
THE UNIVERSITY OF ARIZONA
University Libraries

UA CAMPUS
REPOSITORY

Using Reinforcement Learning to Personalize Dosing Strategies in a Simulated Cancer Trial with High Dimensional Data

Item Type	text; Electronic Thesis
Authors	Humphrey, Kyle
Publisher	The University of Arizona.
Rights	Copyright © is held by the author. Digital access to this material is made possible by the University Libraries, University of Arizona. Further transmission, reproduction or presentation (such as public display or performance) of protected items is prohibited except with permission of the author.
Download date	08/06/2021 01:48:25
Link to Item	http://hdl.handle.net/10150/625341

USING REINFORCEMENT LEARNING TO PERSONALIZE DOSING STRATEGIES IN A SIMULATED CANCER TRIAL WITH HIGH DIMENSIONAL DATA

by

Kyle Humphrey

Copyright © Kyle Humphrey 2017

A thesis submitted to the faculty of the

MEL AND ENID ZUCKERMAN COLLEGE OF PUBLIC HEALTH

In partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE
WITH A MAJOR IN BIOSTATISTICS

In the Graduate College

THE UNIVERSITY OF ARIZONA

2017

STATEMENT BY AUTHOR

The thesis titled "Using reinforcement learning to personalize dosing strategies in a simulated cancer trial with high dimensional data" prepared by Kyle Humphrey has been submitted in partial fulfillment of requirements for a masters degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: *Kyle Humphrey*

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Jin Zhou, Ph.D.
Assistant Professor of Biostatistics

April 26, 2017
Date

Table of Contents

List of Figures

List of Tables

Abstract

In a simulation of an advanced generic cancer trial, I use Q-learning, a reinforcement learning algorithm, to develop dynamic treatment regimes for a continuous treatment, the dose of a single drug. Selected dynamic treatment regimes are tailored to time-varying patient characteristics and to patient subgroups with differential treatment effects. This approach allows estimation of optimal dynamic treatment regimes without a model of the disease process or a priori hypotheses about subgroup membership. Using observed patient characteristics and outcomes from the simulated trial, I estimate Q-functions based on 1) a single regression tree grown by the Classification And Regression Trees (CART) method, 2) random forests, and 3) a slightly modified version of Multivariate Adaptive Regression Splines (MARS). I then compare the survival times of an independent group of simulated patients under treatment regimes estimated using Q-learning with each of the three methods, 10 constant dose regimes, and the best possible treatment regime chosen using a brute force search over all possible treatment regimes with complete knowledge of disease processes and their effects on survival. I also make these comparisons in scenarios with and without spurious high dimensional covariates and with and without patient subgroups with differential treatment effects. Treatment regimes estimated using Q-learning with MARS and random forests greatly increased survival times when compared to the constant dose regimes, but were still considerably lower than the best possible dose regime. Q-learning with a single regression tree did not outperform the constant dose regimes. These results hold across high dimensional and subgroup scenarios. While the MARS method employed produces much more interpretable models than random forests, and therefore has more promise for patient subgroup identification, I show that it is also more sensitive to variations in training data.

1 Introduction

1.1 Personalized medicine

Patients often show significant differences in their responses to treatment. Adverse reactions to drugs alone contribute to significant public health burden despite these drugs being shown to be beneficial (on average) in clinical trials (Pirmohamed et al. 2004). Not only are there subpopulations for whom treatments are especially deleterious, some groups have a more robust response to certain treatments. Significant gains in public health can therefore be made from identifying these subgroups and

the corresponding treatments which are most effective for them, rather than relying on identification of only the most effective treatment overall.

This is the idea of *personalized medicine*[1], which is the commonsensical notion that the best treatment for a given patient depends upon that patients' characteristics (Schleidgen et al. 2013). The term is most often used today when the patient characteristics are biomarkers, perhaps whether metastatic breast cancer cells overexpress human epidermal growth factor receptor 2 (are HER2-positive), and these biomarkers are used to determine the best treatment for a particular individual, and when to use it. In the case of the patients with HER2-positive cells, they should receive a different treatment (trastuzumab) from those with HER2-negative cells (Baselga et al. 2006). However, the general idea goes back at least as far as Hippocrates[2] and is implemented in principle whenever a physician tries to inform a treatment decision by determining whether the cause of an infection is bacterial or viral.

While personalized medicine in principle has a long history and shows considerable promise, there are numerous challenges in identifying these patient subgroups and the optimal treatment for each. Chief among them is the question of how to select from what is often a multitude of plausible subgroups. Narrowing down these subgroups is hindered by disease processes which are usually quite complicated and not well understood. Furthermore, optimal treatments can change over time with changing disease. This is especially true of chronic diseases, however little evidence is available to inform how to tailor treatments over time and using the information from previous treatments.

1.2 Statistical methods to personalize medicine

1.2.1 Traditional methods

Statistically, the personalization of medicine involves the identification of patient subgroups which respond to treatment differently. Traditionally, this is accomplished by investigation of (often ad hoc) treatment by covariate interactions, where different values of a covariate identify subgroups (Byar 1985). Using the HER2 breast cancer example above, we would encapsulate our knowledge of a patient's HER2 status into a binary covariate which equaled one if her tumor was HER2 postitive or zero if her tumor was HER2 negative. We would then statistically test if the coefficient on an interaction term with this indicator and treatment is significantly different from zero.

[1]Personalized medicine is also called precision medicine, stratified medicine, and P4 medicine

[2]Hippocrates is attributed with saying "It is more important to know what sort of person has a disease than to know what sort of disease a person has." (Fischer et al. 2015)

While straightforward, this approach has several problems. First, which subgroups to examine is often a subjective judgement made by the researcher (as is which cutoff to use to create subgroups from continuous covariates). Second, since the number of plausible subgroups can be quite large in typical problems, the number of interaction terms required must also be large. In addition to issues with multiple comparison, this requires much larger sample sizes (i.e. it suffers from the curse of dimensionality).

1.2.2 Algorithms for detecting interactions

In response to the above problems with the traditional approach, several novel algorithms to detect interactions have been developed. Many use recursive partitioning (the basic process of which is described below) with a splitting criterion based on the degree of interaction (Zeileis, Hothorn, and Hornik 2008; Su et al. 2009; Lipkovich et al. 2011; Dusseldorp and Van Mechelen 2014). These (and models based on recursive partitioning in general) have the advantage of being easy to interpret.

Other methods use a two-step process (Cai et al. 2011; L. Zhao et al. 2013; Foster, Taylor, and Ruberg 2011). The first step is to estimate the differential treatment effect of each participant and use this to develop a score to group patients. These scores are then used as a response in the second step to estimate the mean treatment difference between groups. These methods require a parametric or semi-parametric model in the first step which is subject to misspecification.

A third class of methods maximize mean responses across patients and subgroups either by using penalized regression (Qian and Murphy 2011), or by treating the identification of subgroups as a classification problem with each participant weighted by their outcome (Yingqi Zhao et al. 2012; Zhang et al. 2012).

These and other methods are promising but none address how to select the optimal value of a continuous treatment, so it is not clear how to extend them to address our goal of determining optimal dosing strategies. Perhaps more importantly, none of the methods described above directly consider more than a single stage of treatment. In practice treatments are often changed over time based off of patient response and other factors, particularly in the management of chronic disease. To truly personalize medicine, therefore, it is important to determine how to adapt treatment over time to changing situations.

1.3 Overview of paper contents

In this paper, we will see how the obstacles to identifying optimal treatment regimes for a given individual can be addressed using Q-learning, a algorithm from rein-

forcement learning, in conjunction with statistical learning models. We will see how these methods can dramatically increase survival times in a simulated cancer clinical trial where doses of a single drug are chosen at each month of treatment, without a model of the disease process or a priori hypotheses about subgroup membership. The simulation was inspired by Yufan Zhao, Kosorok, and Zeng (2009) who achieved similar results using extremely randomized trees and support vector regression to estimate Q-functions. We will replicate and extend their results to more scenarios (e.g. situations with many meaningless covariates and patient subgroups), contrast their results with results obtained using more interpretable models, and compare all of the above to the best possible treatment sequence chosen with complete knowledge of disease processes and their effects on survival.

We will first briefly overview the basic elements of reinforcement learning, reinforcement learning's relevance for personalized medicine, and Q-learning (C. Watkins 1989) specifically. We will then discuss the supervised learning methods that I used in conjunction with Q-learning to estimate optimal dosing strategies: regression tress constructed using the classification and regression trees (CART) method (Breiman et al. 1984), Multivariate adaptive regression splines (MARS–J. H. Friedman 1991), and random forests (RF–Breiman 2001), with random forests serving as a less interpretable comparison similar to the extremely randomized trees used by Yufan Zhao, Kosorok, and Zeng (2009). Each technique will be illustrated with a simple example taken from the simulation. We will then discuss the details of the simulation setup and its results.

For our purposes, trees (and MARS, which shares many of their desirable properties) have several particularly appealing features. First, the tree and MARS model building procedures conduct covariate selection as part of the model building process. This will help us screen non-informative variables. Second, we need not specify the form of the relationship between the covariates and the outcome, or limit ourselves to linear or additive relationships. This will allow us to choose optimal actions without prior knowledge of disease mechanisms, which are often complicated and not well understood. Third, since single trees and MARS models are quite interpretable, we can in principle inspect them to see which variables interact with treatment and thus identify patient subgroups.

2 Reinforcement learning

2.1 Basic elements and basic process

Reinforcement learning is a branch of machine learning distinct from the two main branches more familiar to statisticians, supervised and unsupervised learning (Sutton and Barto 2016)[3]. Reinforcement learning in general is about how to make good sequences of decisions. It has the following elements:

A learning *agent*, who has goals related to its *environment*, loosely defined as everything outside the agent's direct control. Features of the environment are represented to the agent through *states*. Given a state representation, an agent can take an *action* to affect the environment in order to achieve some goal. A goal is defined to the agent in such a way that maximizing a scalar quantity called the *reward* would achieve the goal. Given a state representation, an agent follows a *policy*, roughly a guide to which actions to take (or which ones to favor) given a state signal. An optimal policy therefore, is a policy that achieves the largest reward over the long run.

The basic process of reinforcement learning involves a learning agent being presented with states, trying a sequence of actions, recording the consequences (rewards) of those actions in each state, estimating the relationship between the actions and their consequences, and choosing a next action that leads to the best consequences (the action that maximizes the reward, so far as the agent can tell). This is where the "reinforcement" in reinforcement learning comes from: actions with favorable consequences (large rewards) tend to be repeated.

2.2 Relationship to supervised and unsupervised learning

Supervised learning involves a learning agent being given a list of correct example actions to take in given situations, with the goal being that the agent then extrapolates the correct example behavior to new situations. Unsupervised learning involves finding hidden structure in collections of example actions and situations where the correct action is unknown. In reinforcement learning problems, agents do not (need to) receive direct instruction regarding which action they should take, instead they can learn which actions are best by trying them out. Further, the goal in reinforcement learning problems is to maximize a reward signal, not find hidden structure (though finding hidden structure may be useful to this end).

[3]Techniques from both supervised and unsupervised learning can, however, be useful in reinforcement learning problems, as we will see for supervised learning methods.

Also unlike other machine learning problems, reinforcement learning problems are closed-loop: the actions of the agent affect the opportunities open to the agent later on, hence consequences of actions in reinforcement learning problems can manifest not only in the next opportunity, but in all subsequent opportunities.

2.3 Connection to personalized medicine

We can think of a personalized treatment as a decision rule, or a *treatment regime*[4] that provides the optimal action (the best treatment to give) given a patients' state (the patient's characteristics–in this context, patients are the important aspect of the environment, about whom we have goals and thus in terms of whom rewards are defined). Since for many diseases, patients often receive different stages of treatments over time, it is useful to generalize a single stage treatment regime into a *dynamic treatment regime* (DTR) or in the terminology of reinforcement learning, a *treatment policy*. A dynamic treatment regime generalizes of the ideas of personalized medicine by dictating treatments at each step of a sequence of possible treatment assignments based on the changing (dynamic) states of each individual patient. In order to determine which dynamic treatment regime is best (optimal) we will employ Q-learning, the most popular method for estimating optimal dynamic treatment regimes.

