ETH
Swiss Federal Institute of Technology Zurich

Seminar for Statistics

Department of Mathematics

Bachelor Thesis Autumn 2018

Maic Rakitta

A Meta-Algorithm with Variable Selection for heterogeneous Causal Effects

Submission Date: December 17th 2018

Adviser: Prof. Dr. Peter L. Bühlmann

Preface

Fascinated by the answers to the question "How do we learn from data?", this thesis is based on my passion for statistics. In the world of today, where the media is full of words like "machine learning", "artificial intelligence" or "big data", where we talk to our gadgets and machines decide in which direction we walk, my desire to get a grasp of this powerful concepts got stronger and stronger. With this work, I had the possibility to dig deeper into the world of statistics and got familiar with the branch of causal inference. Through that, I was able to confirm my passion and concluded that my future studies will lead toward the world of data.

At this point, I would like to thank Prof. Bühlmann for this opportunity to extend my knowledge in the field of statistics. The liberties as well as the guideline is gratefully acknowledged. I also would like to thank Pascal Oswald for the enlightening conversations, Moritz Haegi for his incorrigible English and Stefanie for always having my back.

Abstract

In statistics we are interested in estimating heterogeneous treatment effects to see how different subjects react to experiments. In many fields it is crucial to know, how subgroups with certain features are influenced by treatments. Who reacts strongly? Who has a very small effect? Are there no effects for others?

There are already quite some approaches trying to best approximate heterogeneous treatment effects, one of which is the X-learner introduced by Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel and Bin Yu. The X-learner can use any supervised learning or regression method which best fits the underlying trend of the to analyzing data. A big advantage of the X-learner is, that it is especially efficient for data sets in which the assignment to control an treatment group is very unbalanced.

In this thesis we will dig into the essentials of the X-learner, summarize the most important results and construct a k-fold-Cross-Validation template for the X-learner. Furthermore, we construct and analyze a variable selection algorithm for the X-learner with random forest as base-learners. We show that there exists scenarios, where we reach a significant better performance, if the X-learner has information about the true dependent features. Moreover, we show empirically, that our variable selection algorithm can decrease the number of features that have to be to considered to estimate the heterogeneous treatment effect.

Contents

List of Figures

Chapter 1

Introduction

Let us assume we are part of a pharmaceutical research team and we want to find the effect of a newly created drug curing headache, to make predictions to patients that our drug is the right treat to cure them. Our drug might affect different patients differently, for some there will be a huge effect, whereas there is just a small one for others, or some will not feel any difference. This unequal result is called heterogeneous treatment effect. To know how our drug works for a particular patient we would have to look at the difference of the headache intensity of the patient with the treatment and without it. But one of the fundamental problems of causal inference is that we cannot observe both of this futures at once Holland (1986). Either our patient takes the drug and we will see the intensity after a time t with the drug, or we do not treat the patient and just monitor the headache intensity after time t. So one of the outcomes is always unknown.

The main focus of this paper will be on the work of Künzel, Sekhon, Bickel, and Yu (2017). They proposed a new approach estimating the heterogeneous treatment effect using a Meta-Algorithm, called X-learner. It first estimates for each subject the unknown futures to derive two kinds of different pseudo-effects, which in the end are weighted to one final estimate. It often occurs, that the assignments to a control and treatment group are very unbalanced. For example, for a new medication, it might be much easier to find more people that did not take it, than to ask people to take the drug. The X-learner can handle unbalanced control and treatment groups particularity well. Furthermore, the Meta-Algorithm is not tied to just use one learning method, to estimate unknown values, but rather can use a combination of different algorithms that fit the underlying sub-problems appropriately.

Finding such methods improving heterogeneous treatment effect estimations get more and more interesting. The era of mass data collection gives access to a huge amount of big data-sets, whereas many fields want to use this multivariate information for causal analysis. For example, personalized medicine aims to treat individuals after their genomic information, in marketing a researcher may want to see the influence of an advertising on the shopping behavior or a social researcher analyzes different support programs for unemployed people. It can be crucial to decide who benefits most from which treatment to be efficient and avoid resource waste. For example, to increase efficiency Künzel et al. found impressionable subgroups in two real-life studies: using social pressure to influence voter turnout by Gerber, Green, and Larimer (2008) and reducing prejudgment of transgenders by Broockman and Kalla (2016).

With the growing of possible applications in causal analysis there are next to Künzel et al. (2017) also a lot of other approaches that try to adapt to different scenarios. Some researcher try to estimate the Average Treatment Effect (ATE) like Pearl (2009) or Imbens and Rubin (2015), but this does not take account of the individuality of the individuals rather than just describe the effectiveness of the treatment overall. To take account of the individuality recent approaches try to estimate the so called Conditional Average Treatment Effect (CATE) with different techniques, like: random forest Wager and Athey (2017), recursive partitioning Athey and Imbens (2016), also a Meta-algorithm Foster, Taylor, and Ruberg (2011) or even closer related the R-learner which also gets compared to the X-learner by its authors Nie and Wager (2017).

To consolidate the structure of this work, we state the following fundamental questions:

1. *How does the X-learner introduced by Künzel et al. (2017) work? What are the core results of Künzel et al. (2017) comparing the X-learner to other Meta-learners?*
2. *Considering high-dimensional feature vectors, does the X-learner with random forest as base-learners improve, if we provide information to the second base-learners on which features the true CATE depends?*
3. *Considering high-dimensional feature vectors, can we implement automatic variable selection for the CATE into the X-learner with Random Forest as base-learners?*

For the first question we start with an overview of the necessary definitions (chapter 2) and construction of Künzel et al. algorithm (chapter 3), which we than modify to get an algorithm with Cross-Validation (CV) that can be used for model assessment (introduction about CV can be found in Hastie, James, and Tibshirani (2013)). After we looked at a motivation example for intuition (chapter 4), we will summarize some results of the simulations of Künzel et al. (2017) to see some benefits of the X-learner compared to two other Meta-algorithms (chapter 5). The growing interests in managing multidimensional data makes it very useful to have tools for variable selection. So for question two and three we head with the construed base-knowledge about the X-learner to chapter 6, where we implement a new variable selection algorithm for heterogeneous treatment effects by use of the X-learner that uses random forests Breiman (2001) as base-learners and evaluate the performance of the constructed algorithm in a simulation scenario. All the programming is done in R , where the corresponding pseudo codes can be found in Appendix A and the R codes in Appendix B.

Chapter 2

Setup

Starting point is the Neyman-Rubin potential outcome framework (Rubin (1974), Splawa-Neyman, Dabrowska, and Speed (1990)) and we construct the setup analogously to Künzel et al. (2017). Assume we have n independent and identically distributed samples (X_i, Y_i, W_i), $i = 1, ..., n$. $X_i \in \mathbb{R}^d$ is a vector of feature values, $W_i \in \{0, 1\}$ is the treatment assignment (0 for control, 1 for treatment) and $Y_i = Y_i^{W_i} \in \mathbb{R}$ is the observed outcome, where $Y_i^{W_i}$ is the potential outcome of the i-th subject being either in the control or treatment group. The so called Individual Treatment Effect (ITE) of the i-th individual is defined as

$$\text{ITE} := D_i := Y_i^1 - Y_i^0, \tag{2.1}$$

but unfortunately we can only observe either the treated ($W_i = 1$) or untreated ($W_i = 0$) response of an individual at the time. To decide if it is a good idea to treat a new individual we could use the Average Treatment Effect (ATE), defined as

$$\text{ATE} := \mathbb{E}[Y^1 - Y^0].$$

As described in the following chapter, Künzel et al. go beyond the ATE and try to take the individuality of the subjects into account by estimating the Conditional Average Treatment Effect (CATE), defined as

$$\text{CATE}(x) := \tau(x) := \mathbb{E}[D|X = x] = \mathbb{E}[Y^1 - Y^0|X = x], \tag{2.2}$$

by use of the response under control and treatment, defined as

$$\mu_0(x) := \mathbb{E}[Y^0|X = x] \quad \text{and} \quad \mu_1(x) := \mathbb{E}[Y^1|X = x].$$

To evaluate an estimate of the CATE, they say the estimate is a good approximation whenever it has a small Expected Mean Squared Error (EMSE),

$$\text{EMSE}(\hat{\tau}) = \mathbb{E}_X^{\hat{}}[(\underbrace{\tau(X)}_{\text{true CATE}} - \underbrace{\hat{\tau}(X)}_{\text{estimate}})^2].$$

To see that a good estimate of the CATE also gives a good estimate for the ITE, we state the following statement of Künzel et al. (2017) as a proposition and furthermore give a detailed prove.

Proposition. Consider an observation (X_i, Y_i, W_i), where we assume D_i is the ITE (2.1) and τ_i is the corresponding true CATE (2.2).

If $\hat{\tau}_i$ is the best estimator for the CATE, then it is also the best estimator for the ITE.

