ_i be the parameters of the ITE estimation network. To train this neural network, we need to evaluate the empirical Mean-Squared Error (mse), given by

$$\mathcal{L}_3^{emp}(\widehat{\mathrm{ITE}}) = \sum_{n=1}^{N} \left[\left(G(\Phi_n, 1) - y_n^1 \right)^2 + \left(G(\Phi_n, 0) - y_n^0 \right)^2 \right].$$

However, we cannot compute this loss since (i) the counterfactual treatment outcomes are not given to us, (ii) there is bias that is not removed by the bias-removing auto encoder. We therefore train the neural network using the modified MSE:

$$\widehat{\mathcal{L}}_3^{emp}(\widehat{\mathrm{ITE}}) = \sum_{n=1}^{N} \frac{\left(G(\Phi_n, t_n) - y_n \right)^2}{\widehat{\Pr}(t_n|\Phi_n)}$$

Note that $\widehat{\mathcal{L}}_3^{emp}(\widehat{\mathrm{ITE}})$ is an unbiased estimator of $\mathcal{L}_3^{emp}(\widehat{\mathrm{ITE}})$ if $\widehat{\Pr}(T|\Phi)$ is correctly estimated.

Experiments on IHDP dataset for ITE estimation

In this section, we describe the performance of our algorithm in the IHDP dataset. Note that it is difficult (perhaps impossible) to validate and test the algorithm using actual observational data unless the counterfactual treatment outcomes for each instance are available – which would (almost) never be the case. One way to validate and test our algorithm is by using simulated outcomes. This is the route we follow in these experiments.

Dataset description

The IHDP is intended to enhance the cognitive and health status of low birth weight, premature infants through pediatric follow-ups and parent support groups (Hill 2011). The semi-simulated dataset in (Hill 2011; Johansson, Shalit, and Sontag 2016) is based on covariates from a real randomized experiment that evaluated the impact of the IHDP on the subjects' IQ scores at the age of three: selection bias is introduced by removing a subset of the treated population. All outcomes (response surfaces) are simulated. We used the standard non-linear "Response Surface B" setting in (Hill 2011). The dataset comprises 747 subjects (608 control and 139 treated), and there are 25 covariates associated with each subject.

We implement our algorithm with $L_e = L_d = 2$ layers with 50 hidden units and 2 layers for potential outcome estimation for both treated and control groups with 50 hidden units with dropout 0.3. Table 3 shows the comparison of our algorithm with the benchmarks on the IHDP data (averaged over 100 iterations). In each of these iterations,

new outcomes are drawn from the data-generating model in (Hill, 2011). Within each iteration we randomly divide the IHDP dataset into training set (70%) and testing set (30%). This is a very standard approach in the literature (Hill 2011; Johansson, Shalit, and Sontag 2016).

Benchmarks

We compare our algorithm with following benchmarks:

- **BART** is a non-parametric tree-based method (Hill 2011).
- **Causal Forest** is a random forest variant for causal inference (Wager and Athey 2015).
- **kNN** estimates ITE by using k nearest neighbors.
- **Balancing Neural Networks (BNN)** is a feedforward network for ITE estimation (Johansson, Shalit, and Sontag 2016).

Performance Metrics

We evaluate all the algorithms in terms of the Precision in Estimation of Heterogeneous Effect (PEHE), that is,

$$\text{PEHE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\widehat{\text{ITE}}(\Phi(x_n)) - \text{ITE}(x_n)\right)^2}$$

Obtaining a good (small) PEHE requires accurate estimations of both observed and counterfactual outcomes.

Results

Table 1 shows the PEHE results averaged over 100 iterations. As seen from the Table 1, our algorithm significantly outperforms the benchmarks k-NN, BART, causal forest and BNN on IHDP data. Note that BNN is a strong benchmark as it handles the bias in the data by learning *balanced representations*. The performance gain of the Deep-Treat with respect to BNN shows that using auto-encoders to handle the selection bias is advantageous. Furthermore, extending the BNN to multiple actions is challenging and possibly requires new innovations. Our algorithm on the other hand is able to run on any number of actions, numerical results for this are given in the next section.