3 Q-learning

Following the notation in Chakraborty and Moodie (2013), suppose patients are recruited into a clinical trial and covariates (states) are measured before the trial begins and recorded in a vector, O_1. These covariates will be referred to as the patient history at stage 1: $H_1 \equiv O_1$, which may include the baseline level of the outcome. A first treatment, A_1, is randomly assigned to each patient and each patient is measured again for the same covariates as before, the new values of which are recorded in O_2, and a reward is received, R_2. As mentioned above, this reward can be any scalar, which if maximized, would achieve the goals of the study. Most often rewards are simply set to be a continuous outcome of interest (e.g. kg of weight lost in a weight loss study). The treatment and second observation of covariates (which may include the first stage outcome and/or rewards) are incorporated into each patients' history at stage 2: $H_2 \equiv (O_1, A_1, O_2)$.

A second treatment, A_2 is then assigned to each patient at random (potentially depending on H_2) and is followed by another outcome/reward, R_3. This process can

[4]Sometimes also called an individualized treatment rule

be repeated through as many stages as desired. Note that this scheme can easily be modified for a single terminal outcome by setting $R_2, \ldots, R_K = 0$ for a sequence of K stages (as we do in the simulation below).

The basic process described above is a Markov decision process. Markov decision processes are Markov chains with the addition of actions (adding choices) and rewards (motivation/goals). In a Markov decision process the agent and environment interact over a series of stages. At a stage k, the agent is presented with a state $S_k \in \mathcal{S}$ from the set of possible states, and chooses an action $A_k \in \mathcal{A}(S_k)$ from the set of possible actions available in that state. At the next time step, the agent receives a numerical reward, R_{k+1} as a consequence of the action chosen and is presented with a new state S_{k+1}. The states in a Markov decision process have the Markov property, which means that they summarize all the information from previous steps without a reduction in what's relevant to make the best action decision. Reinforcement learning methods try to determine how the agent should change its policy in response to experience (available data) in order to maximize reward. Hence "solving" a reinforcement learning problem means finding a policy that achieves the most reward over the long run.

The expected return (the sum of rewards) for an agent starting in state s, taking action a, and then following policy π thereafter, called the action-value function for policy π (and in the context of Q-learning, a Q-function), is denoted:

$$Q_k^{\pi}(s,a) \doteq \mathrm{E}_{\pi}\left[\sum_{j=k}^{K} R_{j+1} \mid S_k = s, A_k = a\right] \tag{1}$$

All action-value (Q) functions satisfy a recursive relationship called the Bellman

equation for Q_k^π:

$$\begin{aligned}
Q_k^\pi(s,a) &= \mathrm{E}_\pi\left[\sum_{j=k}^{K} R_{j+1} \mid S_k = s, A_k = a\right] \\
&= \mathrm{E}_\pi\left[R_{k+1} + \sum_{j=k+1}^{K} R_{j+1} \mid S_k = s, A_k = a\right] \\
&= \mathrm{E}_\pi\left[R_{k+1} \mid S_k = s, A_k = a\right] + \sum_{s'}\left[\Pr(S_{k+1} = s' | S_k = s, \pi(S_k = s)) \sum_{j=k+1}^{K} R_{j+1}\right] \\
&= \mathrm{E}_\pi\left[R_{k+1} \mid S_k = s, A_k = a\right] + \sum_{s'}\left[\Pr(S_{k+1} = s' | S_k = s, \pi(S_k = s)) V_{k+1}^\pi(s')\right]
\end{aligned}$$

or

$$\begin{aligned}
Q_k^\pi(s,a) &= \mathrm{E}_{s'\sim\Pr(s'|s,a)}\left[R_{k+1} + V_{k+1}^\pi(s')\right] \\
&= \mathrm{E}_{s'\sim\Pr(s'|s,a)}\left\{R_{k+1} + \mathrm{E}_{a'\sim\pi}[Q_{k+1}^\pi(s',a')]\right\}, \qquad (2)
\end{aligned}$$

where

$$\begin{aligned}
V_k^\pi(s) &\doteq \mathrm{E}_\pi\left[\sum_{j=k}^{K} R_{j+1} \mid S_k = s\right] \qquad (3) \\
&= \mathrm{E}_{a\sim\pi}\left[Q_k^\pi(s,a)\right]. \qquad (4)
\end{aligned}$$

With the Q-function for stage k in hand, one can obtain the optimal Q-function, which is the action-value function obtained when following the optimal policy π^*, the policy that leads to the maximum cumulative rewards. This is done simply by finding the action a that maximizes $Q_k^\pi(s,a)$ for each stage k:

$$Q_k^*(s,a) = \max_a Q_k^\pi(s,a) \qquad (5)$$

The optimal action at stage k is then the action which maximizes the optimal Q-function

$$a_k^* = \operatorname*{argmax}_a Q_k^*(s,a) \qquad (6)$$

It turns out that there is always at least one optimal policy for every Markov decision process (Sutton and Barto 2016).

Optimal Q-functions satisfy a recursive relationship called the Bellman optimality equation for Q^*, which describes the relationship between adjacent stage optimal Q-

functions:

$$Q_k^*(s,a) = \mathrm{E}_{s' \sim \Pr(s'|s,a)} \left\{ R_{k+1} + \max_{a'} [Q_{k+1}^*(s', a')] \right\}. \tag{7}$$

With the Bellman optimality equation, optimal Q-functions, and the state-transition probabilities, $\Pr(S_{k+1} = s' \mid S_k = s, A_k = a)$, we can use value iteration from dynamic programming to iterate through the stages backwards from the last stage to the first stage, obtaining at each stage the optimal Q-function and hence the optimal action. The sequence of optimal actions obtained is an optimal policy π^*. In reinforcement learning problems, however, this dynamic programming approach equation won't work because the state-transition probabilities are unknown.

Q-learning overcomes this limitation by directly estimating the optimal Q-functions, for each stage, starting with the last stage. Dynamic programming is then used iterating backwards through the stages, obtaining estimates of each Q-function, $\hat{Q}_k^*(s,a)$, at each step. Each $\hat{Q}_k^*(s,a)$ can then be used to estimate optimal actions, $\hat{A}^*$ (shown below) for each stage k. Since Q-learning does not require a model of the state-transition probabilities it is described as being "model-free".

The estimated optimal Q-functions from Q-learning have been shown to converge to the true optimal action-value functions with probability one for finite Markov decision processes (MDPs with a finite number of states and actions), given that actions are repeatedly sampled in all states (C. J. C. H. Watkins and Dayan 1992).

When there are many states and/or actions (as there are in most problems) $Q_k^*(s,a)$ can be approximated using functions (as we do below), the parameters of which are adjusted to better match new observations over time.

3.1 Estimating Q-functions and optimal DTRs

In our context, the full recursive form of Q-learning can be written as:

$$\hat{Q}_k^\pi(A_k, H_k) = \mathrm{E}_\pi[r_{k+1} + \max_{a_{k+1}} \hat{Q}_{k+1}^\pi(A_{k+1}, H_{k+1}) \mid A_k, H_k], \quad k = 1, \ldots, K \tag{8}$$

The optimal treatment for each participant $i \in 1, \ldots, n$ at each stage k is the treatment which maximizes the corresponding stage's Q-function:

$$\hat{A}_{ik}^* = \underset{a_{ik}}{\operatorname{argmax}} \hat{Q}_k^\pi(A_{ik}, H_{ik}), \;\; i = 1, \ldots, n; \;\; k = 1, \ldots, K \tag{9}$$

Since Q-functions are conditional expectations, standard regression techniques are applicable and hence the most popular method for estimating Q-functions in the health sciences is ordinary least squares (Chakraborty and Murphy 2014).

That being said, any modeling method can be used, and as mentioned above recently Yufan Zhao, Kosorok, and Zeng (2009) have used extremely randomized trees and support vector regression to fit Q-functions. Note that Q-functions may differ across stages of treatment. The steps of Q-learning are as follows:

1. Set the Q-function after the final stage to be 0, $\hat{Q}^{\pi}_{K+1} \equiv 0$ (though any value will do).

2. Create a pseudo-outcome for the last stage, K, by adding the rewards actually received from that stage to the estimated rewards that would have been obtained if the optimal dynamic treatment regime was followed after stage K:

$$\hat{r}_{K+1} = r_{K+1} + \max_{a_{K+1}} \hat{Q}^{\pi}_{K+1}(A_{K+1}, H_{K+1}) \tag{10}$$

$$\hat{r}_{K+1} = r_{K+1} \tag{11}$$

3. Estimate the preceding stage Q-function, Q^{π}_{K-1}, using the pseudo-outcome just created, $\hat{r}_K$, as the outcome with all aspects of patient history at that stage, H_K, desired as covariates, in addition to (if fitting a linear regression model) treatment by covariate interactions for covariates in H_K thought to be indicative of subgroups with differential treatment effects:

$$\hat{Q}^{\pi}_{K}(A_K, H_K) = \mathrm{E}[\hat{r}_{K+1} \mid A_K, H_K] \tag{12}$$

4. Create the pseudo-outcome for the previous stage $K-1$, again as if the optimal treatment regime will be followed after $K-1$:

$$\hat{r}_{K+1} = r_{K+1} + \max_{a_K} \hat{Q}^{\pi}_{K}(A_K, H_K) \tag{13}$$

5. Repeat until you've estimated the Q-function for the first stage

It is important to note that in practice, the process described above should be repeated through a process called policy iteration. New patients would be recruited and the estimated optimal dynamic treatment regime estimated from the previous trial would be used to assign treatments to each patient. Then, an updated estimate of the optimal treatment regime would be obtained by using Q-learning on these new observations. This updated optimal dynamic treatment regime would then be used to assign treatments to a new group of patients, and the process would be repeated again. After each new trial, the updated optimal dynamic treatment regime gets closer to the true optimal dynamic treatment.

Algorithm 1: Q-learning

initialize $\hat{Q}^{\pi}_{K+1}$ arbitrarily:
$\hat{Q}^{\pi}_{K+1} \leftarrow 0$

for k *in* $K, \ldots, 1$ **do**

Estimate the rewards that would occur if the optimal policy was followed after k:
$\hat{r}_k \leftarrow r_k + \max_{A_{k+1}} \hat{Q}^{\pi}_{k+1}(A_{k+1}, H_{k+1})$

Estimate the Q-function, $Q^{\pi}_k(A_k, H_k)$, for the preceding stage using this reward as the outcome (e.g. through ordinary least squares).

Estimate the optimal treatment(s) for each patient at stage k:
$\hat{A}^*_k \leftarrow \text{argmax}_{A_k} \hat{Q}_k(A_k, H_k)$

end

4 Regression trees using CART

A decision tree is a graphical representation of nested `if-then` statements, with each "decision" being based on whether the `if` statement is true or false. A regression tree is a kind of decision tree where each `if-then` statement splits the data (or subset of the data created by a previous split) according to the value of a single covariate at a time. Since an `if` statement creates two possible outcomes (`TRUE` or `FALSE`), each split is binary and results in two *daughter nodes*, two subsets of the data created according to the value of a particular covariate. Once the desired number of splits is performed, each *terminal node* or *leaf*, a node from which no further splits are made, is assigned an outcome (typically the mean outcome for each observation in that node). Unlike trees in nature, decision trees grow from the root node downward, as shown in the illustrative example below. Since in reinforcement learning, rewards are defined to be continuous, we focus on regression trees constructed using the CART procedure of Breiman et al. (1984).

