ESTIMATING HETEROGENEOUS TREATMENT EFFECTS USING NEURAL NETWORKS WITH THE Y-LEARNER

Anonymous authors
Paper under double-blind review

ABSTRACT

We develop the Y-learner for estimating heterogeneous treatment effects in experimental and observational studies. The Y-learner is designed to leverage the abilities of neural networks to optimize multiple objectives and continually update, which allows for better pooling of underlying feature information between treatment and control groups. We evaluate the Y-learner on three test problems: (1) A set of six simulated data benchmarks from the literature. (2) A real-world large-scale experiment on voter persuasion. (3) A task from the literature that estimates artificially generated treatment effects on MNIST digits. The Y-learner achieves state of the art results on two of the three tasks. On the MNIST task, it gets the second best results.

1 OPENING REMARKS

We consider the problem of estimating the Conditional Average Treatment Effect (CATE) in randomized experiments and observational studies. The CATE is a desirable quantity to estimate, because it allows us to measure how well a given treatment works for an individual conditioned on their observed covariates. Thus, the CATE allows us to better understand the underlying causal mechanisms at play and better personalize treatments at an individual level. Because of its promise, CATE estimation appears across a wide range of disciplines including political science, medicine, economics, and digital experiments (Henderson et al., 2016; Powers et al., 2018; Laddy et al., 2016; Athey & Imbens, 2016; Green & Kern, 2012; Tian et al., 2014; Johansson et al., 2016; Louizos et al., 2017).

CATE estimation has been an especially active area of research in the past year. In Künzel et al. (2017), the authors develop the X-learner and the U-learner. Both of these methods are so-called “meta-learners,” CATE estimation strategies that can be carried out with any sufficiently generic function approximator as a base learner. The authors primarily consider Random Forests and Bayesian Additive Regression Trees (BART) as base learners. In doing so, they are able to provide several convergence guarantees that relate the size of the treatment and control groups to the efficiency of the estimation strategy. Meanwhile, in Nie & Wager (2017) the R-learner is introduced. The authors show that the R-learner delivers excellent performance on extant benchmarks, especially when it is parameterized by deep neural networks. The paper also provides a “quasi-oracle” regret bound for non-parametric regression problems, which they apply to the R-learner.

Motivated by these recent advances, we seek to answer the question: is there a more efficient neural network architecture for CATE estimation? Recent work has been constrained, both by its desire to incorporate formal guarantees and by its desire to work with any general function approximator. While these are worthwhile goals, we are curious how much performance can be improved by designing a CATE estimation strategy that takes advantage of the unique properties of neural networks. In particular, deep neural networks can be continually optimized. This stands in contrast to other estimators like RF and BART, which can not be meaningfully updated once trained. While this distinction may seem small, it crucially allows a single neural networks to be asynchronously optimized with respect to several distinct objectives. It also allows multiple networks to “co-learn,” continually training on small amounts of data and staying in step with one another. Ultimately, we show how one can leverage these properties of neural networks to create a learner that achieves state of the art performance with only a fraction of the data on several CATE estimation tasks. We call our new learner the Y-learner. Code for our experiments will be released at publication and is available to reviewers upon request.
2 CATE Estimation Background and Related Work

2.1 Background and Assumptions

Consider a randomized experiment. In this experiment, there is a population $\mathcal{P}$. Each member of the population shares a common feature space $\mathcal{X}$. Denote by $X_i \in \mathbb{R}^d$ the features for population member $i$. We are interested in how each of these population members responds to some treatment. Let $W_i \in \{0, 1\}$ be 0 if $X_i$ is in the control group (does not receive treatment) and 1 if $X_i$ does receive treatment. Further, let $Y_i(1) \in \mathbb{R}$ be the response of member $i$ when receiving treatment and $Y_i(0) \in \mathbb{R}$ be the response of member $i$ when not receiving treatment. Within the causal inference literature, $Y_i(0)$ and $Y_i(1)$ are called potential outcomes and the above framework is called the potential outcomes framework (Rubin [1974]).

