Lecture 22: Bagging and Random Forest

Wenbin Lu

Department of Statistics
North Carolina State University

Fall 2019

Outlines

- Bagging Methods
  - Bagging Trees
- Random Forest
  - Applications in Causal Inference/Optimal Treatment Decision

Ensemble Methods (Model Averaging)

A machine learning *ensemble* meta-algorithm designed to improve the stability and accuracy of machine learning algorithms

- widely used in statistical classification and regression.
- can reduce variance and helps to avoid overfitting.
- usually applied to decision tree methods, but it can be used with any type of method.

Bootstrap Aggregation (Bagging)

Bagging is a special case of the model averaging approach.

$$\text{Bootstrap aggregation} = \text{Bagging}$$

- Bagging leads to "improvements for unstable procedures" (Breiman, 1996), e.g. neural nets, classification and regression trees, and subset selection in linear regression (Breiman, 1994).
- On the other hand, it can mildly degrade the performance of stable methods such as K-nearest neighbors (Breiman, 1996).

Basic Idea

Given a standard training set D of size n, bagging

- generates B new training sets D_i, each of size n', by sampling from D uniformly and with replacement.
- The B models are fitted using the above B bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

This kind of sample is known as a bootstrap sample.

- By sampling with replacement, some observations may be repeated in each D_i.
- If $n' = n$, then for large n, the set D_i is expected to have the fraction $(1 - 1/e) \approx 63.2\%$ of the unique examples of D, the rest being duplicates.

Bagging Procedures

Bagging uses the bootstrap to improve the estimate or prediction of a fit.

- Given data $\mathbf{Z} = \{(x_1, y_1), ..., (x_n, y_n)\}$, we generate B bootstrap samples $\mathbf{Z}^{*b}$
  - **Empirical distribution** $\hat{\mathcal{P}}$: putting equal probability $1/n$ on each (x_i, y_i) (discrete)
  - Generate $\mathbf{Z}^{*b} = \{(x_{1*}, y_{n*}), ..., (x_{n*}, y_{n*})\} \sim \hat{\mathcal{P}}, b = 1, ..., B$
- Obtain $\hat{f}^{*b}(x)$, $b = 1, ..., B$.
- The Monte Carlo estimate of the bagging estimate

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x).$$

Properties of Bagging Estimates

Advantages:

- Note $\hat{f}_{\text{bag}}(x) \longrightarrow E_{\hat{\mathcal{P}}} \hat{f}^*(x)$ as $B \to \infty$,
- $\hat{f}_{\text{bag}}(x)$ typically has smaller variance than $\hat{f}(x)$;
- $\hat{f}_{\text{bag}}(x)$ differs from $\hat{f}(x)$ only when the latter is nonlinear or adaptive function of data.

Bagging Classification Trees

In classification problems, there are two scenarios

(1) $\hat{f}^b(x)$ is indicator-vector, with one 1 and $K-1$ 0's (hard classification)

(2) $\hat{f}^{*b}(x) = (p_1, ..., p_K)$, the estimates of class probabilities $\widehat{P}^b(Y = k|X = x)$, $k = 1, \cdots, K$ (soft classification). The bagged estimates are the average prediction at x from B trees

$$\hat{f}_k^{\text{bag}}(x) = B^{-1} \sum_{b=1}^{B} \hat{f}_k^{*b}(x), \quad k = 1, \cdots, K.$$

Bagging Classification Trees (cont.)

There are two types of averaging:

(1) Use the majority vote: $\arg\max_k \sum_{b=1}^{B} I\{\hat{f}^b(x) = k\}$.

(2) Use the averaged probabilities. The bagged classifier

$$\hat{G}_{\text{bag}}(x) = \arg\max_k \hat{f}_k^{\text{bag}}(x).$$

Note: the second method tends to produce estimates with lower variances than the first method, especially for small B.