4.1 Tree growing

Our overarching goal in tree construction is to create nodes as homogeneous in the outcome as possible. This is because we will predict a single outcome for all observations in the terminal nodes, meaning nodes heterogeneous in the outcome will yield poor predictions. To achieve this we must determine:

1. What covariate to use to define a split and what value of the covariate to split on

2. When to stop splitting

3. How to assign an outcome to each terminal node

We begin with all of the data (the *root node*) and evaluate the sum of squared errors that result from splitting based upon each unique value of each covariate:

$$SSE = \sum_{i \in \text{node}_L} (y_i - \bar{y}_L)^2 + \sum_{i \in \text{node}_R} (y_i - \bar{y}_R)^2 \tag{14}$$

Where $\bar{y}_L$ is the average outcome in the left node, node_L and $\bar{y}_R$ is the average outcome in the right node, node_R. We choose the split that minimizes the sum of squares error (SSE). This process is then repeated in both of the left and right nodes that result from the split. This feature of recursive splitting, or partitioning, of the data is why this method is also known as *recursive partitioning*. Typically we continue splitting in this manner until the number of observations in a node falls below a threshold (e.g. 20 observations), and assign each terminal node the average outcome of training observations in that node.

Algorithm 2: Regression tree growing algorithm

```
while stopping criteria not met do
    for each terminal node do
        for each X_j ∈ X do
            for each possible split of X_j do
                compute change in SSE resulting from split
            end
        end
        if reduction in SSE from split largest then
            split node
        end
        add newly created nodes to set of terminal nodes
    end
end
```

4.2 Tree pruning

The large tree resulting from the tree growing phase, T_0, typically overfits the data. To combat this, we prune back T_0 using a technique called cost complexity pruning.

The goal in cost-complexity pruning is to find, for each value of a complexity parameter $\alpha \geq 0$, the subtree $T \subset T_0$ created by collapsing any of T_0's internal (non-terminal) nodes, which minimizes the cost complexity criterion:

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{x_i \in \text{node}_m} (y_i - \bar{y}_m)^2 + \alpha|T| \tag{15}$$

where y_i is the outcome value for an observation in node m, $\bar{y}_m$ is the average outcome in node m, and $|T|$ is the number of terminal nodes in subtree T. One can show that there is a unique subtree T_α that minimizes $C_\alpha(T)$ for each given α (Hastie, Tibshirani, and J. Friedman 2009).

In order to find T_α for each α, we use weakest link pruning. As the name implies, in weakest link pruning, we test out collapsing each internal node, choosing to actually collapse the the weakest link: the node that, if collapsed, produces the smallest increase in $\sum_{m=1}^{|T|} \sum_{x_i \in \text{node}_m} (y_i - \bar{y}_m)^2$, the sum of squares of the whole tree. We continue in this fashion until we've returned to the root node. This sequence contains each T_α.

With the T_αs in hand, we choose the α (and hence T_α) that minimizes the sum of square errors produced using cross validation. Alternatively, we can use choose the largest α (smallest T_α) which is within one standard error of the minimum cross validated error.

4.3 Variable importance

Variable importance in trees can be measured by summing the overall reduction in the optimization criteria for each prediction (Breiman et al. 1984). In regression trees, the optimization criteria is typically SSE, so we could sum the reductions in SSE from each splits made based on a particular covariate, repeating the process for every covariate.

4.4 Advantages and disadvantages

The main advantage of regression trees is their easy interpretation, even by non-experts, largely due to the convenient graphical tree representation. CART also intrinsically conducts feature selection as part of the model building process and

Algorithm 3: Regression tree pruning algorithm

```
for each α > 0 do
    while T ⊃ root node do
        compute change in SSE from collapsing each internal node in turn
        collapse node that produces the smallest increase in SSE (weakest link),
         save resulting tree
    end
    for each saved tree do
        compute cross validation error
    end
    Choose the saved tree with the smallest cross validation error (T_α̂)
end
Pick the T_α̂ with the smallest cross validation error as final tree.
Or pick the smallest T_α̂ within one standard error of the T_α̂ with the smallest
 cross validation error
```

easy handles many types of covariates without the need for pre-processing, creating dummy variables, or even specification of the form of the relationship between the covariates and the outcome. There are also methods to handle missing data specific to trees, see Hastie, Tibshirani, and J. Friedman (2009) for details.

Trees however, tend to have lower predictive performance compared to other methods particularly in the regression setting. Part of the reason is that trees tend to be sensitive to changes in data, which is to say they have high variance (though ensemble methods like random forest, discussed below, exploit this property to achieve better performance). In the regression setting in particular, trees are hindered by a lack of smoothness and difficulty in capturing additive structures. Multivariate adaptive regression splines (MARS), which we will discuss next, can be viewed as a modification of the CART technique to overcome these issues.

4.5 Illustrative example

Let's examine CART, MARS, and random forest (below) using a simple sub-problem from the simulation described in detail below. Suppose we want to predict the change in the mass of the ith cancer patients' tumor from its initial mass to the mass after one month of treatment, $Y_i \equiv M_{i1} - M_{i0}$, given the dose of a drug, D_i, which they were (randomly) assigned. The true relationship is given by

$$Y_i = 0.75 - 1.2D_i + \epsilon_i, \quad i = 1, \ldots, 50 \tag{16}$$

Where $\epsilon_i \sim N(0, 0.05)$. 50 patients were simulated with random starting tumor masses sampled from a uniform distribution from 1 to 2, $M_0 \overset{iid}{\sim} \text{Unif}(1, 2)$, and doses were sampled from a uniform distribution between 0 (none) and 1 (the maximum tolerable dose), $D \overset{iid}{\sim} \text{Unif}(0, 1)$. 50 additional patients were simulated in the same fashion as test data.

In this example we used stopping criteria of: (1) terminal nodes had to have at least 5 observations, and (2) nodes with fewer than 15 observations were not split. For pruning, we follow the "one standard error" rule, where we choose the smallest tree within one standard error of the tree with the minimum cross validation error.

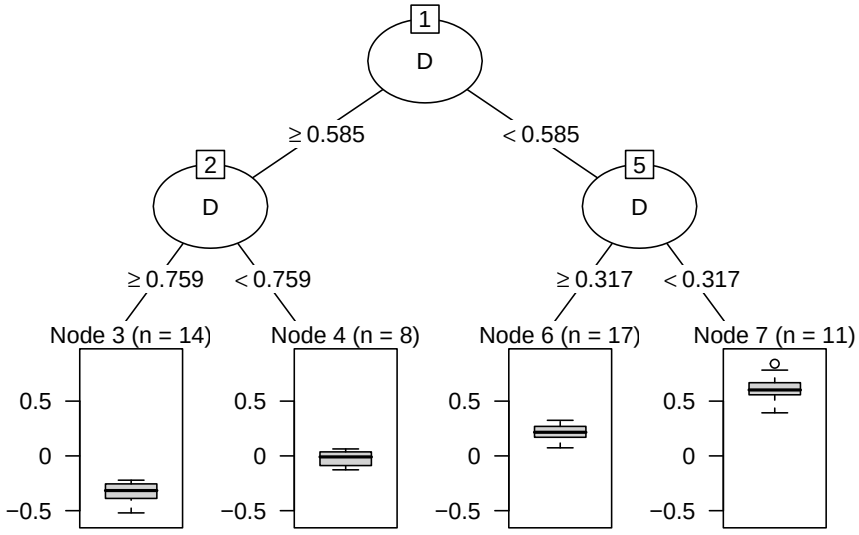

Figure 1: Regression tree fit to example data. For each split, the variable split upon is shown as a circle representing an internal node (with a numbered box giving the node a number). The values of the splitting variable that define resulting node membership are shown breaking the line connecting nodes (e.g. observations with $D \geq 0.585$ end up in node 2, while observations with $D < 0.585$ end up in node 5). Box plots show the distribution of responses in each terminal node (nodes 3, 4, 6, and 7, shown at the bottom of the tree)

This produces the tree in Figure 1. Since the regression function is actually linear, the splits are intuitive: first we split the root node (node 1) at around the midpoint of the distribution of doses ($D = 0.585$), then we split the observations

with doses higher than 0.585 (node 2) at about their median dose, 0.759, producing the terminal nodes 3 and 4. Similarly with doses less than 0.585 (node 5), we split at approximately the median dose (0.317), yielding terminal nodes 6 and 7.

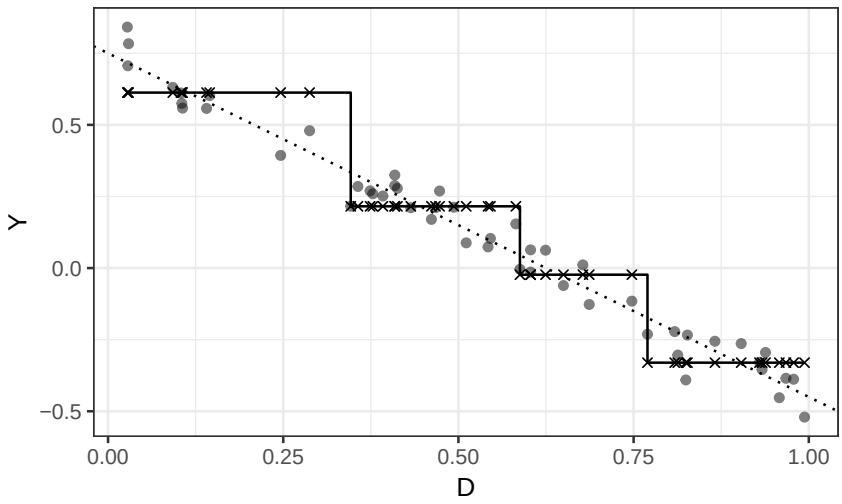

Figure 2: Predicted regression function for example data using CART. The grey dots are the observations and the dotted line is the true relationship from which they were generated. The solid line connects the predicted values, which are shown with ×s.

The resulting predicted regression function is shown as a solid line in Figure 2, with ×s showing the predicted value for each observation in the training data. The grey dots are the actual observations and the dotted line represents the true function from the observations were simulated. Note the piecewise constant form of the function, where all observations in a terminal node receive the same predicted value.

Using this model to predict the change in tumor masses, Y, for the test data results in an root mean squared error of about 0.117 and a coefficient of determination (R^2) of about 0.93. In practice, we would be well advised to relax our stopping criteria to allow smaller terminal nodes in this setting. In general, one only loses computing time by setting stopping criteria that allow very complicated trees in the growing phase, since branches that aren't predictive will be pruned (we did not here to keep the tree simple for the purposes of illustration).

5 Multivariate adaptive regression splines (MARS)

MARS (J. H. Friedman 1991) can be viewed as a modification of the CART procedure to improve performance in the regression setting, or as a generalization of stepwise linear regression methods that incorporates automatic modeling of interactions. MARS accomplishes this latter feature through modeling covariates as piecewise linear basis functions and their products. These piecewise linear basis functions are of the form

$$(x - t)_+ \text{ and } (t - x)_+ \tag{17}$$

where

$$(Z)_+ = \max(Z, 0) = \begin{cases} Z, & \text{if } Z > 0 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

The set in equation 17 is sometimes called a *reflected pair* because the values they take on are symmetric (reflected) across t. Each function in the pair is called a *hinge* function, and can alternatively be denoted $h(x - t)$ and $h(t - x)$ respectively. These hinge functions are joined at t, a *knot*.

5.1 Forward pass

As in forward selection methods, in MARS, we begin constructing our model with only a constant:

$$f(X) = \beta_0 \tag{19}$$

Then, we consider adding a pair of piecewise linear functions made by splitting some covariate X_j at a knot placed at one of the observed values of X_j, x_{ij}:

$$\beta_1(X_j - t)_+ + \beta_2(t - X_j)_+, \; i = 1, 2, \ldots, n; \; j = 1, 2, \ldots, p. \tag{20}$$

To determine which predictor to add and which knot to split upon, each of the j predictors, split at each observed value x_{ij} is evaluated in turn by adding each pair to the model and calculating the training error, e.g. the SSE. The covariate and split point that cause the greatest reduction in error are added to the model.

Henceforth, we consider adding a pair piecewise linear functions with general form

$$\beta_{M+1}h_\ell(X) \cdot (X_j - t)_+ + \beta_{M+2}h_\ell(X) \cdot (t - X_j)_+, \; h_\ell(X) \in \mathcal{M}, \; t \in \{x_{ij}\}. \tag{21}$$

Where $\mathcal{M}$ is the set of hinge functions (or products of hinge functions) already in the model. Note that this equation also holds for the first step, in which case

$h_\ell(X) = h_0(X) = 1$. As before, we add the pair that causes the greatest reduction in training error. To simplify the set of models considered, we may provide a limit on the degree, or order of interactions we are interested in considering. For example with degree = 1, we only consider an additive model (all $h_\ell(X) = h_0(X) = 1$), with degree = 2, we consider a maximum of two-way interactions, and so on. Pairs are added until a user-defined limit on the number of terms in the model is reached, or the reduction in training error is smaller than some predefined threshold.