Let us consider a concrete example that is a favorite in introduction to economics courses. We would like to measure the impact of going to college on future income. It is our intuition that individuals who go to college should earn more. To verify this, we can measure the average treatment effect \\
ATE := $E[Y(1) - Y(0)].$

While the ATE is a useful diagnostic, it only tells us the impact of treatment over an entire population. It can tell us that, on average, going to college will improve future earnings. It cannot recommend whether individual $i$ should go to college based on his profile $X_i$. As an additional problem, the ATE is also susceptible to treatment and control group selection bias. On average, people with greater academic skills tend to go to college. But then who’s to say whether their improved income is because of their college education or because they were simply more skilled to begin with? To offer more personalized recommendations, we need to consider the Conditional Average Treatment Effect (CATE), defined by \\
$$CATE := E[Y(1) - Y(0)|X = x] := \tau(x)$$

Unfortunately, Equation (1) is difficult to estimate. For a given individual $X_i$, it is not possible observe both the outcome under treatment $Y(1)$ and the outcome under control $Y(0)$. You cannot clone a college-bound individual and force the clone to skip college instead just so you can measure both outcomes. While the situation may seem grim, if we are willing to make two strong assumptions then we can make progress on estimating the CATE. The first assumption, called Ignorability, addresses the selection bias issue we discussed above (Rosenbaum & Rubin [1983]). It prevents the existence of a random variable that influences the probability of treatment and the potential outcomes. The second assumption, called Overlap, ensures that no part of $X_i$ lets you uniquely identify whether individual $i$ will be assigned to treatment or control (D’Amour et al., 2017). For example, it prevents a situation wherein every individual under the age of 18 is automatically in the control group. These assumptions are strong, but nevertheless standard. They are true by design in randomized experiments (D’Amour et al. 2017; Künzel et al. 2017; Nie & Wager, 2017).

Assumption 1 (The treatment assignment $W_i$ is unconfounded) \\
$$(Y_i(1), Y_i(0)) \perp W|X.$$ 

Assumption 2 (Overlap) Then there exists constant $0 < e_{\min}, e_{\max} < 1$ such that for all $x \in Support(X)$,

$$0 < e_{\min} < e(x) < e_{\max} < 1.$$ 

Where $e(x)$, the propensity score of $x$ is defined by \\
$$e(x) := P(W = 1|X = x).$$

These two assumptions, plus regularity conditions, allow one to identify the CATE. We can estimate the CATE by proceeding as follows. Define \\
$$\mu_0(x) = E[Y(0)|X = x]$$
$$\mu_1(x) = E[Y(1)|X = x],$$

where $\mu_1$ is the treatment response function. It denotes the outcomes of the units who received treatment. $\mu_0$ is defined analogously for control units. To estimate $\tau(x)$ (the CATE), we compute estimates $\hat{\mu}_0, \hat{\mu}_1$ for $\mu_1$ and $\mu_0$ and then subtract to get \\
$$\hat{\tau}(x) = \hat{\mu}_1(x) - \hat{\mu}_0(x)$$

Below, we will discuss four common strategies for estimating $\hat{\tau}$.
**T-LEARNER**

In the T-Learner, we estimate $\hat{\mu}_0$ and $\hat{\mu}_1$ directly with any arbitrary function approximator. Let $f_i$ be such a function approximator. Then

$$\hat{\mu}_0(x) = f_0(x)$$
$$\hat{\mu}_1(x) = f_1(x)$$

We then estimate the CATE by taking differences:

$$\hat{\tau}(x) = \hat{\mu}_1(x) - \hat{\mu}_0(x)$$  \hspace{1cm} (3)

Under strong assumptions, it is possible to provide convergence rate guarantees for the T-learner (Künzel et al., 2017; Nie & Wager, 2017). In spite of its simplicity, the T-learner is almost always insufficient because it does not share information across the treatment and control outcome estimators (Athey & Imbens, 2015).

**S-LEARNER**

The S-learner tries to be more efficient than the T-learner by sharing information across the treatment and control estimators. It uses only a single function approximator, $f_i$. In addition to the input features $x_i$, this function approximator also receives the binary treatment indicator $w$.

$$\hat{\mu}_0(x) = f(x, w = 0)$$
$$\hat{\mu}_1(x) = f(x, w = 1)$$

The CATE is then estimated as before.