Proof. To see this, we consider the mean squared error of the ITE and the CATE estimate, conditional on the feature vector X_i and compute

$$\begin{aligned}
\mathbb{E}[(D_i - \hat{\tau}_i)^2 | X_i = x_i] &= \mathbb{E}[(D_i - \tau(x_i) + \tau(x_i) - \hat{\tau}_i)^2 | X_i = x_i] \\
&= \mathbb{E}[(D_i - \tau(x_i))^2 + 2(D_i - \tau(x_i))(\tau(x_i) - \hat{\tau}_i) + (\tau(x_i) - \hat{\tau}_i)^2 | X_i = x_i] \\
&\overset{(1)}{=} \mathbb{E}[(D_i - \tau(x_i))^2 | X_i = x_i] + \mathbb{E}[(\tau(x_i) - \hat{\tau}_i)^2 | X_i = x_i] \\
&\quad + 2\mathbb{E}[(D_i - \tau(x_i))(\tau(x_i) - \hat{\tau}_i) | X_i = x_i],
\end{aligned}$$

where in the last equality (1) we used linearity of conditional expectation. We next look at the last term of the expression to find that it is equal to zero.

$$\begin{aligned}
\mathbb{E}[(D_i - \tau(x_i)) \overbrace{(\tau(x_i) - \hat{\tau}_i)}^{X_i\text{-measurable}} | X_i = x_i] &= (\tau(x_i) - \hat{\tau}_i)\mathbb{E}[D_i - \tau(x_i) | X_i = x_i] \\
&= (\tau(x_i) - \hat{\tau}_i)\mathbb{E}[D_i - \mathbb{E}[D_i | X_i = x_i] | X_i = x_i] \\
&\overset{(2)}{=} (\tau(x_i) - \hat{\tau}_i)(\mathbb{E}[D_i | X_i = x_i] - \mathbb{E}[\mathbb{E}[D_i | X_i = x_i] | X_i = x_i]) \\
&\overset{(3)}{=} (\tau(x_i) - \hat{\tau}_i)(\mathbb{E}[D_i | X_i = x_i] - \mathbb{E}[D_i | X_i = x_i]) = 0,
\end{aligned}$$

where we used again linearity (2) and in the equality (3) the tower property of conditional expectation. This finally leads us to

$$\begin{aligned}
\mathbb{E}[(D_i - \hat{\tau}_i)^2 | X_i = x_i] &= \mathbb{E}[(D_i - \tau(x_i))^2 | X_i = x_i] + \mathbb{E}[\overbrace{(\tau(x_i) - \hat{\tau}_i)^2}^{X_i\text{-measurable}} | X_i = x_i] \\
&= \mathbb{E}[(D_i - \tau(x_i))^2 | X_i = x_i] + \mathbb{E}[(\tau(x_i) - \hat{\tau}_i)^2]
\end{aligned}$$

Since the two terms are positive due to the square and because we cannot influence $\mathbb{E}[(D_i - \tau(x_i))^2 | X_i = x_i]$, it follows, that the estimate $\hat{\tau}_i$ that minimizes $\mathbb{E}[(\tau(x_i) - \hat{\tau}_i)^2]$ also leads to the minimal possible term $\mathbb{E}[(D_i - \hat{\tau}_i)^2 | X_i = x_i]$. Hence, considering MSE, the best estimate for the CATE is also the best estimate for the ITE. □

2.1 Propensity Score

In reality treatments are often not randomly assigned to individuals, it can occur that some features are the reason for a certain allocation. Assuming randomized assignment in this case would lead to selection bias. To avoid this selection bias Künzel et al. use propensity score for the X-learner, following Rubin and Rosenbaum (1983), such that the treatment assignment has a Bernoulli-distribution with the propensity score $e(X)$ as probability, in particular $W_i \sim Bern(e(X))$. The propensity score, defined as

$$e(x) := P(W = 1|X = x), \tag{2.3}$$

is the probability that a unit with certain features is assigned to the treatment group, rather than to the control group, where we assume

$$P(W_1, ..., W_n|X_1, ..., X_n) = \prod_{i=1}^{n} e(X_i)^{W_i}(1 - e(X_i))^{1-W_i}.$$

As showed in Rubin and Rosenbaum (1983) we need two assumptions to hold, to correctly estimate the CATE:

Assumption 1. Let the observed outcomes be composed as

$$Y^0 = \mu_0(X) + \epsilon_0 \quad \text{and} \quad Y^1 = \mu_1(X) + \epsilon_1,$$

where ϵ_0, ϵ_1 are random noise. Then we assume, that there are no hidden confounders, i.e. the treatment assignment is independent of the random noise $(\epsilon_0, \epsilon_1) \perp W|X$.

Assumption 2. Let $e(x)$ be the propensity score defined in (2.3). Then we assume

$$\forall x \in X : 0 < \min_{x \in X} e(x) < e(x) < \max_{x \in X} e(x) < 1.$$

It is also possible to have a setup with more than one treatment. For example, we could investigate a medication with different dosages or look at several different trainee programs for non-employees. While Rubin and Rosenbaum showed the propensity score approach solely for the case where there is a control and one treatment group, Imbens (1999) extended the framework to also work for multi-valued treatment assignments. Nevertheless, in this thesis we work for simplification like Künzel et al. only with one treatment.

Chapter 3

The X-learner

As mentioned in the introduction Künzel et al. constructed the X-learner by using information of the underlying sub-problems, that estimates the response under control and treatment before it estimates the CATE. Given the definitions in Chapter 2, we will here have a closer look to the main three steps of the Meta-learner and provide for a good overview the pseudo code of Künzel et al. (2017) in the Appendix A.

1. **Estimation of the response under control and treatment**
   Depending on the feature vector X the X-learner first estimates the response under control and treatment

   $$\mu_0(x) := \mathbb{E}[Y^0|X = x] \qquad \mu_1(x) := \mathbb{E}[Y^1|X = x]$$

   by using any supervised learning or regression method as first base-learners M_0, M_1 and for each response the corresponding features to get

   $$\hat{\mu}_0 = M_0(Y^0 \sim X^0) \qquad \hat{\mu}_1 = M_1(Y^1 \sim X^1).$$

2. **Construction of the imputed treatment effects**
   With the fitted model for $\hat{\mu}_0$ the X-learner estimates for the treated samples with feature vectors X_i^1 the outcomes under control and uses it to estimate the true ITEs D_i, by deriving the so called imputed treatment effects $\tilde{D}_i^1$ and analogously $\tilde{D}_i^0$:

   $$\tilde{D}_i^0 := \hat{\mu}_1(X_i^0) - Y_i^0 \qquad \tilde{D}_i^1 := Y_i^1 - \hat{\mu}_0(X_i^1).$$

   With this information the X-learner can estimate the true CATE $\tau(x)$ in two ways,

   $$\tau_0(x) := \mathbb{E}[\tilde{D}^0|X^0 = x] \qquad \tau_1(x) := \mathbb{E}[\tilde{D}^1|X^1 = x],$$

   using the feature vector X and the imputed treatment effects $\tilde{D}_i^0$ and $\tilde{D}_i^1$. Like in the first step we can use any supervised learning or regression method as second base-learners M_2, M_3 to get

   $$\hat{\tau}_0 = M_2(\tilde{D}^0 \sim X^0) \qquad \hat{\tau}_1 = M_3(\tilde{D}^1 \sim X^1).$$

3. **Final CATE estimation**
   The output CATE estimate of the X-learner is the weighted average of $\hat{\tau}_0$ and $\hat{\tau}_1$, where $g \in [0, 1]$ is the weight function. For any feature vector x the predicted CATE is derived as

   $$\hat{\tau}(x) = g(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x).$$

Especially if we have unbalanced treatment assignments Künzel et al. recommend to use for the weight function an estimation of the propensity score, $g = \hat{e}$, as described in Chapter 2.1. For example, if we have way more observations in the control group, the propensity score will note this and shift the final CATE estimate towards $\hat{\tau}_1$, which than is more reliable than $\hat{\tau}_0$. But also taking the proportion of the two assignment groups can give satisfying results.

3.1 Cross-Validation Template

If we want to examine the performance of different models, we can define a framework and generate data on which we train the models. Based on the framework we then can generate new out-of-sample test data in the same way we generated the training data, to compare the different models. In this way the predictions are honest and not good because of over-fitting of a model. On the contrary, if we want to examine different models on a real life data-set, it can be difficult to generate more data. The researcher often is bound to a certain amount of data and needs to use the given data-set as training and test data. So it is nice to have a method for the X-learner at hand that efficiently uses a bounded number of observations. A strong method to best use a data-set for model fitting and also use it for evaluation is Cross-Validation. So in this chapter we provide an implementation of the X-learner using 10-fold-Cross-Validation for performance evaluation. With this template one can for example tune parameters of the base-learners on a real-world problem and compare the results appropriately.

Let us assume our aim is to make predictions about the treatment effect for a unused observation with feature vector X_{new}. With the trained X-Learner we get an estimate of the CATE, $\hat{\tau}(X_{\text{new}})$, which we can add to the estimate of the untreated effect, $\hat{\mu}_0(X_{\text{new}})$, to get an estimated outcome with the treatment

$$\hat{Y}^1_{\text{new}}(X_{\text{new}}) = \hat{\mu}_0(X_{\text{new}}) + \hat{\tau}(X_{\text{new}}).$$

Especially, if we have way more observations in the control group, this is a good approximation of the treated outcome, because $\hat{\mu}_0$ is estimated based on the control group and Künzel et al. showed in their paper that also the X-learner provides a good estimate in such unbalanced scenarios. If we have more observations in the treatment group we could create an estimate analogously using a difference.