Example

- Sample size $n = 30$, two classes
- $p = 5$ features, each having a standard Gaussian distribution with pairwise correlation $\text{Corr}(X_j, X_k) = 0.95$.
- The response Y was generated according to

$$\Pr(Y = 1 | x_1 \leq 0.5) = 0.2, \quad \Pr(Y = 1 | x_1 > 0.5) = 0.8.$$

- The Bayes error is 0.2.
- A test sample of size $2,000$ was generated from the same population.

We fit

- classification trees to the training sample
- classification trees to each of 200 bootstrap samples

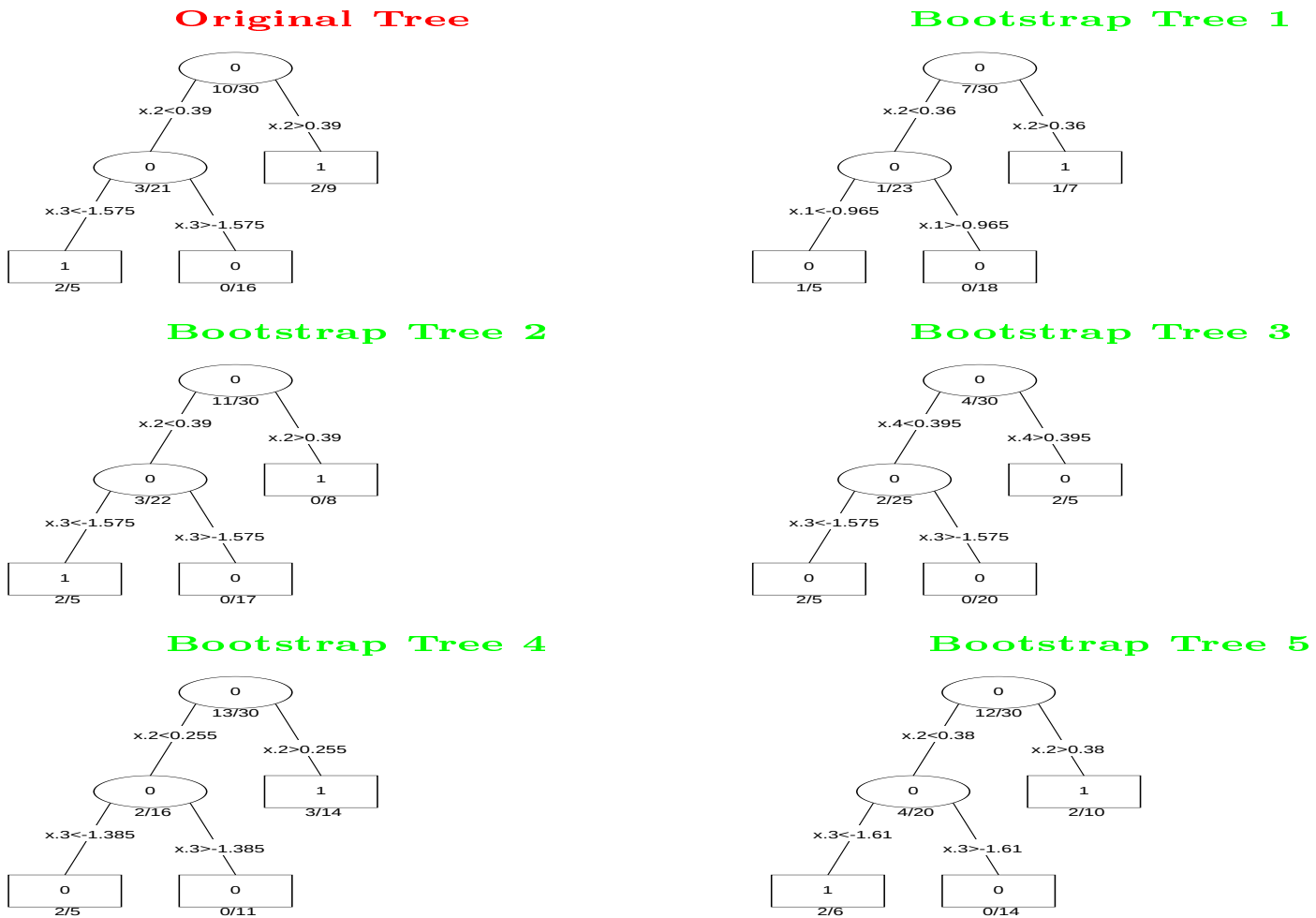


Figure 8.9: *Bagging trees on simulated dataset. Top left panel shows original tree. Five trees grown on bootstrap samples are shown.*

About Bagging Trees

The original tree and five bootstrap trees are all different:

- with different splitting features
- with different splitting cutpoints
- The trees have high variance due to the correlation in the predictors

Averaging reduces variance and leaves bias unchanged.

- Under squared-error loss, averaging reduces variance and leaves bias unchanged.
- Therefore, bagging will often decrease MSE.

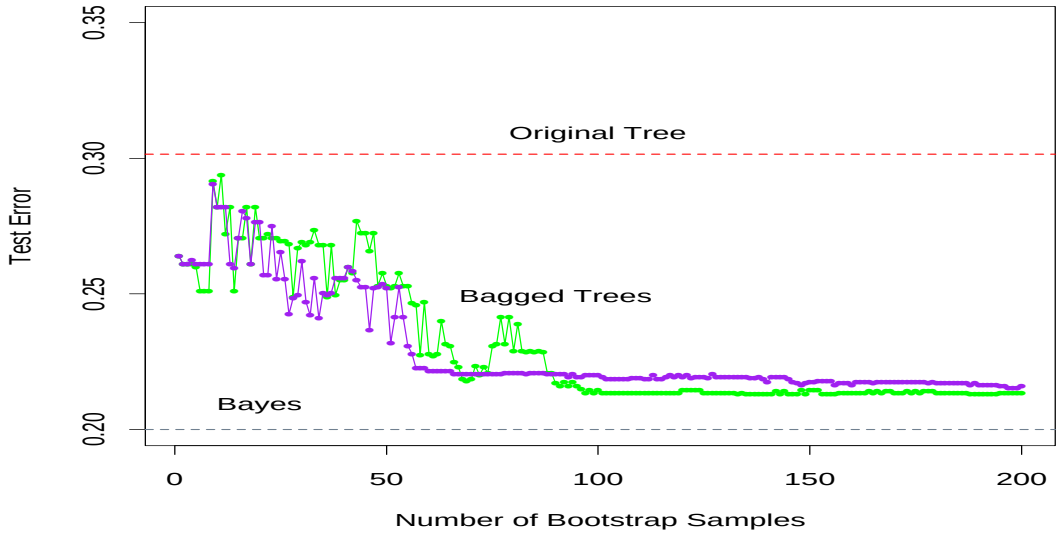


Figure 8.10: *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The green points correspond to majority vote, while the purple points average the probabilities.*

About Bagging

- Bagging can dramatically reduce the variance of unstable procedures like trees, leading to improved prediction
  - Bagging smooths out this variance and hence reducing the test error
  - Bagging can stabilize unstable procedures.
- The simple structure in the model can be lost due to bagging
  - A bagged tree is no longer a tree.
  - The bagged estimate is not easy to interpret.
- Under 0-1 loss for classification, bagging may not help due to the nonadditivity of bias and variance.

Random Forest

Random Forests (Breiman, 2001):

- Random forest is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- Bagging: random subsampling the training set
- Random subspace method (Ho, 1995, 1998): random subsampling the feature space (called "feature bagging"); To reduce the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the trees, causing them to become correlated.
- Random forest: combine bagging and random subspace method.