Algorithm	PEHE
Deep-Treat	1.83 ± 0.07
k-NN	5.30 ± 0.30
Causal Forest	3.86 ± 0.20
BNN	2.20 ± 0.13
BART	3.50 ± 0.20

Table 1: Performance in the IHDP experiment

Experiments on Breast Cancer Dataset for Policy Optimization

In this section, we describe the performance of our algorithm in the breast cancer dataset. Another way to validate and test our algorithm is by using an alternative accepted procedure to infer counterfactuals and to test the prediction of our algorithm against this alternative accepted procedure. This is the route we follow in our experiments.

Dataset description

We apply Deep-Treat to the problem of recommending chemotherapy regimens for breast cancer patients. We evaluate our algorithm on a dataset of 10,000 records of breast cancer patients participating in the National Surgical Adjuvant Breast and Bowel Project (NSABP) by (Yoon, Davtyan, and van der Schaar 2016). Each instance consists of the following information about the patient: age, menopausal, race, estrogen receptor, progesterone receptor, human epidermal growth factor receptor 2 (HER2NEU), tumor stage, tumor grade, Positive Axillary Lymph Node Count(PLNC), WHO score, surgery type, Prior Chemotherapy, prior radiotherapy and histology. The treatment is a choice among six chemotherapy regimes AC, ACT, AT, CAF, CEF, CMF. The outcomes for these regimens were derived based on 32 references from PubMed Clinical Queries. The 32 references are found by performing a narrow search on Pubmed about the chemotherapy regimens of the interest. The treatment outcomes for each patient are derived by aggregating the actual 5-year survival rates of the references by taking their population statistics into account. Hence, the data contains the feature vector x and all derived outcomes for each treatment $\{Y^t\}_{t\in\mathcal{T}}$. The details are given in (Yoon, Davtyan, and van der Schaar 2016).

We generate an artificially biased $\mathcal{D}^N = \{(x_1, t_1, y_1), \ldots, (x_N, t_N, y_N)\}$ from the original data. In order to do that, we randomly select and sequester 20% of the data, which we use to fit a logistic regression from features to outcomes. Denote by $\theta_{0,t}$ the parameter of the logistic regression for treatment t. We use the fitted logistic regression model to generate artificially biased data in the following way. For each instance n, we sample a random treatment $t_n \sim \pi_0(\cdot|x_n) \propto \exp(\kappa\theta_{0,t} \cdot x_n)$ and set the observed outcome to be $y_n \equiv Y_n^{t_n}$. We use $\kappa > 0$ to increase the randomization in the data – a smaller κ makes the data more random. By following this approach, we create a dataset $\mathcal{D}^N = \{(x_1, t_1, y_1), \ldots, (x_N, t_N, y_N)\}$. This is a standard way of creating artificially biased data; see (Swaminathan and Joachims 2015a). We divide the other 80% of the data into 3 parts to train, validate and test the performance of the algorithms.

We evaluate our algorithm with $L_e = 1, L_d = 1$ auto encoder layers (with 25 hidden units in the representation block) and $L_p = 2$ policy layers with 50 hidden units. We use relu as the activation function. We validate the hyper parameters of our algorithm on $\lambda_1^* \in \left[10^{-3}, 10^{-2}, \ldots, 10^{2}\right]$. We choose the hyperparameter that minimizes the policy loss in the validation instances. We use the same technique to validate the hyper parameters of each algorithm. We train each algorithm with the Adam optimizer and with early stopping criteria.