So if we take observations of which we know the true outcome under treatment and also estimate it as above, we can derive the deviance of the two values. In that sense, we construct a template of the X-learner combined with 10-fold-Cross-Validation to make statements about model assessment, in the case where it is not simple to generate test data. For a good overview the pseudo-code is attached in the Appendix A.

1. **Before the loop**
   We estimate the response under control

   $$\mu_0(x) := \mathbb{E}[Y^0|X = x],$$

   based on the control observations (X^0, Y^0), by using any base-learner M_0 that fits best to get an estimate

   $$\hat{\mu}_0 = M_0(Y^0 \sim X^0).$$

Because $\hat{\mu}_0$ does not change in the different iterations, we can do this in advance. We also have to prepare a storage unit to keep track of the error term in each iteration and create ten independent more or less equally sized folds, to create train and test data in each iteration.

2. The loop
In each of the ten iterations we do

i) take each fold once as test and the remaining as training index, to partition the treatment observations (X^1, Y^1) as

$$(X^1_{\text{test}}, Y^1_{\text{test}}) \leftarrow (X^1, Y^1)[\text{ fold }]$$
$$(X^1_{\text{train}}, Y^1_{\text{train}}) \leftarrow (X^1, Y^1)[\text{ fold}^c].$$

ii) train the estimate of the response under treatment $\hat{\mu}_1$ on the training data $(X^1_{\text{train}}, Y^1_{\text{train}})$ using any base-learner M_1

$$\hat{\mu}_1 = M_1(Y^1_{\text{train}} \sim X^1_{\text{train}}).$$

iii) derive the imputed treatment effects, $\tilde{D}^0$ and $\tilde{D}^1$, based on the current estimates of the response functions, $\hat{\mu}_0$ and $\hat{\mu}_1$, the treatment training data $(X^1_{\text{train}}, Y^1_{\text{train}})$ and the whole control data (X^0, Y^0)

$$\tilde{D}^0 = \hat{\mu}_1(X^0) - Y^0 \qquad \tilde{D}^1 = Y^1_{\text{train}} - \hat{\mu}_0(X^1_{\text{train}}).$$

iv) estimate the CATE, $\tau(x)$, in two ways

$$\tau_0(x) := \mathbb{E}[\tilde{D}^0 | X^0 = x] \qquad \tau_1(x) := \mathbb{E}[\tilde{D}^1 | X^1 = x],$$

by use of the imputed treatment effects $\tilde{D}^0$ and $\tilde{D}^1$, the treatment training data $(X^1_{\text{train}}, Y^1_{\text{train}})$ and the whole control data (X^0, Y^0) with any base-learners, M_2 and M_3

$$\hat{\tau}_0 = M_2(\tilde{D}^0 \sim X^0) \qquad \hat{\tau}_1 = M_3(\tilde{D}^1 \sim X^1_{\text{train}}).$$

v) estimate the output CATE, $\hat{\tau}$, of the unused test data X^1_{test}

$$\hat{\tau}(X^1_{\text{test}}) = g(X^1_{\text{test}})\hat{\tau}_0(X^1_{\text{test}}) + (1 - g(X^1_{\text{test}}))\hat{\tau}_1(X^1_{\text{test}}).$$

vi) estimate the treatment outcome of the test data using the CATE, $\hat{\tau}$, and the response under control, $\hat{\mu}_0$,

$$\hat{Y}^1_{\text{test}} = \hat{\mu}_0(X^1_{\text{test}}) + \hat{\tau}(X^1_{\text{test}}).$$

vii) keep track of the deviation with an appropriate loss function ρ

$$\eta_j = \rho(\hat{Y}^1_{\text{test}}, Y^1_{\text{test}}).$$

3. After the loop
Finally we can evaluate the mean of the errors to get a value L for model assessment

$$L = \text{mean}(\eta), \qquad \text{where } \eta = (\eta_1, ..., \eta_{10}).$$

Chapter 4

Motivation: The unbalanced case

After we looked at the construction of the X-learner, we now want to have some insight by looking at a similar intuition example like Künzel et al. used in their paper. A problem that often occurs in treatment analysis is, that one observes many more control units than treatment units. As mentioned before the X-learner can handle unbalanced designs very well, what we will visualize in a simple example. We assume a model with just one feature, where we have very few treated units ($m = 10$) and many more control units ($n = 200$). The response is a step-function with some random noise ϵ, whereas the true CATE is constant one.

$$
\begin{aligned}
&X_i^0, X_j^1 \in [-1, 1], \quad i \in \{1, ..., n\}, \quad j \in \{1, ..., m\} \\
&Y^0(X) = 1 + \frac{1}{4}\mathbb{1}_{[-0.05, 0.05)}(X) + \frac{1}{2}\mathbb{1}_{[0.05, 0.45)}(X) + \frac{1}{4}\mathbb{1}_{[0.45, 0.55)}(X) + \epsilon \\
&Y^1(X) = Y^0(X) + \tau, \qquad \textit{where } \tau \equiv 1, \quad \epsilon \sim \frac{1}{6}\mathcal{N}(\mu = 0, \sigma^2 = 1), \quad X \in \{X_i^0, X_j^1\}
\end{aligned}
$$

First we want to fit the response under control and treatment by looking at the generated data in Figure 4.1. If we look at the treatment observations (X^1, Y^1), we have with just ten data-points very little information. So to not risk over-fitting we use a simple model like linear regression to estimate $\mu_1(x) = \mathbb{E}[Y^1|X = x]$ (blue line). On the other hand, having a lot of observations in the control group (X^0, Y^0), we know more about the true structure. So we are not feared to over-fit with a more complex model like piece-wise linear regression to estimate $\mu_0(x) = \mathbb{E}[Y^0|X = x]$ (red line).

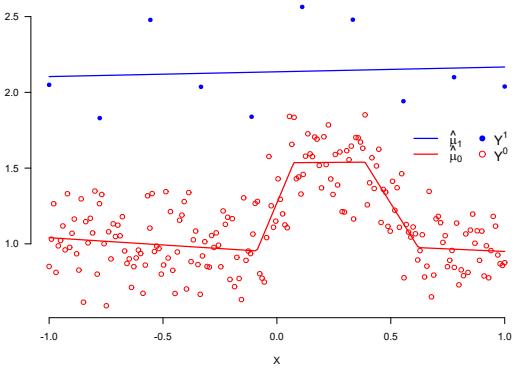

Figure 4.1: Estimated response functions

In Figure 4.3 we can see, that estimating the CATE by taking the difference of the two estimated response functions, $\hat{\tau}(X) = \hat{\mu}_1 - \hat{\mu}_0$, leads to an estimate with huge jumps, whereas it should be a constant, $\tau \equiv 1$. The reason is that we have not enough observations in the treatment group to estimate the complex CATE like this. We correctly avoided over-fitting by not taking a to strong model for the response under treatment μ_1, and simultaneous used a more flexible fit for the response under control μ_0. However, taking

the difference of the response functions for the estimation increases also the flexibility of the CATE, our actual quantity of interest.

The ploy of the X-learner is to not aim for a perfect response estimation, but rather to estimate $\hat{\mu}_0$ and $\hat{\mu}_1$ such that their difference properly estimates the CATE, although the unbalanced assignments. So looking at the second step of the X-learner, we compute the imputed treatment effects

$$\tilde{D}_i^0 := \hat{\mu}_1(X_i^0) - Y_i^0 \qquad \tilde{D}_j^1 := Y_j^1 - \hat{\mu}_0(X_j^1)$$

and plot it in Figure 4.2. Having just ten data-points for $\tilde{D}_j^1$ and many more for $\tilde{D}_i^0$, we may use again a linear and a piece-wise linear model to estimate the CATE in two ways

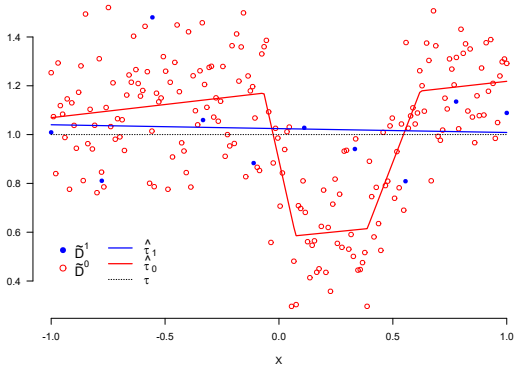

Figure 4.2: Imputed treatment Effects

$$\tau_0(x) := \mathbb{E}[\tilde{D}^0 | X^0 = x] \qquad \tau_1(x) := \mathbb{E}[\tilde{D}^1 | X^1 = x].$$

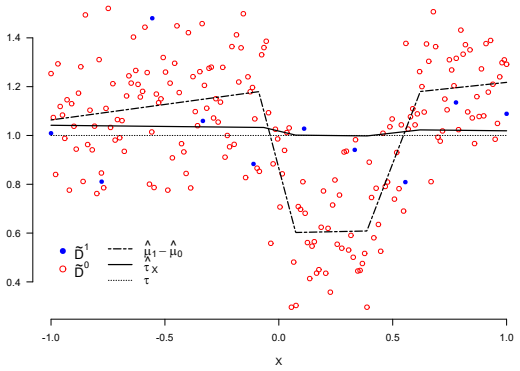

Figure 4.3: Good and bad CATE estimate

On the one hand, if we compare Figure 4.1 and Figure 4.2, we see that the shape of $\hat{\tau}_1$ looks very similar to the shape of $\hat{\mu}_1$. On the other hand, $\hat{\tau}_0$ is like $\hat{\mu}_0$ also piece-wise linear, where the jumps are at the same points, but because of the difference in the other direction.