Learning Algorithm for Building A Tree

Denote the training size by n and the number of variables by p. Assume $m < p$ is the number of input variables to be used to determine the decision at a node of the tree.

- Randomly choose n samples with replacement (i.e. take a bootstrap sample).
- Use the rest of the samples to estimate the error of the tree, by predicting their classes.
- For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
- Each tree is fully grown and not pruned.

Properties of Random Forest

Advantages:

- highly accurate in many real examples; fast; handles a very large number of input variables.
- ability to estimate the importance of variables for classification.
- generates an internal unbiased estimate of the generalization error as the forest building progresses.
- impute missing data and maintains accuracy when a large proportion of the data are missing.
- provides an experimental way to detect variable interactions.
- can balance error in unbalanced data sets.
- compute proximities between cases, useful for clustering, detecting outliers, and (by scaling) visualizing the data
- can be extended to unlabeled data, leading to unsupervised clustering, outlier detection and data views

Properties of Random Forest

Disadvantages:

- Random forests are prone to overfitting for some data sets. This is even more pronounced in noisy classification/regression tasks.
- Random forests do not handle large numbers of irrelevant features

Implementation:

- R packages: randomForest; randomForestSRC (for survival data); grf (generalized random forest for causal effect estimation).

Asymptotics of generalized random forest: Athey et al. (2019)

Applications in Causal Inference/Optimal Treatment Decision

- Potential outcomes: $Y^*(a)$, an outcome that would result if a patient were given treatment $a \in \mathcal{A}$.
- Observed data: response, Y (larger value of Y indicates better outcome); p-dimensional covariates, $X \in \mathcal{X}$; received treatment, $A \in \mathcal{A}$.
- Consider binary treatment case: $\mathcal{A} = \{0, 1\}$.
- Average treatment effect (ATE): $\Delta = E\{Y^*(1) - Y^*(0)\}$.
- Conditional treatment effect (CTE): $\tau(x) = E\{Y^*(1)|X = x\} - E\{Y^*(0)|X = x\}$.
- Average treatment effect on the treated (ATT): $\Delta_1 = E\{Y^*(1)|A = 1\} - E\{Y^*(0)|A = 1\}$.

Assumptions for Causal Inference

A1. Consistency assumption:

$$Y = Y^*(1)A + Y^*(0)(1 - A)$$

A2. No unmeasured confounders assumption (strong ignorablility):

$$\{Y^*(1), Y^*(0)\} \perp\!\!\!\perp A \mid X$$

A3. Positivity assumption: $0 < P(A = 1|X) < 1$ for any X.

Estimation of Causal Effects

- Note that $\Delta \neq E(Y|A=1) - E(Y|A=0)$. (why?)
- In fact,

$$E\left\{\frac{I(A=1)}{\pi(X)}Y\right\} = E\{Y^*(1)\},$$

  where $\pi(X) = P(A=1|X)$.
- Under assumptions A1-A2, we have
  $\tau(x) = E(Y|A=1, X=x) - E(Y|A=0, X=x)$.
- Proof:

$$E\{Y^*(1)|X=x\} = E\{Y^*(1)|A=1, X=x\} = E\{Y|A=1, X=x\}.$$

- How about Δ_1?