The resulting MARS regression function is of the form

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X), \tag{22}$$

where M is the number of terms in the model, and $h_m(X)$ is a hinge function like one of those in equation 17 or a product of such functions.

5.2 Backward pass

Usually, the function produced by the forward pass is too large and overfits the data, we use a pruning (backward deletion) procedure analogous to backward selection linear regression methods. For each term $h_m(X)$ in the model, we estimate how much the error would increase by removing it, then remove the term for which removal increases the error the least. Once we've removed one term, we repeat this procedure and delete another term. At each stage of this procedure we obtain $\hat{f}_\lambda(X)$, the best model of size λ (best model with λ terms). Note that we do not proceed backwards along the path through which the covariates were added (indeed we add pairs of hinge functions on at a time, but remove them one by one).

To determine the optimal number of terms, λ, we can use a resampling technique (e.g. cross validation) but for computationally efficiency, we often use generalized cross validation (GCV).

5.3 Generalized cross validation

Generalized cross validation is a computational shortcut for linear regression models, which produces an error that approximates the leave-one-out cross-validated error. For MARS, the generalized cross-validation criterion is defined as

$$GCV(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - M(\lambda)/N)^2} \tag{23}$$

where $M(\lambda)$ is the effective number of parameters in the model, accounting for both the actual parameters and parameters used in selecting knot positions. Simulations and mathematical results tell us that $M(\lambda)$ should be set to $r + 3K$, the number of terms plus three times the number of knots, but can be set to $r + 2K$ when the degree = 1; when the model is restricted to be additive (Hastie, Tibshirani, and J. Friedman 2009).

5.4 Relationship to CART

As mentioned above, MARS can also be viewed as a modification of CART to improve performance in the regression setting. If we modified the MARS procedure to consider reflected pairs of step functions of the form $I(x-t>0)$ and $I(t-x>0)$ instead of the piecewise linear basis functions and then replaced the term already in the model involved in a new interaction term by the new interaction term (making the original term unavailable for further interactions), then the forward pass in MARS is equivalent to the CART tree-growing algorithm. In this case, the multiplication of a step function by a pair of reflected step functions is equivalent to splitting a node.

5.5 Variable importance

Similar to CART, we can measure the importance of each variable by summing the reduction in the GCV statistic (or SSE or other metric) whenever a term with the variable is added (or equivalently the amount it increases as all terms with containing a given variable are removed).

5.6 Advantages and disadvantages of MARS

MARS has many of the same advantages as CART, in that it requires very little preprocessing of data, it performs variable selection and models interactions as a part of the model building process, and the models produced are quite interpretable. This last point holds even for models with interactions since products of hinge functions are only nonzero when all hinge functions involved in the product are nonzero, allowing the term to operate only over a relatively small subspace of the covariates involved.

However, the hinge functions of MARS may not be flexible enough to model smooth functions as accurately as some other methods (this can be addressed by bagging, discussed below). Also, higher order interactions will only enter the model if their lower order interactions improve predictive performance, and this need not be true in practice. As with trees, correlated predictors, which don't significantly

impact performance, but can complicate interpretation. Suppose there were two perfectly correlated predictors to consider. Then the choice between the two at any step is essentially random. This could hinder interpretation because the same piece of information may show up in different parts of the model under different names.

5.7 Illustrative example

Returning to our change in tumor mass example from above, let's fit a regression equation using MARS. Here, no further terms were added if there were already 21 terms in the model or adding additional terms improved R^2 by less than 0.001. Terms were pruned using generalized cross validation as discussed above.

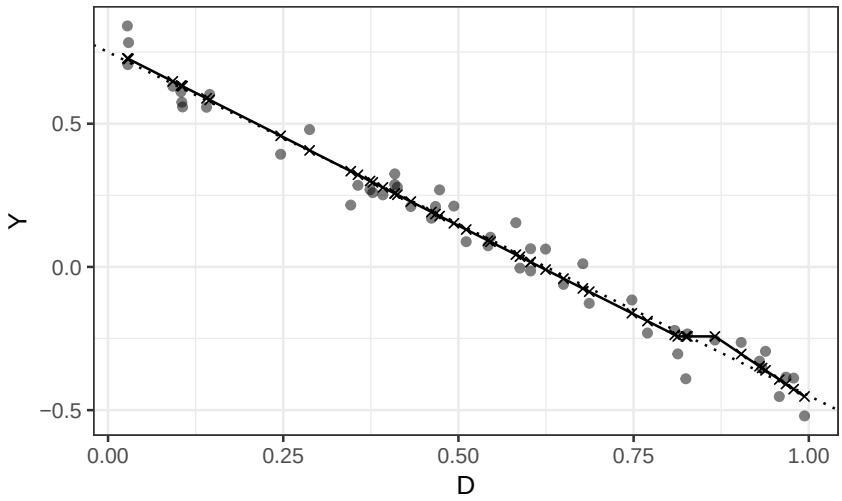

Figure 3: Predicted regression function for example data using MARS. As in Figure 2, the grey dots represent observations, the dotted line represents the true function from which observations were simulated, the solid line represents the predicted regression function, and the ×s represent predicted values for each observation.

The resulting predicted regression function is shown in Figure 3 as a solid line. Grey dots again represent observations, the dotted line represents the true relationship from which data were simulated, and predicted values for each observation are represented by ×s. The fitted MARS equation is

$$\hat{\mathrm{E}}[Y \mid D] \approx -0.24 + 1.24(0.81 - D)_+ - 1.64(D - 0.87)_+ \tag{24}$$

Note that the hinges don't meet because the hinges that entered paired with each remaining hinge were removed in the pruning/backward pass phase. The gap between hinges is shown in our plot as a horizontal line of $Y = -0.24$ for $0.81 < D < 0.87$.

Using this model to predict the testing data yields a root mean squared error of 0.037 and a coefficient of determination (R^2) of 0.99. As these measures indicate, MARS does a much better job of capturing the linear nature of the relationship between Y and D than CART does, with many fewer "splits". Note also that we can restrict MARS to only consider predictors to enter as linear terms (without splitting into hinge functions), which we would probably do here given that the fitted regression line is nearly linear anyway, and would then be easier to interpret.

6 Random forest

6.1 Bagging

As mentioned above, single trees tend to have quite high variance (small changes in the data can produce very different trees), however when grown sufficiently deep or bushy, they can have relatively low bias. Bootstrap aggregating, or *bagging* for short, is a general technique that is useful whenever a modeling technique has these properties (high variance, low bias).

Consider a collection of independent and identically distributed random variables, $X_1, X_2, \ldots, X_n$, each with variance σ^2 and sample mean $\bar{X}$. It's easy to show that

$$\text{Var}[\bar{X}] = \frac{\sigma^2}{n}, \tag{25}$$

the variance of the sample mean is lower by a factor of n than the variance of any single observation. If we could build trees on many different samples then average the results, we could exploit this property to drastically reduce the variance and therefore increase performance.

Typically, however, we only have one sample to work with, but we can take B bootstrap samples of our one sample to approximate many different training samples. We could then build a model on each sample, then aggregate the results from each model into a single prediction. In the case of regression problems, we would average the predicted outcome for each test observation from each model.

Bagging also provides a built-in way to estimate the test error by only averaging predictions for any given tree using the *out-of-bag sample*, the observations that were not in the bootstrap sample used to train that tree (about 1/3 of the observations

Algorithm 4: Bagging

```
for i in 1 to B do
    Take a bootstrap sample of the data
    Fit model to this sample (e.g. grow an unpruned tree)
end
```

on average). This out-of-bag sample typically approximates the cross validation estimate of the error rate well, but requires much less computation.

Unfortunately, we can't in practice achieve a B fold decrease in the variance from a single tree because our trees are not independent. Suppose there was a single very strong predictor. For any tree grown from a bootstrap sample, the first few splits would then tend to be made on this strong predictor. This in turn strongly influences the remaining shape of the tree, leading to correlated trees across bootstrap samples.

6.2 Random forest

Statistically, we can decrease correlation by introducing randomness. So Breiman (2001) suggested that we decrease correlation between trees built on bootstrap samples by only considering a random subset $m_{try} \subset p$ of covariates to split upon for each split instead of considering all p.

Algorithm 5: Random forest

```
for i in 1 to B do
    take a bootstrap sample of the data
    Grow a tree on this sample:
    while stopping criteria not met do
        for each split do
            randomly select m_try of p original covariates
            split on a covariate in m_try causing largest decrease in SSE
        end
    end
    (do not prune)
end
```

This generally leads to better performance than bagging alone and often approaches the performance of more complicated techniques. How should be choose B? There is little danger of overfitting by growing more trees in random forest, so

B is typically set at a large enough number for the error rate to level off, Kuhn and Johnson (2013) recommend starting at 1000. Recommendations for m_{try} are often set at a default of $\sqrt{p}$ or $p/3$, the latter specifically recommended for regression problems, though in practice m_{try} should be chosen using resampling techniques. In the case of bagging, there is typically little gained after $B = 10$ or so. Kuhn and Johnson (2013) recommend setting $B = 50$.

6.3 Variable importance

Breiman (2001) originally proposed randomly permuting one variable at a time in the out-of-bag sample for each tree, then calculating the reduction of predictive performance compared to the non-permuted out-of-bag sample. These differences for each tree would then be averaged across the whole forest and used to rank variables importance (higher reduction, more important). We could also extend the variable importance measure for a single tree to the whole forest by averaging the decrease in SSE for each split upon a given variable over the forest (higher decrease, more important).

6.4 Advantages and disadvantages

Random forests and bagging give significantly more accurate predictions over a single tree, but are also significantly harder to interpret. As with MARS and single trees, where correlated predictors show up is unpredictable and this can dilute variable importance. For example if two perfectly correlated variables were considered, the variable importance for each would be half of what it would be, if it alone had been considered.

6.5 Illustrative example

Returning to our change in tumor mass example from above, let's fit a regression equation using random forest. Actually, in this case since there is only one predictor, random forest is equivalent to bagging and therefore fewer trees will suffice. Here we grow 100 trees each that stop splitting when splitting results in nodes with fewer than 5 observations.

The predicted regression function averaged over the 100 trees is shown in Figure 4. Again, the observations are given by grey dots, the true function from which data were simulated is shown as a dotted line, the predicted observations are shown as $\times$s which are connected by a solid line. Notice how averaging the bagged trees yields a

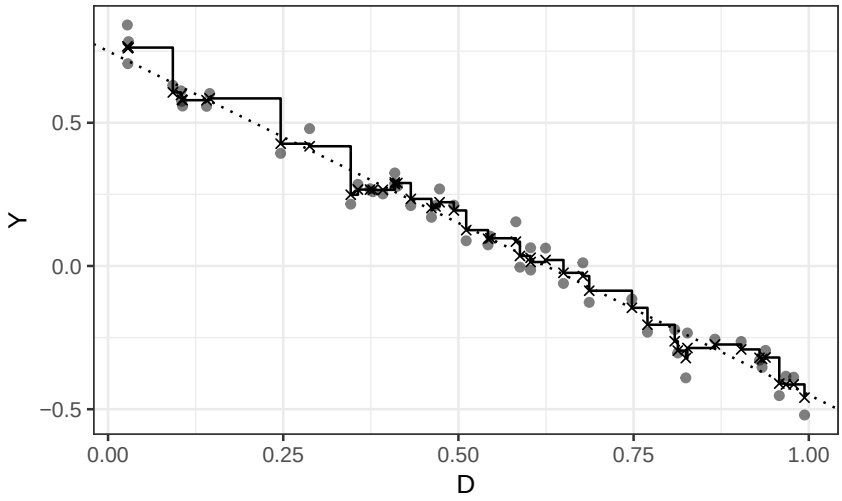

Figure 4: Predicted regression function for example data using Random Forest. As in Figures 2 and 3, the grey dots represent observations, the dotted line represents the true function from which observations were simulated, the solid line represents the predicted regression function, and the ×s represent predicted values for each observation.

line much closer to the truth than the single tree shown above, though each tree was grown using similar stopping criteria as the CART example.