**U-LEARNER/R-LEARNER**

We first consider the U-Learner. Let $M$ be the main treatment effect function defined by $M(x) = E[Y|X = x]$ and $e$ be the treatment propensity given by $P(W = 1|X = x)$. The U-learner weights the estimated treatment effect by an estimated treatment propensity to form the CATE estimator. More concretely,

$$\hat{u}_\text{obs} = f_\text{obs}(Y_{\text{obs}} \sim X)$$
$$\hat{e} = f_e(W \sim X)$$
$$R_i = (Y_i - \hat{u}_\text{obs}(X_i))/(W_i - \hat{e}(X_i))$$
$$f_\tau = f_\tau(R \sim X)$$

The R-Learner learner was proposed in Nie & Wager (2017). It is an extension to the U-learner that provides some regularization and breaks estimation into a two step process. The authors prove the R-learner has several nice convergence guarantees. They also demonstrate that it achieves state of the art performance on several problems. See Nie & Wager (2017) for more details on the R-Learner.

**X-LEARNER**

This procedure makes use of imputed treatment effects to transfer information between treatment and control. Define $\hat{\mu}_0$ and $\hat{\mu}_1$ as in the T-learner. For each of these estimates, we can produce a corresponding imputed treatment effect

$$\hat{D}_{i,1} = Y_i(1) - \hat{\mu}_0(X_{i,1})$$
$$\hat{D}_{i,0} = \hat{\mu}_1(X_{i,0}) - Y_{i,0}$$

Note that this learner does in fact use the control estimator $\hat{\mu}_0$ on the treatment data $X_1$ and similarly for $\hat{\mu}_1$ and $X_0$. This is the correct way to impute the treatment effect estimate from $\hat{\mu}_0$ and $\hat{\mu}_1$. From here, we can get to the CATE by estimating the imputed treatment effects and then summing the estimates

$$\hat{\tau}_1 = f_1(\hat{D}_1 \sim X_1)$$
$$\hat{\tau}_0 = f_0(\hat{D}_0 \sim X_0)$$

For a theoretically grounded justification of this procedure, including convergence rate analysis, see Künzel et al. (2017).
2.2 Related Work

In addition to the work discussed above (Nie & Wager, 2017; Künzel et al., 2017; Athey & Imbens, 2015), there exists a variety of interesting work. We are particularly interested in work that develops better CATE estimation strategies, and in work about estimating causal effects with neural networks. In (Ramachandra, 2018), the author shows how to use autoencoders for generalized neighborhood matching, which one method of generating contractuals for estimating individual and average treatment effects. (Magliacane et al., 2017) is concerned with domain adaptation using causal inference to handle shifts caused by measurements taken in different contexts. Meanwhile, in (Johansson et al., 2016), representation learning via neural networks and domain adaptation are used to answer problems from counterfactual inference. (Louizos et al., 2017) considers the use of deep latent variable models to handle confounders. In (Alaa et al., 2017), parallels are drawn between causal inference and multi-task learning. Finally, in (Shalit et al., 2017) the authors develop an Integral Probability Metric based algorithm for measuring the ITE. We are eager to hear about more related work in this area, so please let us know if we have missed anything.

3 The Y-Learner

Our development of the Y-learner started by examining a deficiency in the X-learner. Recall that the X-learner is a two step procedure. In the first stage, the outcome functions, \( \hat{\mu}_0 \) and \( \hat{\mu}_1 \), are estimated and the individual treatment effects are imputed:

\[
D_1^i := Y(1) - \hat{\mu}_0(X_i) \quad \text{and} \quad D_0^i := \hat{\mu}_1(X_i) - Y_i(0).
\]

In the second stage, estimators for the CATE are derived by regressing the features \( X \) on the imputed treatment effects.

\[
\hat{\tau}_1 = f_{\tau_1}(\hat{D}_1 \sim X_1) \quad \text{and} \quad \hat{\tau}_0 = f_{\tau_0}(\hat{D}_0 \sim X_0).
\]

In the X-learner paper, random forests were used to obtain the estimates \( \hat{\mu}_0 \) and \( \hat{\mu}_1 \). However, suppose we used neural networks instead. In fact, suppose \( f_{\theta_0} \) and \( f_{\theta_1} \) estimate \( \mu_0 \) and \( \mu_1 \). Then we can write

\[
D_1^i := Y(1) - f_{\theta_0}(X_i) \quad \text{and} \quad D_0^i := f_{\theta_1}(X_i) - Y_i(0).
\]