In the last step of the X-learner we combine these results to our final CATE estimate by choosing the weight to be an estimate of the propensity score as defined in chapter 2.1. Because we have more units in the control group, our result is postponed towards $\hat{\tau}_1$.

$$\hat{\tau}(x) = \underbrace{g(x)}_{\text{close to 0}} \hat{\tau}_0(x) + \underbrace{(1 - g(x))}_{\text{close to 1}} \hat{\tau}_1(x) \approx \hat{\tau}_1(x)$$

As we can see in Figure 4.3 this results in an estimate closer to the true CATE compared to if we just take the difference of the response functions.

Chapter 5

Simulation Results Summary

Before we investigate our own simulation study in chapter 6, we summarize the result of the detailed simulation study of Künzel et al. (2017). They showed that there are certain situations where the X-learner works particularly well, or that overall the X-learner always gives a more or less satisfying result compared to some other Meta-Algorithms estimating the CATE. To show that, the authors created six prototypical situations, on which they trained next to the X-learner also the so called S- and T-learner. The six situations are[1]:

1: Unbalanced control and treatment assignment.
2: The true CATE is a complex linear function.
3: The true CATE is a complex non-linear function.
4: The true CATE is zero, where the response functions are globally linear.
5: The true CATE is zero, where the response functions are piecewise linear.
6: The true CATE is zero, where the treatment assignment is non-random.

The three Meta-learner have in common that they are not bound to one base-learner. The base-learners they used in the simulation study were modified versions of random forest (RF) and bayesian additive regression trees (BART), with the intention that this machine learning algorithms work well for a large variety of different situations. Nevertheless, depending on the data, other machine learning algorithms can perform particularly better. For a better understanding of the results of Künzel et al. we will briefly introduce the S- and T-learner and point out the advantages and disadvantages.

5.1 The S-learner

The S-learner treats the treatment indicator W as a feature in the same way it treats the other features X to estimate the combined response function μ_S. As the X-learner it can use any desired base learner M.

$$\mu(x,w) := \mathbb{E}[Y|X=x, W=w] \qquad \text{estimated by} \qquad \hat{\mu}_S := M(Y \sim (X,W))$$

The estimate $\hat{\mu}_S$ is used for the estimation of the CATE by taking the difference of the combined response function conditional on the control and treatment assignment,

$$\hat{\tau}_S(x) = \hat{\mu}_S(x,1) - \hat{\mu}_S(x,0).$$

[1] More details on the six simulation setups can be found in Künzel et al. (2017) Appendix A.

Because the S-learner handles the treatment indicator W not any differently than the other features X, some algorithms that do variable selection could exclude the treatment assignment, leading to no distinction between control and treatment. Therefore, this can be advantageous, if there is no treatment effect, but on the other hand, can lead to a bias towards zero, although there is a treatment effect.

+ Performs well, if there is no treatment effect or if at least the CATE is often zero.
− Can be biased towards zero.

Result: The S-learner worked good for Simulation 4, 5 and 6.

5.2 The T-learner

The first step of the T-learner is exactly the same as for the X-learner. It estimates the response under control and treatment, μ_0 and μ_1, by use of any base learners, M_0 and M_1.

$$\begin{aligned} \mu_0(x) &:= \mathbb{E}[Y^0|X=x] \\ \mu_1(x) &:= \mathbb{E}[Y^1|X=x] \end{aligned} \quad \text{estimated by} \quad \begin{aligned} \hat{\mu}_0 &= M_0(Y^0 \sim X^0) \\ \hat{\mu}_1 &= M_1(Y^1 \sim X^1) \end{aligned}$$

The CATE is than estimated by taking the difference of the response under control and treatment,

$$\hat{\tau}_T(x) = \hat{\mu}_1(x) - \hat{\mu}_0(x).$$

It shall be noted, that the T-learner fits the control and treatment group separately, doing no combination of the data. Therefore, it is difficult for the T-learner to learn common behavior of the response functions.

+ Performs well, if there are no common trends in the response functions μ_0 and μ_1, while the true CATE is rather complicated.
− Performs poorly, if the true CATE is simple[2].

Result: The T-learner worked good for Simulation 2 and 3, but extraordinary bad for Simulation 4 and 5.

5.3 The X-learner

The mechanics of the X-learner are explained in detail in chapter 3, where the motivation example of chapter 4 supports understanding the advantages of the X-learner in an unbalanced scenario. For deeper insight, there exist also some theoretical proofs in Künzel et al. (2017) concerning the behavior of the X-learner in such unbalanced cases.

+ Performs well, if there are unbalanced treatment and control groups.
+ Performs well, if there exist some structural assumptions on the CATE.
+ The X-learner has a good overall performance compared to the S- and T-learner.
− The X-learner gets outperformed by the S-learner, if the true CATE is zero.

[2] See also the Motivation example in chapter 4.

Result: The X-learner was never the worst Meta-Algorithm for all of the six simulations. It outperformed the S- and T-learner in Simulation 1, 2 and 3. In Simulation 4, 5 and 6 the X-learner got outperformed by the S-learner, but it is significantly better than the T-learner.

Furthermore, Künzel et al. also compared RF and BART as base-learners in each simulation. Both of them are regression tree based algorithms, where both act globally by concerning all of the data for each prediction. Compared to each other, RF behaves better for data that is more locally structured, whereas BART seems to hold its focus on the global structures. After comparing the two base-learner for each of the six situations, they concluded a good rule of thumb:

Small datasets: X-learner with BART as base-learners in each stage

Bigger datasets: X-learner with RF as base-learners in each stage

Chapter 6

X-learner with Variable Selection

In this chapter we want to investigate variable selection of the X-learner with random forest as base-learners. Let us assume we have a medication that should cure a certain disease. Unfortunately, we do not exactly know, which of a huge set of feature genes are influential to the medication. However, we assume it influences some of this genes, such that the disease intensity decreases. So what we want to investigate is, that if we knew that this medication influence only certain genes of a patient, and we provide this information to the X-learner, does this increase the prediction accuracy of the CATE in terms of test MSE? We will first look at different true CATEs that follow the description above and train the X-learner once with all of the features and once with only the true depended features. Having some insight of the influence of knowing the right features, we continue by constructing a greedy algorithm, that automatically does variable selection integrated in the X-learner with random forest. The algorithm is than evaluated in some simulation examples.

6.1 Simulation Setup

Similar to Künzel et al. (2017) we provide here the construction of the data for all of the simulations of this chapter[1]. For all of the simulations, μ_0 is a simple linear function, depending on a d-dimensional feature vector,

$$\mu_0(x) := x^T \beta_{\mu_0}, \qquad \text{where } \beta_{\mu_0} \sim \text{Unif}([5,10]^d),$$

whereas the true CATE is different in every simulation, but always depending only on a subgroup of the features. μ_1 will than be simply derived by adding the true CATE to the response under control. The different CATE's are:

i) Simple linear CATE for $d = 20, x = (x_1, ..., x_{20})$ and $\beta_1, \beta_2, \beta_3 \sim \text{Unif}([20,30])$:

$$\tau(x) := \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \tag{6.1}$$

ii) Complex linear CATE for $d = 20, x = (x_1, ..., x_{20})$ and $\beta_1, \beta_2 \sim \text{Unif}([3,5])$:

$$\tau(x) := \beta_1 x_1^2 + \beta_2 x_2 + x_2 x_3 \tag{6.2}$$

[1]For all the details the complete R codes can be found in Appendix B.

iii) Complex non-linear CATE for $d = 100, x = (x_1, ..., x_{100})$ and $\beta_1, \beta_2, \beta_3, \beta_4 \sim \text{Unif}([20, 30])$:

$$\tau(x) := \beta_1 \cos(\beta_2 x_1)^2 + \beta_3 x_2 + x_2 x_3 + \beta_4 x_4 - x_4 x_5^2 \tag{6.3}$$

The training and test data is generated as follows:

1. Generate feature matrix $X \in \mathbb{R}^{n \times d}$, where n is the number of observations and d the feature dimension. For each observation we generate the d-dimensional feature vector with correlation matrix Σ:

$$X_i \overset{iid}{\sim} \mathcal{N}(0, \Sigma), \qquad \text{for } i = 1, ..., n.$$

2. For each simulation scenario, we derive the predefined true response functions, μ_0, μ_1, and the true CATE τ.

3. Next we can simulate for each observation the potential outcomes under control and treatment, considering some noise:

$$\begin{aligned} Y_i(0) &= \mu_0(X_i) + \epsilon_i(0) \\ Y_i(1) &= \mu_1(X_i) + \epsilon_i(1) \end{aligned} \qquad \text{where} \qquad \epsilon_i(0), \epsilon_i(1) \sim \mathcal{N}(0, 1).$$

4. Finally, we assign the observations to control and treatment groups according to the propensity score e defined in chapter 2.1,

$$W_i \sim Bern(e(X_i)).$$

6.2 Knowing the true feature dependency

Given the generating steps and the three different true CATEs, we construct six simulations and plot the test MSE of the X-learner trained on all the features (red) in Figure 6.1. In addition, we also plot the result, if we provide to the second base-learner the information on which of the d features the true CATE depends (blue). Because in this simulations we know all the true functions, we can easily generate new out-of-sample data to derive the plotted test MSE. The plots show that there exists indeed cases where knowing the true dependency of the CATE and providing the information to the algorithm, can significantly improve the performance of the X-learner. Therefore, it makes sense in scenarios like this to search for the most influential features rather than just do estimations based on all of the features. With this result in mind we head to the next subsection, where we create a automatic variable selection algorithm for the X-learner.