When using this model to predict the test data we obtain a root mean squared error of 0.045 and an R^2 of 0.984. This compares much more favorably to MARS than a single tree, however it comes at the cost of a lack of clarity in the relationship between Y and D in the model.

7 Simulation

Largely inspired by Yufan Zhao, Kosorok, and Zeng (2009) and Yufan Zhao, Zeng, et al. (2011), I estimated optimal dynamic treatment regimes for a cancer clinical trial for an advanced generic cancer. This trial had three stages of treatment, each stage being one month long. A dose of a single drug was given at the beginning of each stage, potentially a different dose for each stage. My primary goal was to find dynamic treatment regimes that would maximize survival time. To achieve this, I first simulated a trial to serve as training data where at the beginning of each of the

three months ($t = 0, 1, 2$), patients' tumor masses and qualities of life were measured, and each patient was assigned a random dose of the drug.

7.1 Patient model

7.1.1 Transition functions

Patient toxicity measured at month t, W_t, which I define as the negative of wellness or quality of life[5], was modeled as a function of tumor mass and dosage of treatment:

$$W_t^* = 0.1M_{t-1} + 1.2(D_{t-1} - 0.5) + W_{t-1} \tag{26}$$

$$W_t = \begin{cases} W_t^* & \text{if } W_t^* > 0 \\ 0 & \text{if } W_t^* \leq 0 \end{cases} \tag{27}$$

This model represents the belief that higher tumor masses at month $t-1$, M_{t-1}, increase the measured toxicity at month t, W_t (by decreasing patient wellness). For a given M_{t-1}, doses at month $t-1$, D_{t-1}, above 0.5 increase W_t, while D_{t-1} below 0.5 decrease W_t due to fewer toxic effects from treatment. Further, W_t is bounded by 0, which corresponds to no toxic effects.

Tumor mass at month t, M_t, was modeled as a function of toxicity measured the previous month, W_{t-1}, and dose assigned the previous month D_{t-1}:

$$M_t^* = [0.15W_{t-1} - 1.2(D_{t-1} - 0.5) + M_{t-1}]I(M_{t-1} > 0) \tag{28}$$

$$M_t = \begin{cases} M_t^* & \text{if } M_t^* > 0 \\ 0 & \text{if } M_t^* \leq 0 \end{cases} \tag{29}$$

lower overall health (higher W_{t-1}) leads to greater tumor growth, while for a given W_{t-1}, D_{t-1} above 0.5 decreases M_t and D_{t-1} below 0.5 allows M_t to increase, with M_t bounded by 0 since mass is strictly positive.

7.1.2 Survival model

Survival times (in months) at month t were generated from an exponential distribution. The rate parameter for this distribution was modeled as a function of the

[5]For simplicity, I take wellness and quality of life to be equivalent and exactly the opposite of toxicity

tumor mass and toxicity which would result at time $t+1$:

$$\lambda_t(M_{t+1}, W_{t+1}) = \exp(-5.5 + W_{t+1} + 1.2M_{t+1} + 0.75W_{t+1}M_{t+1}). \tag{30}$$

This indicates that we take tumor mass to be more important in survival than toxicity. The interaction between tumor mass and toxicity reflects the belief that higher levels of both tumor mass and toxicity lead to a greater reduction in survival than either alone.

7.2 Basic scenario

For the simulated trial that would serve as training data, I began by simulating random doses of treatment for 1000 patients for each of the three months of treatment. The doses were randomly chosen from a uniform distribution over (0, 1) at the beginning of each month:

$$D_t \overset{iid}{\sim} \text{Unif}(0, 1), \quad t = 0, 1, 2 \tag{31}$$

here $D_t = 0$ represents no treatment, and $D_t = 1$ represents the maximum tolerable dose.

Each patient's initial tumor mass, M_0, and negative of quality of life (toxicity), W_0, were generated independently from uniform distributions between 0 and 2:

$$W_0 \overset{iid}{\sim} \text{Unif}(0, 2), \qquad M_0 \overset{iid}{\sim} \text{Unif}(0, 2) \tag{32}$$

At each month t, the next month's tumor mass, M_{t+1}, and toxicity, W_{t+1}, were computed in response to dose at month t, D_t, for each patient as described above. Similarly, the rate parameter for the survival distribution, $\lambda_t(M_{t+1}, W_{t+1})$, was updated for each patient from these next month's tumor masses and toxicities. Survival times, S_t were then randomly sampled from the exponential distribution with the just computed rate parameter:

$$\begin{aligned} S_t &\sim \text{Exp}(\lambda_t(M_{t+1}, W_{t+1})) && (33)\\ &= \text{Exp}(\exp(-5.5 + W_{t+1} + 1.2M_{t+1} + 0.75W_{t+1}M_{t+1})) && (34) \end{aligned}$$

If $S_{it} \leq 1$ for patient i (i.e. the patient would not survive another month), the reward received was computed as the total log survival time: log of the total number of months the patient had survived (including the fraction of a month before their death, S_{it}). If patient i survived up to the final month of treatment $t = 2$, the

rewards were computed as $\log(S_{i2} + 2)$, the log of the simulated survival time after the final treatment plus the the number of months already elapsed.

$$R_{t+1} = \begin{cases} \log(S_t + \sum_{j=0}^{t} j) & \text{if } S_t \leq 1 \text{ or } t = 2 \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

Note that I use the log of the survival time as a continuous reward (and hence outcome) and do not model it as a censored time-to-event outcome.

7.3 Interaction scenario

The interaction scenario was identical to the basic scenario just described with the addition of two baseline covariates, X_1 and X_2 which were simulated from uniform distributions over (0, 1):

$$X_1, X_2 \overset{iid}{\sim} \text{Unif}(0, 1) \tag{36}$$

Using the values of these covariates, four approximately equally sized subgroups were constructed:

$X_1 < 0.5$ & $X_2 < 0.5$**:** patients simulated identically to those in the basic scenario.

$X_1 > 0.5$ & $X_2 < 0.5$**:** these patients are 50% more sensitive to the effects of the drug:

$$W_t^* = W_t' = 0.1M_{t-1} + 1.2(1.5D_{t-1} - 0.5) + W_{t-1} \tag{37}$$

$X_1 < 0.5$ & $X_2 > 0.5$**:** the treatment is 50% more effective for these patients:

$$M_t^* = M_t' = [0.15W_{t-1} - 1.2(1.5D_{t-1} - 0.5) + M_{t-1}]I(M_{t-1} > 0) \tag{38}$$

$X_1 > 0.5$ & $X_2 > 0.5$**:** the drug is 50% more effective for these patients, but they are also 50% more sensitive to its effects:

$$W_t^* = W_t' = 0.1M_{t-1} + 1.2(1.5D_{t-1} - 0.5) + W_{t-1} \tag{39}$$

$$M_t^* = M_t' = [0.15W_{t-1} - 1.2(1.5D_{t-1} - 0.5) + M_{t-1}]I(M_{t-1} > 0) \tag{40}$$

7.4 High-dimensional noise scenarios

In the noise scenarios, 100 additional variables: $Z = (Z_1, \ldots, Z_{10})$ and $V = (V_1, \ldots, V_{90})$ were also simulated with distributions as follows:

$$Z_1, \ldots, Z_5 \overset{iid}{\sim} N(1, 1) \tag{41}$$

$$Z_6, \ldots, Z_{10} \overset{iid}{\sim} N(-1, 1) \tag{42}$$

$$V_1, \ldots, V_{90} \overset{iid}{\sim} N(0, 1) \tag{43}$$

That is, $Z_1, \ldots, Z_5$ were sampled independently from a normal distribution with mean 1 and standard deviation 1, $Z_6, \ldots, Z_{10}$ were sampled independently from a normal distribution with mean -1 and standard deviation 1, and $V_1, \ldots, V_{90}$ were sampled independently from a standard normal distribution. These covariates were used in two separate scenarios described below.

7.4.1 Noise scenario

In the noise scenario, the Z and V variables above were made available to the models, but had no effect on W_t, M_t, or S_t.

7.4.2 Predictive noise scenario

In the predictive noise scenario, the Z and V variables above were made available to the models and the Z variables had a small influence on the rate parameter in the survival distribution:

$$\lambda_t'(M_{t+1}, W_{t+1}) = \exp\left(-5.5 + W_{t+1} + 1.2M_{t+1} + 0.75W_{t+1}M_{t+1} + 0.05\sum_{i=1}^{10} Z_i\right) \tag{44}$$

but did not change the optimal dose, as this modification only shifts the expected survival up or down.

7.4.3 Both interaction and noise scenarios

The interaction and noise scenarios were also combined together, to create the same subgroups as defined above both in the presence of pure noise, and in the presence of predictive noise, as described above.

7.5 Tuning parameters & details of model fit

CART: Nodes with 15 or fewer observations were not split, nor were splits made if they produced nodes with fewer than 5 observations, or did not increase the overall R^2 by at least 10^{-5}. In the pruning step, the T_α with the smallest 10 fold cross validation error was chosen as the final tree. This was achieved via the `rpart` package (Therneau, Atkinson, and Ripley 2015) in R (R Core Team 2016).

MARS: Only interactions with the treatment variable, D_t, were allowed, and these were restricted to second degree interactions. The forward pass was stopped if more than 200 terms were created, or further terms would decrease the R^2 by less than 0.001. The number of terms in the model, λ, was chosen using generalized cross validation. Modeling was performed with the `earth` package (Milborrow 2017) in R called through the `caret` package (Khun 2016).

RF: Initially 500 trees were grown, with D_t forced to be considered at each split and the out-of-bag samples used to choose m_{try} from an even grid of five values from 2 to p. However, in each scenario, $m_{try} = p$ was at or very near the minimum out-of-bag error, so $m_{try} = p$ (bagging) was performed with with 250 trees. Trees were grown until further splits would result in terminal nodes with less than five observations. Modeling and resampling was performed using the `caret` package in R to call the `ranger` package (Wright and Ziegler 2017).

7.6 Training procedure

With the the survival times resulting from each simulated trial for each scenario listed above, Q-learning was applied using each of CART, MARS, and random forest to estimate the Q-functions at each stage. State signals (M, W, Z, V) were assumed to be Markov, so only the covariates from a particular stage were used in fitting the corresponding Q-function. The fitted models for each stage under each scenario were saved for estimating optimal treatments for patients in the validation set. Maximum rewards in each stage of Q-learning were estimated by generating predicted rewards from the set of doses $D_t = 0, 0.01, 0.02, \ldots, 1$ and choosing the largest.

To quantify the influence the training data have on the resulting estimated optimal dynamic treatment regimes, I repeated the processes described in the preceding paragraph for 100 unique training sets each with 1000 patients.

7.7 Validation

Initial tumor masses (M_0) and toxicities (W_0) for 2000 new patients, were simulated identically to those in each scenario for the training data. In order to make treatment regimes more comparable, I then made copies of these 2000 individuals, one set of 2000 for each of the following regimes:

10 constant dose regimes: the same doses of $0.1, 0.2, \ldots,$ or 1 given at each stage

Best: the dose that maximizes the expected survival at each month, given complete knowledge of the transition functions and their influence on survival. Doses

were chosen using an exhaustive tree search: considering all possible sequences of three doses and choosing the sequence that produced the largest expected survival time, with each dose chosen from the set $\{0, 0.01, 0.02, \ldots, 1\}$. Sequences also could not have any intermediate survival times leading to death (being less than one month).

CART, MARS, RF: Out of doses from the set $\{0, 0.01, 0.02, \ldots, 1\}$, the dose corresponding to the maximum predicted rewards for each of the 100 CART, MARS, and random forest models for each stage.

The transition and survival functions were identical to the training data, with one exception: to further increase the comparability between regimes, the expected survival time was used as the actual survival time (and the determination of whether a participant had died), rather than the survival time using random draws from an exponential distribution.