Benchmarks

We compare the accuracy of our algorithm with the following benchmarks:

Algorithm/Metric	Accuracy	Improvement score	Kappa coefficient	Improvement score
Deep-Treat	$\mathbf{79.05\% \pm 0.65\%}$	-	70.96%	-
PONET	$72.87\% \pm 1.54\%$	22.77%	62.40%	13.72%
POEM	$64.68\% \pm 1.88\%$	40.68%	51.05%	39.00%
MLP	$62.74\% \pm 0.66\%$	43.77%	48.36%	46.73%
LR	$62.28\% \pm 0.88\%$	44.45%	47.72%	48.70%
Logging	$27.84\% + 0.08\%$	70.96%	0%	-

Table 2: Performance in the Breast Cancer Experiment ($\kappa = 0.25$)

- **Policy Optimization with Neural Networks (PONET)** is our algorithm without the bias removing network.
- **POEM** is the algorithm from (Swaminathan and Joachims 2015a) where the objective is optimized with Adam.
- **Multilayer Perceptron (MLP)** is the MLP algorithm on the concatenated input (x, t) .
- **Logistic Regression (LR)** is the separate LR algorithm on input x on each treatment t.
- **Logging** is the logging policy performance.

For MLP and LR, we compute a policy based on the estimated outcomes of each treatment. Denote by $G(x, t)$ the estimated outcome for treatment t on a patient with feature vector x. The recommended treatment is then $\hat{t}(x) = \arg\max_{t\in\mathcal{T}} G(x, t)$. Note that MLP, LR and PONET algorithms do not address the selection bias.

Performance Metrics

In this subsection, we introduce the metrics that are used to compare the performance of each algorithm. For the n^{th} instance, let t_n^* denote the treatment with highest outcome, i.e., $t_n^* = \arg\max_{t\in\mathcal{T}} Y_n^t$. Let $\mathcal{D}_{test}$ denote the instances in the testing set and $N_{test} = |\mathcal{D}_{test}|$. We define the (absolute) accuracy of an algorithm A that returns a policy π_A as

$$\text{Acc}(A) = \frac{1}{N_{test}} \sum_{j\in\mathcal{J}_{test}} \sum_{t\in\mathcal{T}} \pi_A(t|x_j)\mathbb{I}(t_n^* = t).$$

We run each algorithm, A, for 10 iterations on the breast cancer dataset. In each iteration, we generate artificially biased data by following the above procedure, and train, validate and test the performance of each algorithm on randomly divided datasets. We report the average of the iterations with 95% confidence intervals.

We define loss with respect to the "perfect" algorithm that would predict accurately all of the time, so the *loss* of the algorithm A is $1 - \text{Acc}(A)$. We evaluate the improvement of our algorithm over each other algorithms as the ratio of the actual loss reduction over the possible loss reduction, expressed as a percentage:

$$\text{Improvement Score}(A) = \frac{\text{Acc}(\textit{Deep-Treat}) - \text{Acc}(A)}{1 - \text{Acc}(A)}$$

The Improvement Score of each algorithm A with respect to our algorithm is presented in Table 1. We also present Cohen's Kappa Coefficient (alongside an improvement score for this) which measures the improvement of the algorithms over the logging policy, given by:

$$\text{Kappa}(A) = \frac{\text{Acc}(A) - \text{Acc}(\text{Logging})}{1 - \text{Acc}(\text{Logging})}$$

Results

Performance Comparisons Table 2 shows the performance on each algorithm. As seen, our algorithm is able to outperform each benchmark in terms of the Acc and Kappa metrics. The performance gain of Deep-Treat with respect to PONET and MLP shows that naive policy optimization algorithms tend to fit a function that does not generalize well on the counterfactuals. Our algorithm is also performing better than the POEM algorithm by allowing a more generalized class of policies (POEM optimizes over linear stochastic policies). In comparison with POEM, Deep-Treat finds the best chemotherapy regime for an additional 11% of the 10,000 patients. This means that, in the sample of 10,000 women with breast cancer, 1100 additional women would receive the best chemotherapy regime, and hence have the best chance of survival.

Effect of the logging policy In this subsection, we evaluate Deep-Treat on different logging policies. We generate different logging policies by varying the parameter κ on our data-generating process. Table 3 shows the entropy of $\widehat{\Pr}(T)$ on the generated data. Larger entropy indicates more randomized data. As seen from the table, smaller κ generates more randomized datasets and larger κ generates more biased datasets.