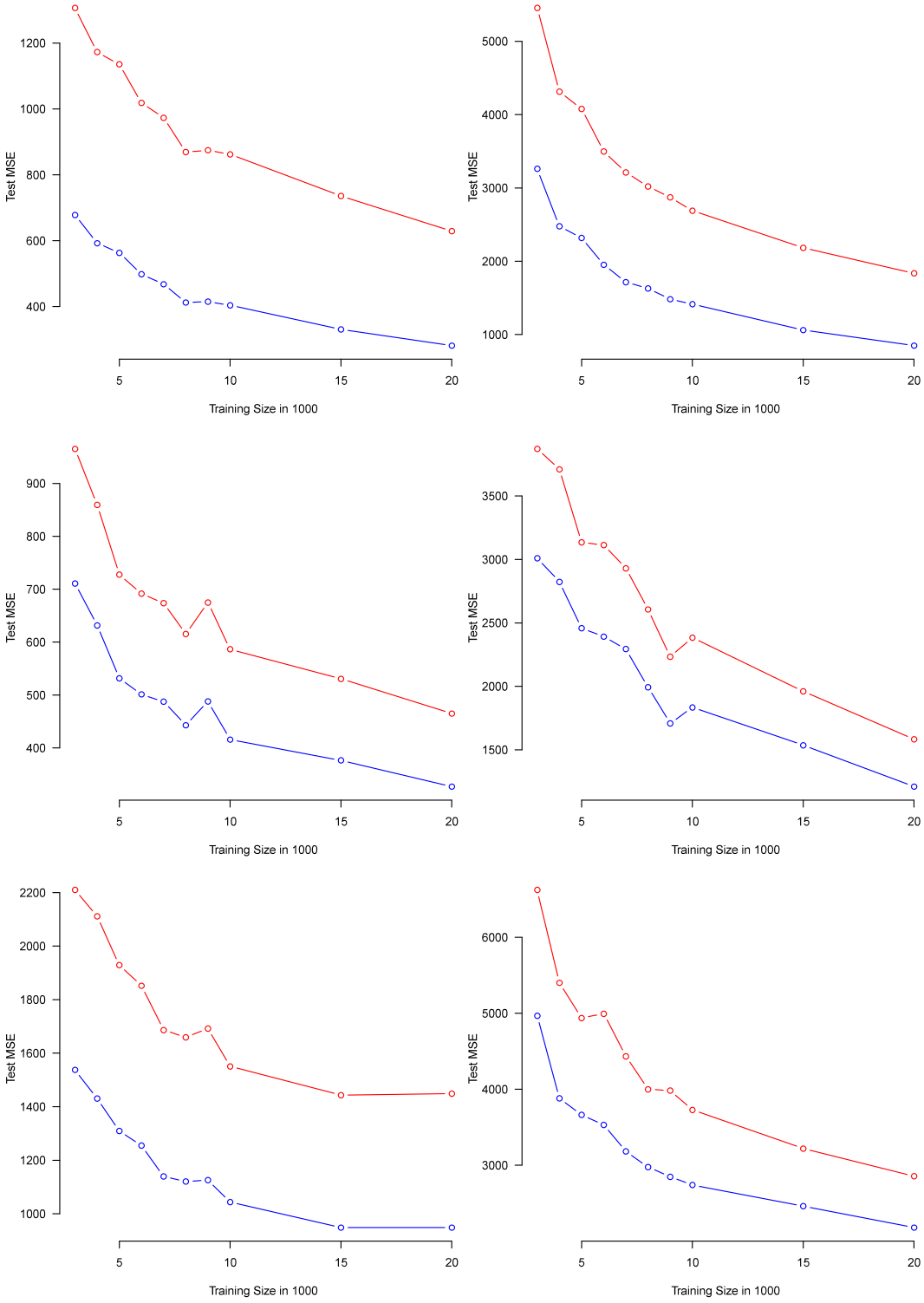

Figure 6.1: *Comparison of the X-learner without (red) and with (blue) providing information about feature dependency to the second base-learners.* Top left: *True CATE* (6.1), *propensity score 0.5.* Top right: *True CATE* (6.1), *propensity score 0.01.* Middle left: *True CATE* (6.2), *propensity score 0.5.* Middle right: *True CATE* (6.2), *propensity score 0.01.* Bottom left: *True CATE* (6.3), *propensity score 0.5.* Bottom right: *True CATE* (6.3), *propensity score 0.01.*

6.3 Automatic Variable Selection

The starting-point is the one we described in the beginning of this chapter. We know that the true CATE depends on just a subgroup of the given features, but we do not know on which of them. Our aim is to create a greedy-algorithm that automatically does variable selection for the CATE. In this example, we use random forest for all base-learners and the variable selection is based on the common variable importance measure for random forests. The first two steps are without any changes compared to the X-learner, but once more listed for better understanding. We implement the following algorithm, where for a good overview the pseudo-code can be found in Appendix A:

1. **Estimation of the response under control and treatment**
   Depending on the feature vector X the X-learner first estimates the response under control and treatment

   $$\mu_0(x) := \mathbb{E}[Y^0|X = x] \qquad \mu_1(x) := \mathbb{E}[Y^1|X = x]$$

   by using random forest (RF) as first base-learners and for each response the corresponding features to get

   $$\hat{\mu}_0 = \mathrm{RF}(Y^0 \sim X^0) \qquad \hat{\mu}_1 = \mathrm{RF}(Y^1 \sim X^1).$$

2. **Construction of the imputed treatment effects**
   With the fitted model for $\hat{\mu}_0$ the algorithm estimates for the treated samples with feature vectors X_i^1 the outcomes under control and uses it to derive the imputed treatment effects $\tilde{D}_i^1$ and analogously $\tilde{D}_i^0$:

   $$\tilde{D}_i^0 := \hat{\mu}_1(X_i^0) - Y_i^0 \qquad \tilde{D}_i^1 := Y_i^1 - \hat{\mu}_0(X_i^1).$$

3. **Variable Selection for the CATE**
   Based on the imputed treatment effects the algorithm starts variable selection on all of the features $X^{\mathrm{VS}} = (X_0, X_1) := (X^0, X^1)$ and stops after it has eliminated K features. For each of the K iterations it does

   i) estimate the CATE $\tau(x)$ in two ways,

   $$\tau_0(x) := \mathbb{E}[\tilde{D}^0|X^0 = x] \qquad \tau_1(x) := \mathbb{E}[\tilde{D}^1|X^1 = x],$$

   using the corresponding features X^{VS}, the imputed treatment effects $\tilde{D}_i^0, \tilde{D}_i^1$ and RF as second base-learners to get

   $$\hat{\tau}_0 = \mathrm{RF}(\tilde{D}^0 \sim X_0) \qquad \hat{\tau}_1 = \mathrm{RF}(\tilde{D}^1 \sim X_1).$$

   ii) based on variable importance measure[2] (imp) find the least important features

   $$f^{\hat{\tau}_0} = \min_{f \in X^{\mathrm{VS}}} \mathrm{imp}^{\hat{\tau}_0}(X^{\mathrm{VS}}), \qquad f^{\hat{\tau}_1} = \min_{f \in X^{\mathrm{VS}}} \mathrm{imp}^{\hat{\tau}_1}(X^{\mathrm{VS}})$$

[2] As can be seen in the corresponding R codes, one can for example use the incMSE of the importance() function. The incMSE is the increase of the MSE if a variable is assigned values by random permutation.

iii) if the two models do not choose the same feature as the least important, it compares $f^{\hat{\tau}_0}$ and $f^{\hat{\tau}_1}$, to decide which of them is the least important one concerning both models by

$$f_{\text{tot}}^{\hat{\tau}_0} = \text{imp}^{\hat{\tau}_0}(f^{\hat{\tau}_0}) + \text{imp}^{\hat{\tau}_1}(f^{\hat{\tau}_0})$$
$$f_{\text{tot}}^{\hat{\tau}_1} = \text{imp}^{\hat{\tau}_0}(f^{\hat{\tau}_1}) + \text{imp}^{\hat{\tau}_1}(f^{\hat{\tau}_1})$$
$$f_{\text{tot}} = \min_{f \in X^{\text{VS}}}(f_{\text{tot}}^{\hat{\tau}_0}, f_{\text{tot}}^{\hat{\tau}_1}).$$

iv) exclude the most unimportant chosen feature f_{tot} from the features $X^{\text{VS}} = (X_0, X_1)$.