To compare the regimes, I calculated the arithmetic mean survival time for the 2000 test patients under each treatment regime. I chose the arithmetic mean because it is more sensitive to very large and very small values, which in this setting should play a larger role. This is because a regime that produces very short survival times for one subgroup of patients but has a median or geometric mean close to another regime that does not is worse relative to our goals. I then calculated the mean and standard deviation of the 100 resulting mean survival times for each modeling technique.

7.7.1 Variable importance

Variable importance was calculated for each model at each stage, as a percentage of the most important variable. For the CART models, variable importance was calculated as the total reduction in SSE resulting from each split on a particular variable. For the MARS models, the importance of each variable was calculated as the increase in SSE which would result from removing each term with a particular variable in it. Variable importance in each tree in the random forest models was calculated in the same way as a single CART tree, then these importances were averaged across all trees in the forest.

8 Simulation results

The arithmetic mean survival times under each treatment regime are shown in Figure 5 for the scenarios without covariate by treatment interaction (patient subgroups

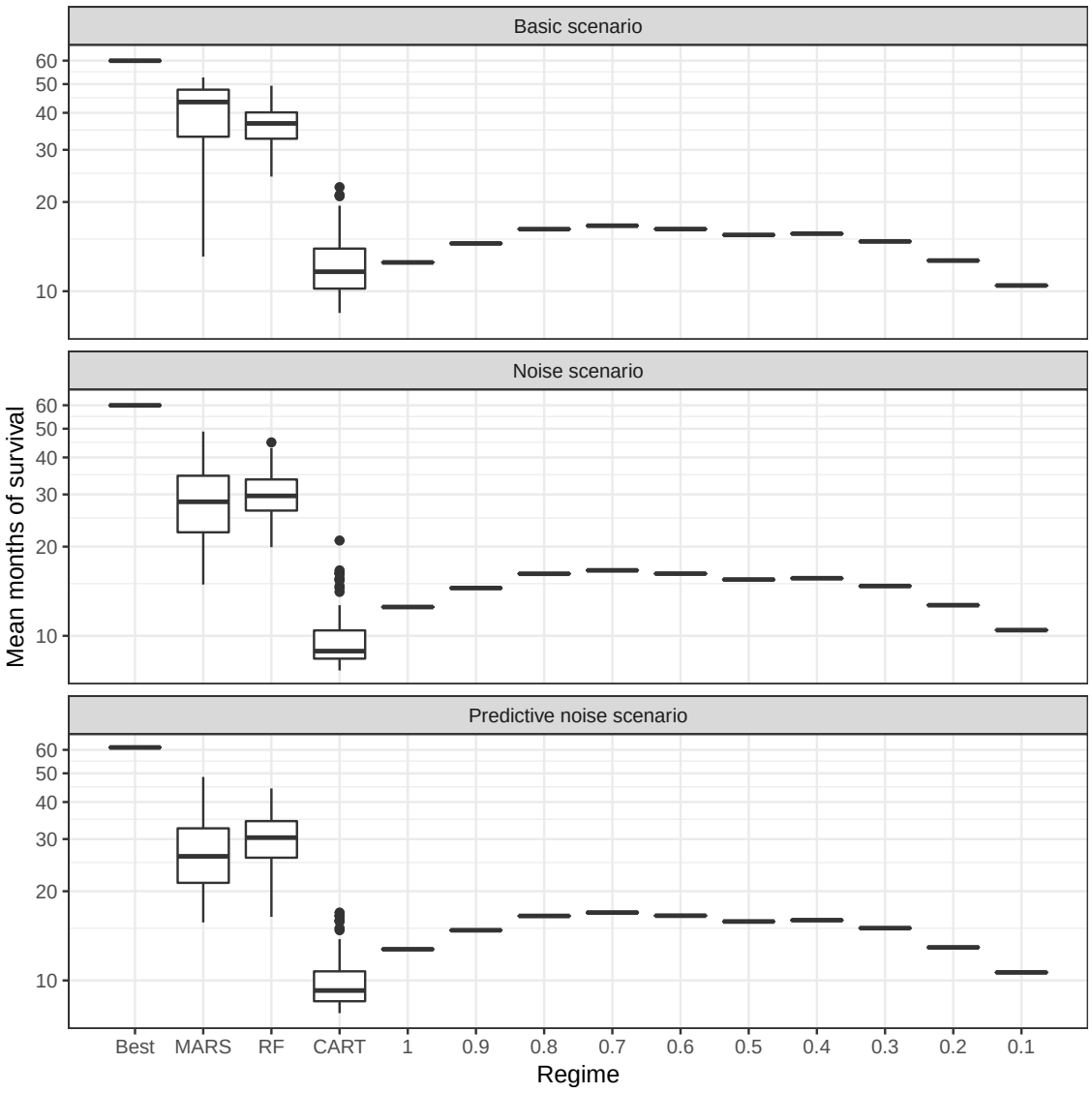

Figure 5: Mean months of survival under each regime for the scenarios without interaction, log scale. The mean months of survival on the test set (initial tumor masses and toxicities for 2000 patients) for the regimes estimated using Q-learning (MARS, RF–here equivalent to bagging, and CART), each of the 100 treatment regimes estimated from each of the 100 training sets are shown in box plots. The best regime and the constant dose regimes $(1, 0.9, \ldots, 0.1)$ do not vary across training set and therefore have only one mean survival time for the test set. Observations more than $1.5 \times IQR$ from either hinge (lower or upper quartile) are shown as points.

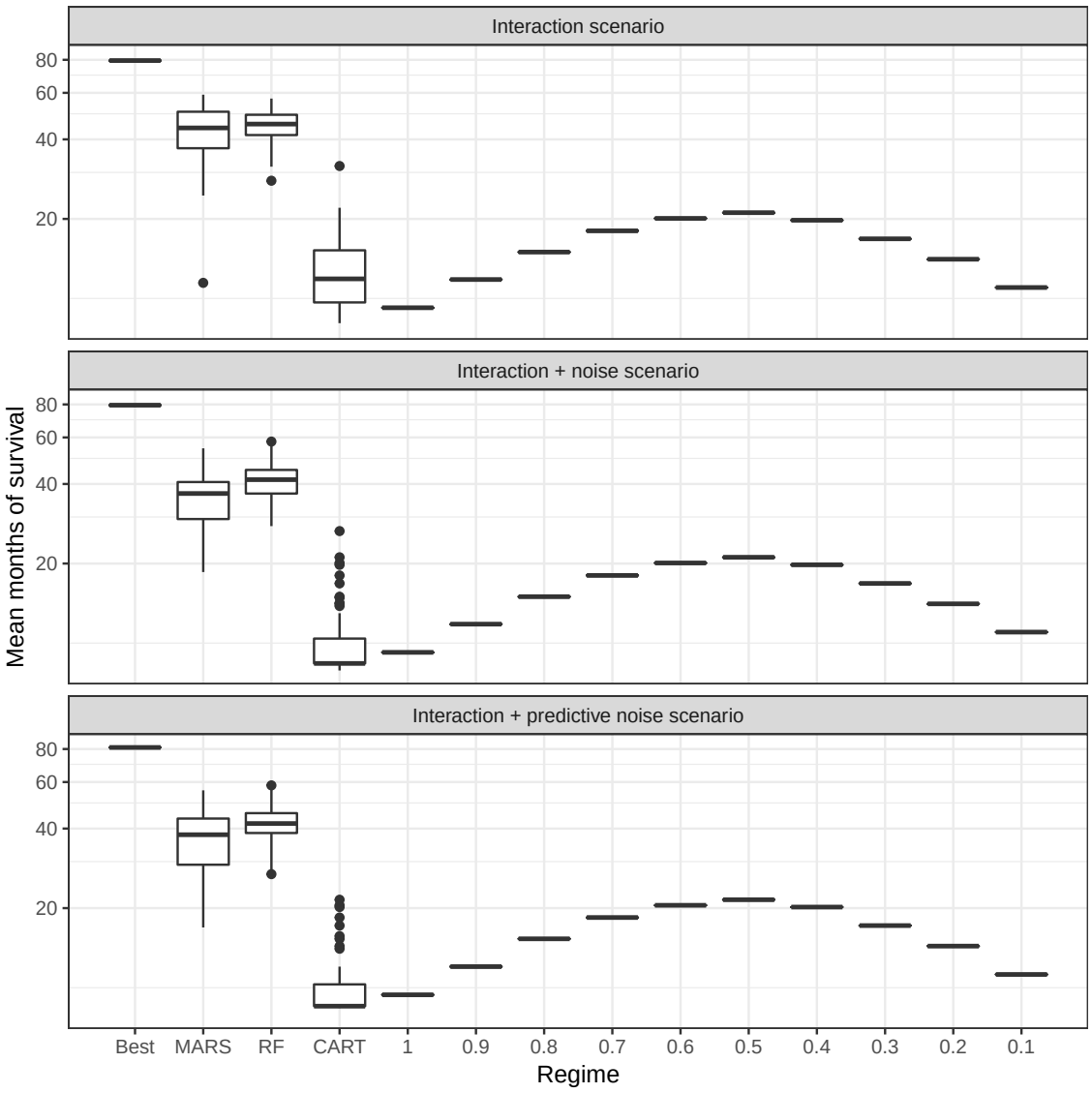

Figure 6: Mean months of survival under each regime for scenarios with interaction, log scale. As in Figure 5, mean months of survival for the regimes estimated using Q-learning (MARS, RF–here equivalent to bagging, and CART), each of the 100 treatment regimes estimated from each of the 100 training sets are shown in box plots. The best regime and the constant dose regimes $(1, 0.9, \ldots, 0.1)$ do not vary across training set and therefore have only one mean survival time for the test set. Observations more than $1.5 \times IQR$ from either hinge (lower or upper quartile) are shown as points.

Table 1: Mean months of survival under each scenario (in months), averaged over 2000 test patients for each regime. Each test set of 2000 had identical initial tumor masses and toxicities (M_0, W_0). Best is the optimal dose sequence obtained from evaluating every possible three dose sequence. MARS, RF (random forest, which is equivalent to bagging in these scenarios), and CART (regression trees using the CART procedure) represent the average survival times when using the optimal dose regime estimated from Q-learning using each method to fit the Q-functions, averaged over the 100 training set replicates. B is the basic scenario, N is noise scenario, PN is the predictive noise scenario, I is the interaction scenario, I+N is the interaction and noise scenario, and I+PN is the interaction and predictive noise scenario. Each scenario is described in detail in section 7.

Regime	B	N	PN	I	I+N	I+PN
Best	60.0	60.0	61.1	79.5	79.5	81.1
MARS	39.8	29.1	27.7	43.9	35.8	37.0
RF	36.3	30.0	30.7	45.8	41.2	42.1
CART	12.5	9.9	10.0	12.8	10.0	9.9
1	12.5	12.5	12.7	9.2	9.2	9.4
0.9	14.5	14.5	14.8	11.8	11.8	12.0
0.8	16.2	16.2	16.5	15.0	15.0	15.3
0.7	16.6	16.6	16.9	18.0	18.0	18.4
0.6	16.2	16.2	16.5	20.1	20.1	20.5
0.5	15.5	15.5	15.8	21.1	21.1	21.5
0.4	15.6	15.6	16.0	19.8	19.8	20.2
0.3	14.7	14.7	15.0	16.8	16.8	17.2
0.2	12.7	12.7	12.9	14.1	14.1	14.4
0.1	10.5	10.5	10.6	11.0	11.0	11.2

with differential treatment response) and in Figure 6 for the interaction scenarios–those with covariate by treatment interaction. "Best" shows the arithmetic mean survival time for the 2000 test patients when following the best possible dose sequence. The optimal regimes estimated using Q-learning are labelled as "MARS", "RF", and "CART" for Q-learning using MARS, random forest (which in these situations was equivalent to bagging), and regression trees using the CART procedure to estimate the Q-functions, respectively. In Figures 5 and 6, each of the 100 mean survival times obtained using the optimal regime derived from the 100 different training sets on the test set are displayed as separate points in box plots for the three modeling techniques. Comparison regimes are shown with only a single line because they do not require training and are therefore invariant across training set. Table 1 displays the arithmetic mean survival times under each regime, averaged over all of the 100 test set survival times for each of the regimes estimated using each Q-learning technique on each training set.