$$E\{Y^*(0)|A=1\} = E\{Y^*(0)A\}/P(A=1) = E\{Y^*(0)\pi(X)\}/P(A=1).$$

Estimation of CTE

- Two ways of using random forests:
  - Method I: Fit $Y \sim RF(X, A)$. Denote the resulting estimator by $\widehat{\mu}(X, A)$. Then, $\widehat{\tau}(x) = \widehat{\mu}(x, 1) - \widehat{\mu}(x, 0)$.
  - Method II: Fit $Y \sim RF(X)$ separately for $A = 1$ and $A = 0$ groups. Denote the resulting estimators by $\widehat{\mu}_1(X)$ and $\widehat{\mu}_0(X)$, respectively. Then, $\widehat{\tau}(x) = \widehat{\mu}_1(x) - \widehat{\mu}_0(x)$.
- Method II is usually better than method I (more accurate).
- Under the assumptions A1-A2, it can be shown that the optimal treatment decision rule is estimated by $d^{opt}(x) = I\{\widehat{\tau}(x) > 0\}$.
- The decision rule $\widehat{\tau}(x)$ obtained by RF may be difficult to interpret clinically. In practice, it may prefer a simple and interpretable decision rule. We may consider a tree classification based on the labels $Z_i \equiv I\{\widehat{\tau}(X_i) > 0\}$ and features X_i.
- More accurately, we should consider a weighted classification of Z_i on X_i with the weight $w_i = |\widehat{\tau}(X_i)|$.

Estimation of ATE

- Inverse probability weighted estimation (IPW):

$$\widehat{\Delta} = \sum_{i=1}^{n} \frac{A_i}{\widehat{\pi}(X_i)} Y_i - \sum_{i=1}^{n} \frac{(1-A_i)}{1-\widehat{\pi}(X_i)} Y_i,$$

  where $\widehat{\pi}(X)$ is estimated propensity score, e.g., using logistic regression or random forest.

- IPW estimator is lack of efficiency (why?).
- Augmented inverse probability weighted estimation (AIPW):

$$\begin{aligned}\widehat{\Delta} &= \sum_{i=1}^{n} \frac{A_i}{\widehat{\pi}(X_i)} Y_i - \sum_{i=1}^{n} \left\{ \frac{A_i}{\widehat{\pi}(X_i)} - 1 \right\} \widehat{\mu}_1(X_i) \\ &\quad - \left[\sum_{i=1}^{n} \frac{1-A_i}{1-\widehat{\pi}(X_i)} Y_i - \sum_{i=1}^{n} \left\{ \frac{1-A_i}{1-\widehat{\pi}(X_i)} - 1 \right\} \widehat{\mu}_0(X_i) \right].\end{aligned}$$

Estimation of ATE/ATT

- AIPW estimator is doubly robust: $\widehat{\Delta}$ is consistent when either $\widehat{\pi}(x)$ is consistent or $\widehat{\mu}_0(x)$ and $\widehat{\mu}_1(x)$ are consistent.
- Alternative estimators of ATE (regression estimators):

$$\widehat{\Delta} = \frac{1}{n}\sum_{i=1}^{n}\{\widehat{\mu}_1(X_i) - \widehat{\mu}_0(x)\},$$

$$\widehat{\Delta} = \frac{1}{n}\sum_{i=1}^{n}\Big[A_i\{Y_i - \widehat{\mu}_0(X_i)\} + (1-A_i)\{\widehat{\mu}_1(X_i) - Y_i\}\Big].$$

- Estimator of ATT:

$$\widehat{\Delta}_1 = \frac{\sum_{i=1}^{n} A_i Y_i}{\sum_{i=1}^{n} A_i} - \frac{\sum_{i=1}^{n} \frac{(1-A_i)\widehat{\pi}(X_i)}{1-\widehat{\pi}(X_i)} Y_i}{\sum_{i=1}^{n} A_i}.$$