4. **Estimate CATE on the remaining features**
After finishing variable selection we remain with the $d-K$ most important features. We estimate the CATE in two ways using RF and the remaining features by

$$\hat{\tau}_0^{\text{VS}} = \text{RF}(\tilde{D}^0 \sim X_0) \qquad \hat{\tau}_1^{\text{VS}} = \text{RF}(\tilde{D}^1 \sim X_1).$$

5. **Final CATE estimation**
The output CATE estimate of the algorithm is the weighted average of $\hat{\tau}_0^{VS}$ and $\hat{\tau}_1^{VS}$, trained on the finally choosen remaining features, where $g \in [0,1]$ is again the weight function. For any feature vector x the predicted CATE is derived as

$$\hat{\tau}^{\text{VS}}(x) = g(x)\hat{\tau}_0^{\text{VS}}(x) + (1 - g(x))\hat{\tau}_1^{\text{VS}}(x).$$

It shall be noted that this algorithm only works for situations where the assignment to control and treatment group is more or less balanced or if there is a huge number of training observations. Otherwise, one would have to find a good rule for variable selection, that considers the unequal amount of data that is used for the fit and importance measure. For example, one could again use the propensity score as weight for the selection of the most unimportant variable in step 3.iii).

To check if the constructed algorithm works for the balanced case, we first have a closer look to the simulation with the true CATE (6.1). We run the variable selection algorithm and plot the test MSE for $K = 0, ..., 10$ on the left hand side of Figure 6.2. Furthermore, the "true" line indicates the performance of the X-learner with given the information about the true dependency to the second base-learners. The plot shows that the automatic variable selection algorithm performs better than the X-learner trained on all of the features for all chosen K values. On the right hand side of Figure 6.2 we can see in percentage if after repeatedly excluding $K = 10$ features, X_1 (red), X_2 (blue) and X_3 (green) are still considered as important variables. Indeed, already for a small training sample size it keeps the truly depending three features X_1, X_2 and X_3 almost every time after it eliminated half of the initially 20 features. Furthermore, if we increase the training sample size we get even more accurate converging closer to 100%.

This shall not be taken for granted, as we can see in Figure 6.3. In this simulation we take the same CATE (6.1) but shift the lower and upper limits of the distributions for all the β such that zero is contained: $\beta_{\mu_0} \sim \text{Unif}([-5,5]^d)$ and $\beta_1, \beta_2, \beta_3 \sim \text{Unif}([0,7])$. The dependency of the CATE gets less affecting, while the parameters can be really close or even equal zero, such that the importance measure more often does not find the truly depending features, and therefore exclude them. The left hand side of the plot indeed shows that the automatic variable selection algorithm performs in this example for $K > 2$ worse than the X-learner trained on all of the features, even though it would perform

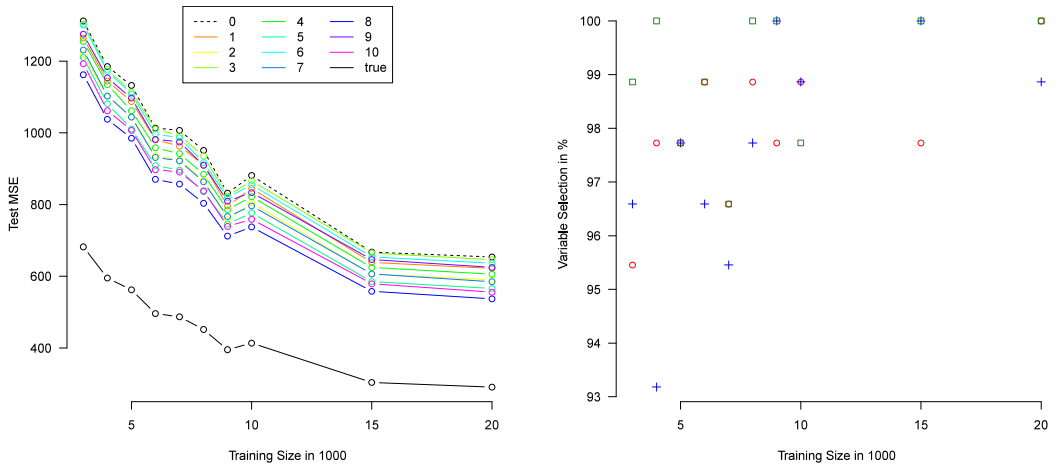

Figure 6.2: *Performance of the variable selection algorithm on the true CATE* (6.1). Left: *Test MSE of the variable selection algorithm for different values of K. The "true" line indicates the performance of the X-learner with given the information about the true dependency to the second base-learners.* Right: *Ratio of containing the features X_1 (red), X_2 (blue) and X_3 (green) for the automatic variable selection algorithm with $K = 10$.*

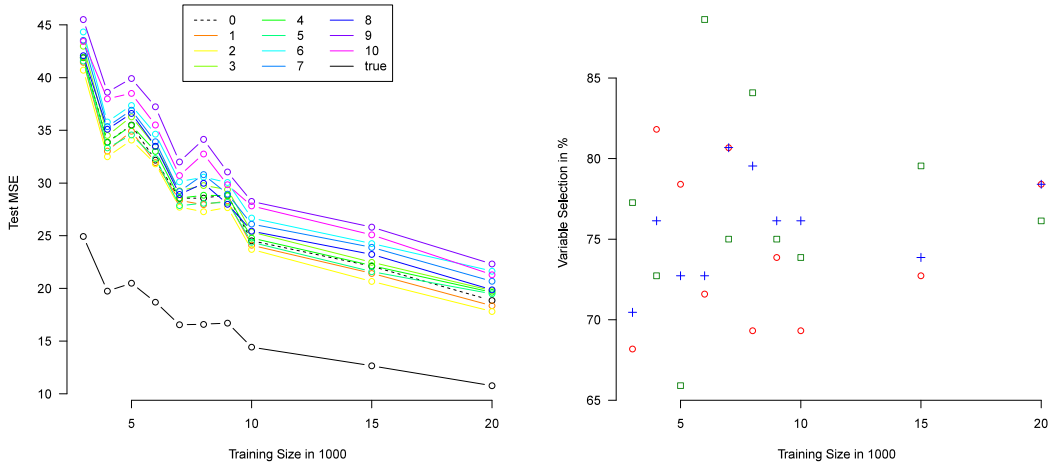

Figure 6.3: *Performance of the variable selection algorithm on the true CATE* (6.1) *with modified parameters.* Left: *Test MSE of the variable selection algorithm for different values of K. The "true" line indicates the performance of the X-learner with given the information about the true dependency to the second base-learners.* Right: *Ratio of containing the features X_1 (red), X_2 (blue) and X_3 (green) for the automatic variable selection algorithm with $K = 10$.*

again much better if it knew the true dependency ("true" line). On the right hand side of Figure 6.3 we can see that the truly depending features X_1, X_2 and X_3 are more often not considered as the most important variables after $K = 10$ exclusions. For small training samples sizes it seems to randomly variate around 77%, while for increasing training size it converges closer to 77%. So without further modifications in this situation the algorithm does not correctly classify the truly depending features as most important ones.

At last we want to have a look how many variables we can eliminate such that we still can feel certain that the true dependent features are not excluded. First we look again at the true CATE (6.1) with its initial parameters and a training sample size of 20'000 observations. We run for each $K = 0, ..., 17$ several times the automatic variable selection algorithm and plot the ratio of how certain the true values are still contained. As we can see on the left hand side of Figure 6.4 for this CATE and 20'000 training observations we can eliminate up to 15 of the 20 features, preserving that all three true CATE-features are at least 97.7% contained under the remaining 5 features. On the other hand, if we consider the true CATE (6.3), also a training sample size of 20'000 observations and try to eliminate $K = 0, ..., 95$ features, we can see that already for small K the algorithm eliminates the wrong features more often. Illustrated by the simulations of before, we could try to increase training sample size to increase the power of the variable selection. On the contrary, it is also possible that this is because the variable importance measure is just not able to correctly classify the true depended five features as we have seen in the example with the true CATE (6.1) and modified parameters. In this case the greedy variable selection algorithm does again not result in a better test MSE, because it is not able to best locally choose the least important feature. Nevertheless, we saw that there certainly exist scenarios, where it can correctly decrease the test MSE while excluding more than half of the initial features.