Suppose we also want to use neural networks for the second stage. Then we can write

\[
\hat{\tau}_1 = f_{\tau_1}([Y(1) - f_{\theta_0}(X_i)] \sim X_1) \quad \text{and} \quad \hat{\tau}_0 = f_{\tau_0}([f_{\theta_1} - Y_i(0)] \sim X_0).
\]
When written in this way, it is clear that we should at least try to jointly optimize $f_r$ and $f_\theta$. That is, when we are optimizing $f_{\tau_1}$, we should also backprop through the network $f_{\theta_0}$ and similarly for $f_{\tau_0}$ and $f_{\theta_1}$. If we were using random forests, capturing this dependence would not be possible since random forests are largely fixed once trained. However, with neural networks this presents no problem. Neural networks also allow us to do some additional housekeeping. For instance, we only need to keep a single neural network to output the imputed treatment effects under this joint optimization strategy. Further, a two-stage estimation procedure is no longer necessary. We can simply train the imputation networks and the CATE estimation networks concurrently on the same data. The algorithm is presented as Algorithm 1 and also as a diagram in Figure 1.

**Algorithm 1** Y-Learner Pseudo Code

```plaintext
1: if $W_i == 0$ then
2: Update the network $f_{\theta_0}$ to predict $Y_i^{obs}$
3: Update the network $f_{\theta_1}$ to predict $Y_i^{obs} + f_r(X_i)$
4: Update the network $f_r$ to predict $f_{\theta_1}(X_i) - Y_i^{obs}$
5: end if
6: if $W_i == 1$ then
7: Update the network $f_{\theta_0}$ to predict $Y_i^{obs} - f_r(X_i)$
8: Update the network $f_{\theta_1}$ to predict $Y_i^{obs}$
9: Update the network $f_r$ to predict $Y_i^{obs} - f_{\theta_1}(X_i)$
10: end if
```

This process describes training the Y-Learner for one step given a data point $(Y_i^{obs}, X_i, W_i)$.

**Co-learning Networks and the Y-Learner**

While testing the Y-learner, we made an curious discovery. We noticed that it almost always obtained much better performance than the X-learner. This was not surprising, because we figured the joint optimization strategy of backpropagating through $f_{\theta_0}$ and $f_{\theta_1}$ in lines 4 and 9 of Algorithm 1 would allow those estimators to more directly benefit the final CATE estimation network $f_r$. However, if we stopped backpropagating through $f_r$, we saw there was no major loss in performance. The Y-learner still outperformed the X-learner by a large amount.

This seemed strange to us, since the Y-learner is structurally quite similar to the X-learner. One key difference between the two is that the Y-learner updates $f_{\theta_0}$, $f_{\theta_1}$, and $f_r$ continuously and in-step with one another, whereas in the X-learner the imputation networks $D_0$ and $D_1$ are fixed before training the CATE estimation networks $f_{\tau_0}$ and $f_{\tau_1}$. We hypothesized that perhaps this continual ‘co-learning’ process may help improve training. In other problems, such as generative adversarial networks, it is well known that the learning rate for co-learning networks is important. If one network learns too fast or too slow, it will make the other network unstable [Goodfellow et al., 2014]. In certain imitation learning algorithms, there is a more direct analogy. In these algorithms, one co-learns two networks: One critic network to tell the agent what to do and another action network to actually do it. Suppose this algorithm is run to completion. Subsequently we use the fully trained critic network to train a new action network from scratch. This seems like it should work, but it will usually fail [Ho & Ermon, 2016].