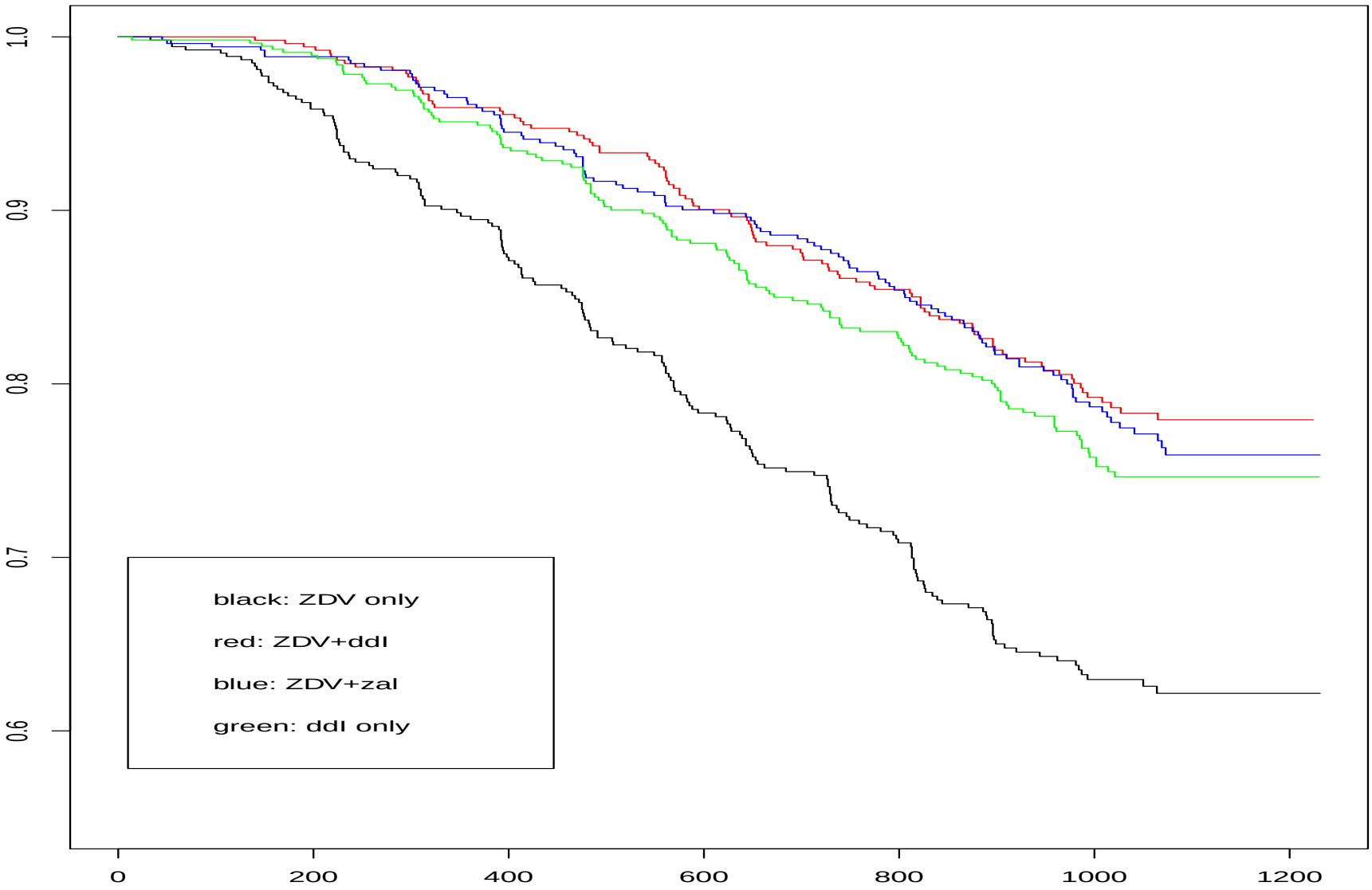
$$\widehat{\Delta}_1 = \sum_{i=1}^{n} A_i\{Y_i - \widehat{\mu}_0(X_i)\} / \sum_{i=1}^{n} A_i.$$

AIDS Data Example

AIDS Clinical Trials Group Study 175 (ACTG175)

- HIV-infected patients with CD4 counts 200 $\sim$ 500/mm^3
- Randomized to four treatment groups:
  - zidovudine alone (ZDV)
  - zidovudine plus didanosine (ZDV+ddI)
  - zidovudine plus zalcitabine (ZDV+zal)
  - didanosine alone (ddI)
- 12 baseline clinical covariates, such as age (years), weight (kg), Karnofsky score (scale of 0-100), CD4 count (cells/mm^3) at baseline
- Primary endpoints of interest: (i) CD4 count at 20 ± 5 weeks post-baseline; (ii) the first time that a patient had a decline in their CD4 cell count of at least 50%, or an event indicating progression to AIDS, or death.