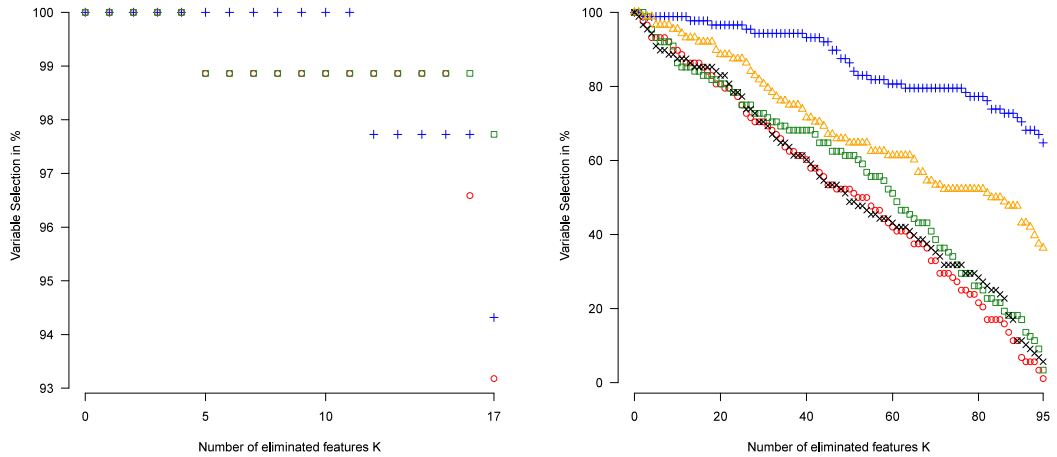

Figure 6.4: *Ratio of containing the true CATE features.* Left: *True CATE* (6.1) *with features* X_1 *(red),* X_2 *(blue) and* X_3 *(green) for the automatic variable selection algorithm with 20'000 training observations.* Right: *True CATE* (6.3) *with features* X_1 *(red),* X_2 *(blue),* X_3 *(green),* X_4 *(orange) and* X_5 *(black) for the automatic variable selection algorithm with 20'000 training observations.*

Chapter 7

Summary

We examined the X-learner invented by Künzel et al. for estimation of heterogeneous treatment effects. Furthermore, we used the X-learner to construct a new variable selection algorithm, to investigate if we can increase the performance in certain situations.

For the first fundamental question

1. *How does the X-learner introduced by Künzel et al. (2017) work? What are the core results of Künzel et al. (2017) comparing the X-learner to other Meta-learners?*

we started with the framework and necessary definitions in chapter 2, where we showed that approximating the CATE is a good estimation for the heterogeneous treatment effect, since the best estimate for the CATE is also the best estimate for the ITE. We than screened the construction of the X-learner algorithm in chapter 3 and in a short digression, constructed a 10-fold-CV X-learner algorithm for real-world data investigation. Next we tried to bring the X-learner closer to the reader, by going through a simple example in chapter 4. Chapter 5 summarizes the essential results of Künzel et al. (2017) of the comparison of the X-learner to two different Meta-learners. We summarized that the X-learner has a really good overall performance compared to the T- and S-learner. Having a good background, we focused on the remaining two fundamental questions, by constructing a new simulation study in chapter 6. For the second fundamental questions,

2. *Considering high-dimensional feature vectors, does the X-learner with random forest as base-learners improve, if we provide information to the second base-learners on which features the true CATE depends?*

we created three different scenarios. We trained for each of them once the X-learner with random forest as base-learners and once with additionally providing the information to the second base-learners on which of the features the true CATE depends. We saw that in this examples, knowing the true dependency is essential for giving the best possible prediction concerning test MSE. Led by this observation, we answered the last fundamental question

3. *Considering high-dimensional feature vectors, can we implement automatic variable selection for the CATE into the X-learner with Random Forest as base-learners?*

by building a greedy variable selection algorithm that locally excludes unimportant features and showed that in certain situations it can correctly decrease the number of features that have to be considered. Nevertheless, we also presented examples where this automatic variable selection algorithm does not classify the right features as most important ones,

and therefore excludes the true dependent features, which led to a bad performance of the algorithm.

In the world of today, where a lot of high-dimensional data is collected, it can be of high interest, having an algorithm that automatically decreases the number of to observe features. In this work we derived one possible approach for the X-learner with random forest, that may motivate a deeper investigation of the X-learner using variable selection.

7.1 Future Work

While in this work we focused on one possible approach using the X-learner with random forest and a greedy-algorithm to investigate empirically some simulation setups, there is a huge potential to expand the investigation to more general frameworks. For example, one could examine the algorithm using bayesian additive regression trees, as it is also a described as a strong algorithm for the base-learners of the X-learner. Furthermore, it would be interesting to search for some theoretical boundaries for the possibility of setups and accuracy of correct elimination. Moreover, it might be interesting to apply and investigate the performance of the algorithm on real-world data sets. While we did not investigate real-world data in this thesis, we provided in chapter 3 a X-learner CV algorithm, that could be used as skeleton for performance evaluation, in the case where one wants to efficiently use a given data set as training and test set.

The X-learner contributes to the investigation of heterogeneous treatment effects. It might be interesting to also examine the X-learner with other statistical analysis methods to gain more insights into causal inference.

Bibliography

Athey, S. and G. Imbens (2016). Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences of the United States of America 113*(27), 7353–60.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Broockman, D. and J. Kalla (2016). Durably reducing transphobia: A field experiment on door-to-door canvassing. *American Association for the Advancement of Science 352*(6282), 220–224.

Foster, J. C., J. M. Taylor, and S. J. Ruberg (2011). Subgroup identification from randomized clinical trial data. *Statistics in Medicine 30*(24), 2867–2880.

Gerber, A. S., D. P. Green, and C. W. Larimer (2008). Social pressure and voter turnout: Evidence from a large-scale field experiment. *American Political Science Review 102(1)*, 33–48.

Hastie, T., G. James, and R. Tibshirani (2013). An introduction to statistical learning. *Springer-Verlag New York Inc.*.

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association 81*(396), 945–960.

Imbens, G. W. (1999). The role of the propensity score in estimating dose-response functions. Working Paper 237, *National Bureau of Economic Research*.

Imbens, G. W. and D. B. Rubin (2015). Causal inference for statistics, social, and biomedical sciences: An introduction. *Cambridge University Press*.

Künzel, S. R., J. S. Sekhon, P. J. Bickel, and B. Yu (2017). Meta-learners for Estimating Heterogeneous Treatment Effects using Machine Learning. *ArXiv: 1706.03461 [math.ST]*.

Nie, X. and S. Wager (2017). Quasi-Oracle Estimation of Heterogeneous Treatment Effects. *ArXiv: 1712.04912 [stat.ML]*.

Pearl, J. (2009). Causality: Models, reasoning and inference. *2nd ed. Cambridge University Press*.

Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology 66*(5), 688–701.

Rubin, D. B. and P. R. Rosenbaum (1983). The Central Role of the Propensity Score in Observational Studies for Causal Effects. *Biometrika 70*(1), 41–55.

Splawa-Neyman, J., D. M. Dabrowska, and T. P. Speed (1990). On the Application of Probability Theory to Agricultural Experiments. Essay on Principles. Section 9. *Statistical Science 5*(4), 465–472.

Wager, S. and S. Athey (2017). Estimation and inference of heterogeneous treatment effects using random forests. *ArXiv: 1510.04342v4 [stat.ME]*.

Appendix A

Pseudo Codes

Algorithm 1 X-learner

procedure X-LEARNER(X, Y, W, g)

$\hat{\mu}_0 = M_0(Y^0 \sim X^0)$ // Estimate response functions

$\hat{\mu}_1 = M_1(Y^1 \sim X^1)$

$\tilde{D}_i^0 = \hat{\mu}_1(X_i^0) - Y_i^0$ // Compute imputed treatment effects

$\tilde{D}_i^1 = Y_i^1 - \hat{\mu}_0(X_i^1)$

$\hat{\tau}_0 = M_2(\tilde{D}^0 \sim X^0)$ // Estimate CATE for treated and control

$\hat{\tau}_1 = M_3(\tilde{D}^1 \sim X^1)$

$\hat{\tau}(x) = g(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x)$ // Average the estimates

Algorithm 2 X-learner K-fold-Cross-Validation

procedure X-LEARNER-CV(X, Y, W, g)

$\hat{\mu}_0 = M_0(Y^0 \sim X^0)$ // Estimate response under control

partition $S = (X^1, Y^1)$ into $S_1, ..., S_K$ // K equally sized independent folds

for k in $\{1, ..., K\}$ **do**

$(X_{test}^1, Y_{test}^1) \leftarrow S_k$ // Creating test and training data

$(X_{train}^1, Y_{train}^1) \leftarrow S_1, ..., S_{k-1}, S_{k+1}, ..., S_K$

$\hat{\mu}_1 = M_1(Y_{train}^1 \sim X_{train}^1)$ // Estimate response under treatment

$\tilde{D}^0 := \hat{\mu}_1(X^0) - Y^0$ // Compute imputed treatment effects

$\tilde{D}^1 := Y_{train}^1 - \hat{\mu}_0(X_{train}^1)$

$\hat{\tau}_0 = M_2(\tilde{D}^0 \sim X^0)$ // Estimate CATE for treated and control

$\hat{\tau}_1 = M_3(\tilde{D}^1 \sim X_{train}^1)$

$\hat{\tau}(x) = g(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x)$ // Average the estimates