Table 2: Standard deviation of mean survival times across training set replicates for Q-learning models. B is the basic scenario, N is noise scenario, PN is the predictive noise scenario, I is the interaction scenario, I+N is the interaction and noise scenario, and I+PN is the interaction and predictive noise scenario. Each scenario is described in detail in section 7.

Regime	B	N	PN	I	I+N	I+PN
MARS	9.8	8.8	8.3	9.6	7.9	9.0
RF	5.2	5.2	5.8	6.0	6.4	6.5
CART	3.0	2.5	2.2	4.0	3.2	2.7

In the basic (B) scenario, MARS and random forest produce similar mean survival times, about two thirds of the best survival times, with MARS performing slightly better on average. However, as will be repeated throughout scenarios, MARS exhibits more sensitivity to the training set (higher variance) as shown in Figure 5 and in Table 2, which shows standard deviations in mean months of survival across training sets. As is also repeated across scenarios, a single regression tree (labelled "CART" throughout) performs very poorly, worse overall than many constant dose regimes. When the 100 noise variables are added in the noise (N) scenario, there is approximately a 10 month decrease in mean survival times for MARS and RF, bringing them to about half of the best possible survival times and about a 2 month decrease for CART. MARS then performs slightly worse than random forest when

noise variables are added. Allowing some of the noise variables to shift survival times up or down in the predictive noise (PN) scenario produces results very similar to the noise scenario without predictive noise.

Results from the interaction scenarios, scenarios with the four treatment subgroups created by X_1 and X_2, shown in Figure 6 and Table 1, were similar to those in the scenarios without interaction, but shifted towards longer survival. Despite the shift upwards, the trend is very similar to the scenarios without interaction. In the interaction only scenario (I), MARS and random forest produce similar mean survival times, a little more than half of those produced by the best possible regime, with MARS showing greater variability across training sets. Addition of noise variables to the interaction scenario (I+N), results in a reduction of around 10 months of survival time for MARS and random forest, bringing them to half or a bit less than half of the best possible survival time. As in the scenarios without interaction, MARS decreases more than random forest when the noise variable are added. Allowing 10 of the noise variables to shift survival up or down by a little in the interaction and predictive noise (I+PN) scenario has little effect on the performance of the estimated regimes. CART produces results very similar to those in the scenarios without interaction.

Variable importances as a percentage of the most important variable for each method, stage, and training set replicate are shown in Figures 7 and 8. The interaction variables X_1, X_2, noise variables with no predictive value, $V_1, \ldots, V_{90}$, and noise variables with some predictive value in the scenarios with predictive noise, $Z_1, \ldots, Z_{10}$ are shown together as X, V, and Z respectively. Specifically, the Z label in Figures 7 and 8 shows violin plots of the relative importance for all of the variables in the Z vector for each scenario (and for every stage, so 3000 importances for each violin). Table 3 shows the mean and standard deviation of the relative importances over the 100 different training models and the three stages, for each variable group vector and each method. For example, the average importance of a V variable for all MARS models (at all stages) in the noise scenario (N) is 5.4, with a standard deviation of 11.1. Mean importances were quite similar between stages.

As shown in Figures 7 and 8 and Table 3, M and W are the most important variables across all model types. This aligns well with the higher importance of M and W on survival in the simulated data. Further, M is almost always the most important variable, which agrees with the higher weight that was placed on it in the simulation relative to W.

Figures 7 and 8 and Table 3 also show that this version of MARS is much better than the other modeling techniques at identifying interacting variables and the treatment (D) as important. However, it is also much more prone to identify noise

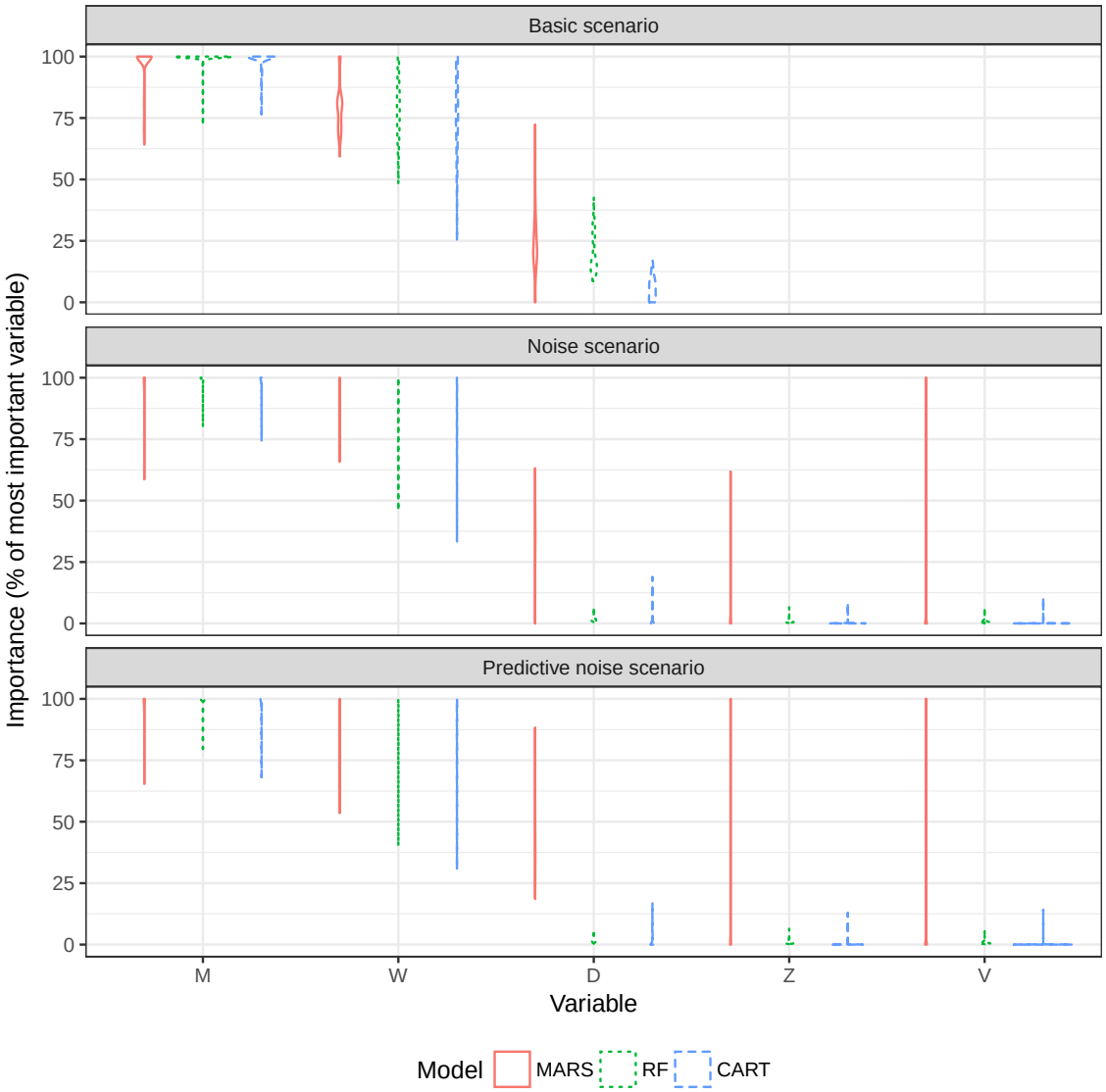

Figure 7: Variable importances for the scenarios without interaction. Importances are given as a percentage of the most important variable in each model. Importances from each stage and each training set replicate are shown together in each violin (e.g. each of the 60 importances for tumor mass for each stage for each model are shown together in the three respective violins above M for each scenario). All of the importances for each noise variable in the Z vector are labelled Z, while the variables in the V and X vectors are labelled V and X respectively.

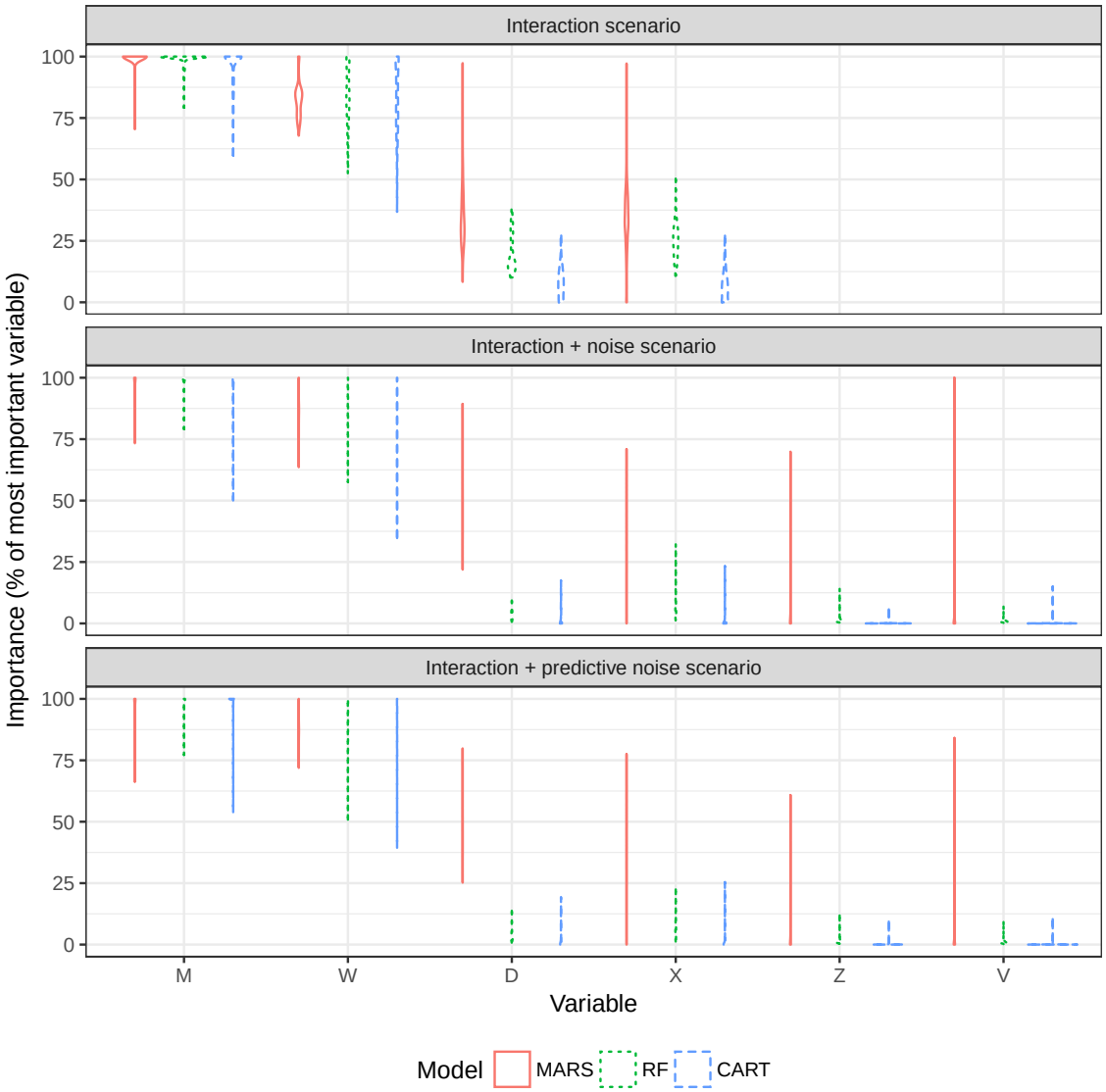

Figure 8: Variable importances for the interaction scenarios. Importances are given as a percentage of the most important variable in each model. Importances from each stage and each training set replicate are shown together in each violin (e.g. each of the 60 importances for tumor mass for each stage for each model are shown together in the three respective violins above M for each scenario). All of the importances for each noise variable in the Z vector are labelled Z, while the variables in the V and X vectors are labelled V and X respectively.