κ	2^{-2}	2^{-1}	2^0	2^1	2^2
Entropy	1.71	1.49	1.13	1.02	0.9545

Table 3: Entropy of $\widehat{\Pr}(T)$

Figure 2 presents the accuracy of Deep-Treat and logging policy based on different κ values. The accuracy of the Deep-Treat increases with accuracy of the logging policy if there is enough randomization in the data. This is because Deep-Treat improves the logging policy performance by replacing the suboptimal treatments of the logging policy with the optimal ones. In order for Deep-Treat to do that there should be enough randomization in the data so that Deep-Treat can improve upon observing outcomes of the alternative treatments received by similar patients. If there is no such randomization, the accuracy of the Deep-Treat decreases.

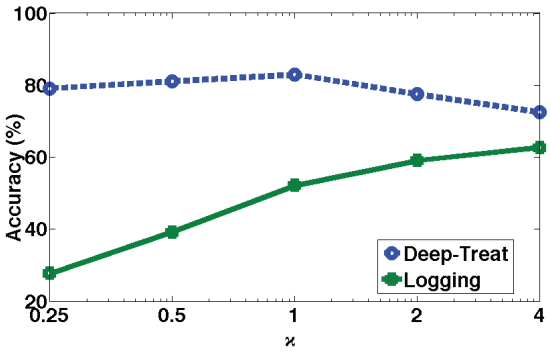

Figure 2: Effect of the logging policy

Conclusion

In this paper we introduced a new approach for learning balanced representations and optimal policies from observational data. The heart of our method is the bias removing auto-encoder which trades off between information loss and bias and the policy optimization feedforward neural network which learns a mapping from a representation space to a probablity distribution over treatments. Our algorithm is widely applicable in domains beyond medicine due to the wealth of observational data that exists in a wide variety of areas, and we leave it to a future work to explore these possibilities. Although our algorithm is able to achieve significant empirical gains with respect to strong benchmarks, we hope to see development of a deeper theoretical understanding of learning balanced representations in the multiple treatment setup, believing this is an interesting open problem.

Acknowledgement

The authors would like to thank the reviewers for their helpful comments. The research presented in this paper was supported by NSF 1407712, NSF 1533983 and Mathematical Data Science (MDS).

References

Alaa, A. M., and van der Schaar, M. 2017. Bayesian inference of individualized treatment effects using multi-task gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*.

Atan, O.; Zame, W. R.; Feng, Q.; and van der Schaar, M. 2016. Constructing effective personalized policies using counterfactual inference from biased data sets with many features. *arXiv preprint arXiv:1612.08082*.

Athey, S., and Imbens, G. W. 2015. Recursive partitioning for heterogeneous causal effects. *arXiv preprint arXiv:1504.01132*.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Dudík, M.; Langford, J.; and Li, L. 2011. Doubly robust policy evaluation and learning. In *International Conference on Machine Learning (ICML)*.

Hill, J. L. 2011. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics* 20(1).

Jiang, N., and Li, L. 2016. Doubly robust off-policy evaluation for reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Joachims, T., and Swaminathan, A. 2016. Counterfactual evaluation and learning for search, recommendation and ad placement. In *International ACM SIGIR conference on Research and Development in Information Retrieval*, 1199–1201.

Johansson, F.; Shalit, U.; and Sontag, D. 2016. Learning representations for counterfactual inference. In *International Conference on Machine Learning (ICML)*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Pearl, J. 2009. *Causality*. Cambridge university press.

Prentice, R. 1976. Use of the logistic model in retrospective studies. *Biometrics* 599–606.

Rosenbaum, P. R., and Rubin, D. B. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70(1):41–55.

Shalit, U.; Johansson, F.; and Sontag, D. 2017. Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning (ICML)*.

Swaminathan, A., and Joachims, T. 2015a. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16:1731–1755.

Swaminathan, A., and Joachims, T. 2015b. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems*, 3231–3239.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

Wager, S., and Athey, S. 2015. Estimation and inference of heterogeneous treatment effects using random forests. *arXiv preprint arXiv:1510.04342*.

Yoon, J.; Davtyan, C.; and van der Schaar, M. 2016. Discovery and clinical decision support for personalized healthcare. *IEEE journal of biomedical and health informatics*.