$\hat{Y}_{test}^1 \leftarrow \hat{\mu}_0(X_{test}^1) + \hat{\tau}(X_{test}^1)$ // Calculate deviation

$\eta_j = \rho(\hat{Y}_{\text{test}}^1, Y_{\text{test}}^1)$

return $L = \text{mean}(\eta_1, ..., \eta_{10})$

Algorithm 3 X-learner with Random Forest and Variable Selection

procedure X-LEARNER-VS(X, Y, W, g, K)

$\hat{\mu}_0 = \text{RF}(Y^0 \sim X^0)$ // Estimate response functions
$\hat{\mu}_1 = \text{RF}(Y^1 \sim X^1)$

$\tilde{D}_i^0 = \hat{\mu}_1(X_i^0) - Y_i^0$ // Compute imputed treatment effects
$\tilde{D}_i^1 = Y_i^1 - \hat{\mu}_0(X_i^1)$

$(X_0, X_1) = X^{\text{VS}} := X$ // Initially we train on all features

for k in $\{1, ..., K\}$ **do**

$\hat{\tau}_0 = \text{RF}(\tilde{D}^0 \sim X_0)$ // Estimate CATE for treated and control
$\hat{\tau}_1 = \text{RF}(\tilde{D}^1 \sim X_1)$

$f^{\hat{\tau}_0} = \min_{f \in X^{\text{VS}}} \text{imp}^{\hat{\tau}_0}(X^{\text{VS}})$ // Choose least important features separately
$f^{\hat{\tau}_1} = \min_{f \in X^{\text{VS}}} \text{imp}^{\hat{\tau}_1}(X^{\text{VS}})$
$f_{\text{tot}}^{\hat{\tau}_0} = \text{imp}^{\hat{\tau}_0}(f^{\hat{\tau}_0}) + \text{imp}^{\hat{\tau}_1}(f^{\hat{\tau}_0})$
$f_{\text{tot}}^{\hat{\tau}_1} = \text{imp}^{\hat{\tau}_0}(f^{\hat{\tau}_1}) + \text{imp}^{\hat{\tau}_1}(f^{\hat{\tau}_1})$ // Choose least important feature over all
$f_{\text{tot}} = \min_{f \in X^{\text{VS}}} (f_{\text{tot}}^{\hat{\tau}_0}, f_{\text{tot}}^{\hat{\tau}_1})$
$X^{\text{VS}} := X^{\text{VS}} \setminus f_{\text{tot}}$ // Exclude least important feature

$\hat{\tau}_0^{\text{VS}} = \text{RF}(\tilde{D}^0 \sim X_0)$ // Estimate CATE for treated and control
$\hat{\tau}_1^{\text{VS}} = \text{RF}(\tilde{D}^1 \sim X_1)$ // considering only the $d - K$ most important features

$\hat{\tau}(x) = g(x)\hat{\tau}_0^{\text{VS}}(x) + (1 - g(x))\hat{\tau}_1^{\text{VS}}(x)$ // Average the estimates

Appendix B

R Codes

The R codes that were used for all of the discussions and plots can be found on GitHub via https://github.com/rakittam/R-Codes-for-Bachelor-Thesis.git. Because it is nice to have one code snipped fast at hand, we provide the following R code for the top left plot of Figure 6.1.

```
# Balanced treatment, 20-dimensional, simple linear CATE

set.seed(1)

# MSE function
testMSE <- function(eX, d, n.sample, n.test){

  # vector storeing the MSE of the different models
  MSEit <- numeric(2)

  ############################################################
  # Data Preparations
  ############################################################

  # features and noise
  n <- n.sample + n.test
  S <- genPositiveDefMat("c-vine",dim=d)
  X <- rmnorm(n,varcov=S$Sigma)

  noise0 <- rnorm(n,0,1)
  noise1 <- rnorm(n,0,1)

  # response functions
  betaC <- runif(n = 3, min = 20, max = 30)
  CATE <- X[,1:3] %*% betaC # true CATE
  beta <- runif(n = d, min = 5, max = 10)
  mu0 <- X %*% beta # true response function under control
  mu1 <- mu0 + CATE # true response function under treatment

  # potential outcomes
  Y0 <- mu0 + noise0
  Y1 <- mu1 + noise1

  # data preparations
  data <- data.frame(X,Y0,Y1, CATE)
  index.train <- 1:n.sample
  data.train <- data[index.train,]
  data.test <- data[-index.train,]
  index <- sample(1:n.sample, n.sample*eX)
  data.contr <- data.train[-index,]
  data.treat <- data.train[index,]


```

```r
  ############################################################
  # X-Learner RF
  ############################################################

  # estimated response function under control and treatment
  mu0.hat.RF <- randomForest(data.contr$Y0~., data = data.contr[,1:d])
  mu1.hat.RF <- randomForest(data.treat$Y1~., data = data.treat[,1:d])

  # imputed treatment effects
  mu0X1.pred.RF <- predict(mu0.hat.RF, newdata = data.treat)
  mu1X0.pred.RF <- predict(mu1.hat.RF, newdata = data.contr)
  D0.RF <- mu1X0.pred.RF - data.contr$Y0
  D1.RF <- data.treat$Y1 - mu0X1.pred.RF

  # CATE estimates
  tau0.hat.RF <- randomForest(D0.RF~., data = data.contr[,1:d])
  tau1.hat.RF <- randomForest(D1.RF~., data = data.treat[,1:d])

  # evaluate CATE on test data
  tau0.pred.RF <- predict(tau0.hat.RF, newdata = data.test)
  tau1.pred.RF <- predict(tau1.hat.RF, newdata = data.test)

  CATE.hat.RF <- eX * tau0.pred.RF + (1-eX) * tau1.pred.RF
  CATE.true <- data.test$CATE

  MSEit[1] <- mean((CATE.hat.RF - CATE.true)^2)

  ############################################################
  # X-Learner RF with Variable Selection
  ############################################################

  # CATE estimates
  tau0.hat.RF <- randomForest(D0.RF~., data = data.contr[,1:3])
  tau1.hat.RF <- randomForest(D1.RF~., data = data.treat[,1:3])

  # evaluate CATE on test data
  tau0.pred.RF <- predict(tau0.hat.RF, newdata = data.test)
  tau1.pred.RF <- predict(tau1.hat.RF, newdata = data.test)

  CATE.hat.RF <- eX * tau0.pred.RF + (1-eX) * tau1.pred.RF

  MSEit[2] <- mean((CATE.hat.RF - CATE.true)^2)

  ############################################################
  # Return the vector of MSE of all models
  ############################################################

  return(MSEit)

}

############################################################
# Initialisation
############################################################

library(foreach)
library(doParallel)

eX <- 0.5 # propensity score
d <- 20 # number of covariates X_i
n.test <- 1000 # factor for number of test observations
n.factor <- 1000 # number of train observations
n.tausend <- c(3, 4, 5, 6, 7, 8, 9, 10, 15, 20)

# second argument is number of methods
meanMSE <- matrix(NA, ncol = length(n.tausend), nrow = 2)
old <- Sys.time() # get start time
```

```
for (i in 1:length(n.tausend)) {

  registerDoParallel(88) # set number of cores for parallel computation

  n.sample ← n.factor*n.tausend[i] # number of observations

  # repeat for certain training size
  MSE ← foreach(j=1:176,
                .packages = c("clusterGeneration", "mnormt", "randomForest"),
                .combine = cbind) %dopar% {
    testMSE(eX, d, n.sample, n.test)
  }

  meanMSE[,i] ← apply(MSE, 1, mean)

  new ← Sys.time() - old # calculate difference
  print(i)
  print(meanMSE[,i])
  print(new) # print in nice format

}

# plot results
par(mfrow=c(1,1))
plot(n.tausend, meanMSE[1,], type="b", col="red") #RF
plot(n.tausend, meanMSE[2,], type="b", col="blue") #RF with variable selection

par(mfrow=c(1,1))
plot(n.tausend, meanMSE[1,], type="b", bty="n",las=1, ann=F, col="red",
     xaxt = 'n', ylim = c(min(meanMSE),max(meanMSE))) #RF
lines(n.tausend, meanMSE[2,], type="b", col="blue") #RF with variable selection
mtext("Test MSE", side=2, line=3, las=3)

par(mfrow=c(1,1))
plot(n.tausend, meanMSE[1,], type="b", bty="n",las=1, ann=F, col="red",
     ylim = c(min(meanMSE),max(meanMSE))) #RF
lines(n.tausend, meanMSE[2,], type="b", col="blue") #RF with variable selection
mtext("Test MSE", side=2, line=3, las=3)

par(mfrow=c(1,1))
plot(n.tausend, meanMSE[1,], type="b", bty="n",las=1, ann=F, col="red",
     ylim = c(min(meanMSE),max(meanMSE))) #RF
lines(n.tausend, meanMSE[2,], type="b", col="blue") #RF with variable selection
mtext("Test MSE", side=2, line=3, las=3)
mtext("Training Size in 1000", side=1, line=3, las=1)
```

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**

Ich bestätige mit meiner Unterschrift:

- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.