Table 3: Mean variable importance across stages and model replicates. Importances are given as a percentage of the most important variable in each model. Standard deviations are shown in parentheses. Mean and standard deviation variable importance over all of the importances for each noise variable in the Z vector are labelled Z, while the mean and standard deviation importance across all variables in the V and X vectors are labelled V and X respectively.

Variable	B	N	PN	I	I+N	I+PN
MARS						
M	97.6 (7.5)	97.2 (8.0)	96.7 (8.2)	98.8 (5.1)	97.7 (6.3)	97.8 (6.2)
W	78.6 (9.3)	82.7 (8.1)	83.5 (8.7)	82.2 (6.7)	85.3 (7.3)	86.1 (7.1)
D	24.4 (13.3)	36.2 (7.7)	37.1 (8.6)	35.1 (12.8)	45.4 (9.8)	45.4 (8.7)
X				33.4 (12.1)	43.7 (13.6)	43.9 (12.6)
Z		5.4 (11.1)	8.0 (13.2)		5.7 (11.6)	7.8 (13.5)
V		5.3 (10.8)	5.3 (10.9)		5.9 (11.9)	5.8 (11.8)
RF						
M	99.7 (2.0)	99.6 (2.1)	99.5 (2.2)	99.3 (2.3)	99.0 (3.1)	98.9 (3.5)
W	79.5 (11.5)	79.0 (12.8)	79.9 (13.2)	85.1 (11.6)	86.0 (10.9)	86.2 (11.5)
D	19.4 (7.8)	1.7 (0.8)	1.9 (0.9)	19.8 (7.5)	3.7 (1.8)	3.7 (2.1)
X				24.9 (6.8)	6.5 (3.6)	6.5 (3.5)
Z		0.9 (0.5)	0.9 (0.6)		1.2 (0.6)	1.2 (0.7)
V		0.8 (0.4)	0.8 (0.4)		1.1 (0.5)	1.1 (0.5)
CART						
M	98.8 (3.8)	98.6 (4.2)	98.5 (4.3)	97.3 (6.1)	96.7 (7.8)	96.5 (7.9)
W	78.7 (15.3)	77.5 (16.4)	79.4 (16.3)	84.1 (14.2)	83.6 (14.3)	84.4 (14.6)
D	5.7 (4.3)	2.0 (3.6)	2.0 (3.7)	8.1 (6.2)	2.3 (3.9)	2.5 (4.2)
X				6.9 (5.9)	1.1 (3.0)	1.1 (2.9)
Z		0.0 (0.4)	0.1 (0.5)		0.0 (0.3)	0.0 (0.4)
V		0.0 (0.4)	0.0 (0.4)		0.0 (0.4)	0.0 (0.4)

variables as important. As with mean survival times, there is higher variance across training sets in the importances MARS assigns to treatment and interacting variables.

Random forest is best at identifying the noise variables as not important, but also gives a relatively low importance to interacting and treatment variables. A single regression tree (CART) shows a similar trend in importances as the random forest models, but with near zero importance for noise variables and lower importance for dose and interacting (X) variables.

9 Discussion

Q-learning using random forest (which in these scenarios was equivalent to bagging) and Multivariate Adaptive Regression Splines modified to only interact variables with treatment can increase survival times compared to constant dose regimes by large margins in this simulation. However, Q-learning using single regression trees grown with the CART procedure often performed worse than many constant dose regimes.

This agrees with the results of Yufan Zhao, Kosorok, and Zeng (2009), but also shows that the more interpretable MARS models can produce comparable results to more complex "black box" models. Further, it shows the results in context with the best possible sequence, and gives inklings of the variability in performance due to differences in training data.

The lack of performance from single regression trees is likely due to the splitting criterion being based on predictive performance and not on a measure of the degree of interaction with the treatment variable. With SSE as the splitting criterion, one can choose to grow a deep, bushy tree that is relatively unbiased but also has high variance (overfits the data), or one can prune to a less variable but more biased tree. In our scenarios, the treatment variable, dose, plays much less of a role in the absolute survival time than tumor mass or toxicity (as shown in the variable importance measures) so it tends to only be split upon further down the tree. To model the effect of dose well, therefore, a large tree is required. However, large trees tend to overfit and hence also don't consistently model the effect of dose well. Attempting to correct this by pruning results in most (or all) of the effects due to dose being pruned off because the cost-complexity criterion use for pruning is also not based on capturing the differential effects of treatment. An analogous explanation applies for the interacting variables.

Restricting MARS to only interact covariates with the treatment gave comparable performance to random forest on average, but the survival times produced were more

variable, and for some training data were quite low. This most likely due to the restriction on interactions with treatment, which are harder to consistently detect using SSE as the metric for variable addition and removal. This instability could be likely be improved through bagging, though bagging would negate much of the interpretably gained from using MARS.

Using these methods, subgroup identification is difficult. With MARS, although the true interacting variables often have high importance measures, for any given set of training data so may spurious variables. Random forest works well despite low importances for dose and interacting variables, which while a little larger on average than noise variables, would be hard to distinguish in practice.

Due to constraints in computing resources, I could not obtain more precise estimates of the sensitivity of the results to different training sets by simulating more training set replicates. Further, censoring of observations was ignored. If similar methods will be applied to real data with survival outcomes, censoring will need to be incorporated.

The above being said, I showed that statistical learning methods that were designed to maximize predictive performance can be used to personalize treatment regimes with a continuous treatment, increasing mean length of survival in simulated cancer patients.

While this approach is more effective than constant doses in this setting, performance can almost certainly be improved by modifying the techniques used to fit Q-functions to use magnitude of interaction as the basis for model building and evaluating performance. This could also greatly increase the interpretability of the Q-functions, and would make subgroup identification much easier.

Extending this framework to observational data could have a large impact. For example, through solving some casual inference questions, these techniques could be applied to electronic medical record data for people with chronic diseases. This could lead to a much better understanding of how to to tailor treatment sequences to a patient's individual characteristics and their changing disease.

References

Baselga, José et al. (2006). "Adjuvant trastuzumab: a milestone in the treatment of HER-2-positive early breast cancer." In: *The oncologist* 11 Suppl 1, pp. 4–12. DOI: 10.1634/theoncologist.11-90001-4.

Breiman, Leo (2001). "Random forests". In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324.

Breiman, Leo et al. (1984). *Classification and Regression Trees*.

Byar, D P (1985). "Assessing apparent treatment-covariate interactions in randomized clinical trials". In: *Statistics in medicine* 4.September 1984, pp. 255–63. DOI: 10.1002/sim.4780040304.

Cai, Tianxi et al. (2011). "Analysis of randomized comparative clinical trial data for personalized treatment selections". In: *Biostatistics* 12.2, pp. 270–282. DOI: 10.1093/biostatistics/kxq060.

Chakraborty, Bibhas and Erica E.M. Moodie (2013). *Statistical Methods for Dynamic Treatment Regimes*. Statistics for Biology and Health. New York, NY: Springer New York, pp. 31–52. ISBN: 978-1-4614-7427-2. DOI: 10.1007/978-1-4614-7428-9.

Chakraborty, Bibhas and Susan A Murphy (2014). "Dynamic Treatment Regimes". In: *Annual Review of Statistics and Its Application* 1.1, pp. 447–464. DOI: 10.1146/annurev-statistics-022513-115553.

Dusseldorp, Elise and Iven Van Mechelen (2014). "Qualitative interaction trees: a tool to identify qualitative treatment-subgroup interactions". In: *Statistics in Medicine* 33.2, pp. 219–237. DOI: 10.1002/sim.5933.

Fischer, Tobias et al. (2015). *Individualized Medicine: Ethical, Economical and Historical Perspectives*. Ed. by Tobias Fischer et al. ISBN: 978-3-319-11718-8. DOI: 10.1007/978-3-319-11719-5.

Foster, Jared C., Jeremy M G Taylor, and Stephen J. Ruberg (2011). "Subgroup identification from randomized clinical trial data". In: *Statistics in Medicine* 30.24, pp. 2867–2880. DOI: 10.1002/sim.4322.

Friedman, Jerome H. (1991). "Multivariate Adaptive Regression Splines". In: *The Annals of Statistics* 19.1, pp. 1–67. DOI: 10.1214/aos/1176347963.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer New York. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7.

Khun, Max (2016). *caret: Classification and Regression Training*. URL: https://cran.r-project.org/package=caret.

Kuhn, Max and Kjell Johnson (2013). *Applied Predictive Modeling*. New York, NY: Springer New York. ISBN: 978-1-4614-6848-6. DOI: 10.1007/978-1-4614-6849-3.

Lipkovich, Ilya et al. (2011). "Subgroup identification based on differential effect search-A recursive partitioning method for establishing response to treatment in patient subpopulations". In: *Statistics in Medicine* 30.21, pp. 2601–2621. DOI: 10.1002/sim.4289.

Milborrow, Stephen (2017). *earth: Multivariate Adaptive Regression Splines*. URL: https://cran.r-project.org/package=earth.

Pirmohamed, M et al. (2004). "Adverse drug reactions as cause of admission to hospital: prospective analysis of 18 820 patients". In: *BMJ* 329.7456, pp. 15–19. DOI: 10.1136/bmj.329.7456.15329/7456/15[pii].

Qian, Min and Susan A Murphy (2011). "Performance guarantees for individualized treatment rules". In: *The Annals of Statistics* 39.2, pp. 1180–1210. DOI: 10.1214/10-AOS864.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. URL: https://www.r-project.org/.

Schleidgen, Sebastian et al. (2013). "What is personalized medicine: sharpening a vague term based on a systematic literature review". In: *BMC Medical Ethics* 14.1, p. 55. DOI: 10.1186/1472-6939-14-55.

Su, Xiaogang et al. (2009). "Subgroup Analysis via Recursive Partitioning". In: *Journal of Machine Learning Research* 10, pp. 141–158. ISSN: 15324435.

Sutton, Richard S and Andrew G Barto (2016). *Reinforcement Learning: An Introduction*. 2nd. Unpublished draft. URL: http://incompleteideas.net/sutton/book/bookdraft2016sep.pdf.

Therneau, Terry, Beth Atkinson, and Brian Ripley (2015). *rpart: Recursive Partitioning and Regression Trees*. URL: https://cran.r-project.org/package=rpart.

Watkins, Christopher J. C. H. and Peter Dayan (1992). "Q-learning". In: *Machine Learning* 8.3-4, pp. 279–292. DOI: 10.1007/BF00992698.

Watkins, CJCH (1989). "Learning From Delayed Rewards". PhD Thesis. King's College, Cambridge.

Wright, Marvin N and Andreas Ziegler (2017). "ranger : A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". In: *Journal of Statistical Software* 77.1, pp. 1–17. DOI: 10.18637/jss.v077.i01.

Zeileis, Achim, Torsten Hothorn, and Kurt Hornik (2008). "Model-based recursive partitioning". In: *Journal of Computational and Graphical Statistics* 17.2, pp. 492–514. DOI: 10.1198/106186008X319331.

Zhang, B et al. (2012). "Estimating optimal treatment regimes from a classification perspective". In: *Stat* 1, pp. 103–114. DOI: 10.1002/sta.411.

Zhao, Lihui et al. (2013). "Effectively Selecting a Target Population for a Future Comparative Study." In: *Journal of the American Statistical Association* 108.502, pp. 527–539. DOI: 10.1080/01621459.2013.770705.

Zhao, Yingqi et al. (2012). "Estimating Individualized Treatment Rules Using Outcome Weighted Learning". In: *Journal of the American Statistical Association* 107.499, pp. 1106–1118. DOI: 10.1080/01621459.2012.695674.

Zhao, Yufan, Michael R Kosorok, and Donglin Zeng (2009). "Reinforcement learning design for cancer clinical trials". In: *Statistics in Medicine* 28.26, pp. 3294–3315. DOI: 10.1002/sim.3720.

Zhao, Yufan, Donglin Zeng, et al. (2011). "Reinforcement Learning Strategies for Clinical Trials in Nonsmall Cell Lung Cancer". In: *Biometrics* 67.4, pp. 1422–1433. DOI: 10.1111/j.1541-0420.2011.01572.x.