To test the effect of co-learning on the Y-learner, we ran the following experiment. On one of the simulated datasets from Section 4, we ran 4 learners. First, the standard X-learner with neural networks. Second, the Y-learner with full backpropagation through $f_{\theta_0}$, $f_{\theta_1}$ when training $f_r$. This is labeled ‘Y’. Third, the Y-learner with no backpropagation through $f_{\theta_0}$, $f_{\theta_1}$ when training $f_r$. This is labeled ‘Y no backprop.’ For the final learner, we train a Y-learner to completion. We then hold the trained $f_r$ fixed and use the same dataset to train a new $f_{\theta_1}$ and $f_{\theta_2}$ from scratch. Finally, we hold the $f_{\theta_1}$ and $f_{\theta_2}$ that we just trained fixed and use them to train a new $f_r$ from scratch. The goal of the last learner is to test the importance of co-learning $f_{\theta_1}$, $f_{\theta_2}$, and $f_r$ for the Y-learner. We label this experiment ‘no co-learning.’ To our surprise, the no co-learning experiment performed much worse than the standard y-learner and the y-learner no backprop experiments. This is evidence supporting our conjecture that co-learning is an important component of the Y-learner. Further research in this area is likely needed to draw more definitive conclusions.
4 Evaluation

4.1 Experiments on Simulated Data

The first task we consider is a synthetic data benchmark used in Künzel et al. (2017). This benchmark has six different data generating processes. Each synthetic dataset is designed to present some difficulty in estimating the CATE. For example, the treatment propensity might be unbalanced, the relationship between the treatment effect and the outcome might be complex, or there might be confounding variables. See Künzel et al. (2017) for a full description of all of the data generating processes.

Figure 2: Testing the importance of co-learning in the Y-learner.

Figure 3: Performance of R, X, Y, S, and T learners on six simulated data benchmark tasks. The data is synthetically generated to make estimating the CATE difficult. We see that the Y-learner delivers the best performance on simulations 1, 2, and 4. On simulations 3, 5, and 6 it delivers comparable final performance to all extant methods. On most simulations, the Y-learner requires the least data to learn a good CATE estimate.
4.2 Get-Out-The-Vote Experiments

This experiment was designed to measure the impact of social pressure on voter turnout in US elections (Gerber et al., 2017). The experiment was carried out by sending a mailer to individuals during the 2014 midterm election season. The mailer contained information about the individual’s voting history in past elections, as well as information about the expected voter turnout for people with similar demographics to the recipient. The inputs, $X$, are the individuals’ demographic information. The outcome $Y$ is a binary indicating whether or not the individual cast a vote. The individual treatment effect $\mu_1(x)$ measures the impact of receiving a flier on voter turnout. The control outcome $\mu_0$ can be thought of as the impact of not receiving a flier on an individual’s propensity to vote. Since encouraging strong voter turnout is an important problem for democracy, it would be valuable to know what kinds of voter encouragement work best across different populations.

4.3 MNIST Experiments

This first version of this task was developed in Nie & Wager (2017), though it was later removed from that paper for unknown reasons. A newer version was proposed in Künzel et al. (2018). In this task, MNIST digits are given a treatment effect. The value of the treatment effect is a function of the number depicted in the image. The task is interesting because the input data is an image. Traditional CATE estimation strategies were not capable of learning treatment effects from raw image inputs. However, when CATE estimators are parameterized by neural networks, image inputs present no special challenges.

5 Closing Remarks

In this paper, we proposed the Y-learner for CATE estimation. The Y-learner was designed specifically with neural networks in mind. It takes advantage of the ability of neural networks to continually optimize against multiple objectives. We noted that the Y-learner was differentiated from the X-learner by its co-learning strategy. The Y-learner achieves excellent performance on three benchmark
Figure 6: Results on the MNIST task. The X, R, and T learners have fairly flat learning curves and end with MSEs of 12.8, 8.6, and 14.1 respectively, so they are omitted here. The S-learner does much better than the Y-learner until there are around 14,000 points in the training set.

problems, including one simulated data benchmark, one real data benchmark, and one benchmark that estimated CATEs over images.

We are left with several open questions. While we did not perform a theoretical analysis on the convergence rate of the Y-learner, it seems likely that the tools from [Nie & Wager, 2017; Künzel et al., 2017] would allow us to do so. There exists a body of related work on imputing or otherwise handling missing counterfactuals using deep learning techniques. The Y-learner too provides a technique for imputing the missing counterfactuals needed for CATE estimation. It would be investigate the links between our scheme and the the recently proposed methods surveyed in this paper. As always, the problem of dealing with confounding variables remains an interesting one. It would be interesting to adapt the Y-learner so that it can tackle this problem more directly.
REFERENCES