Survival Curves



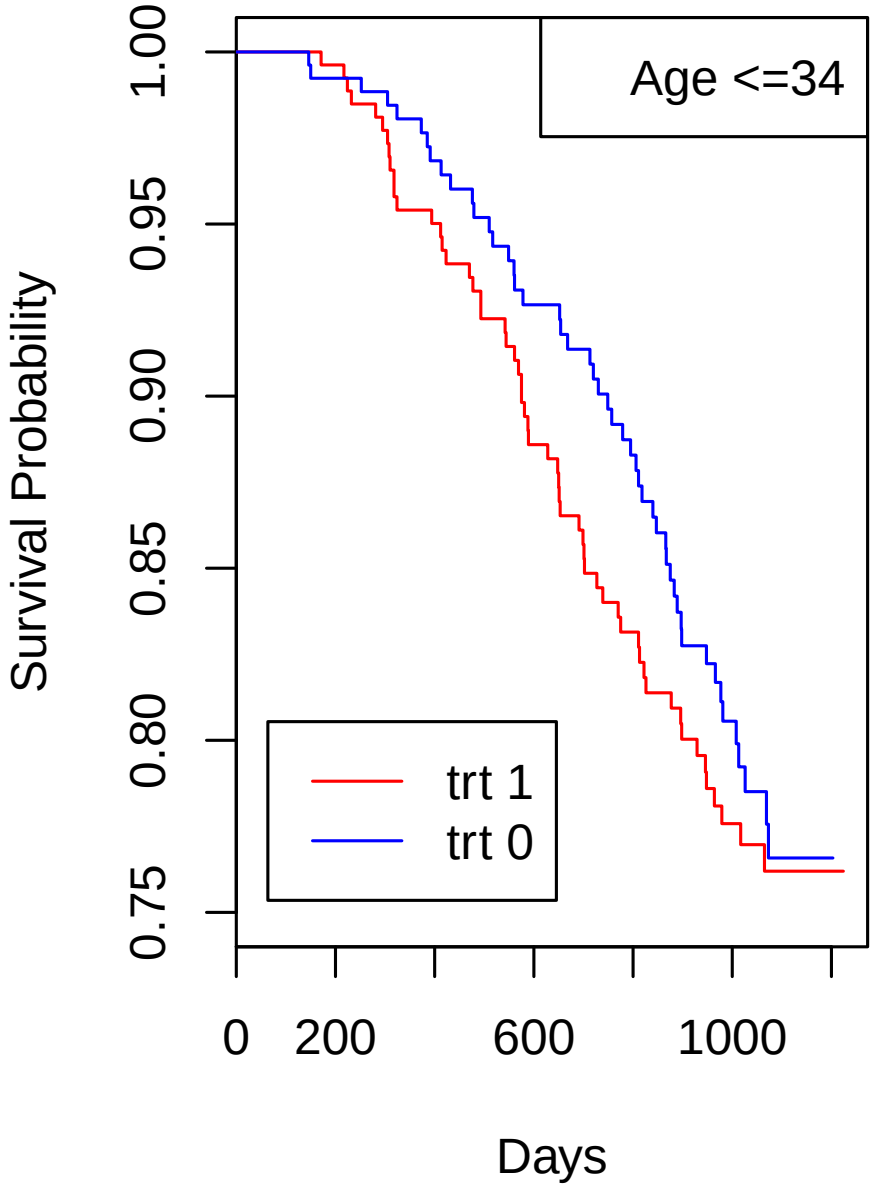
A Closer Look of Top 2 Treatments

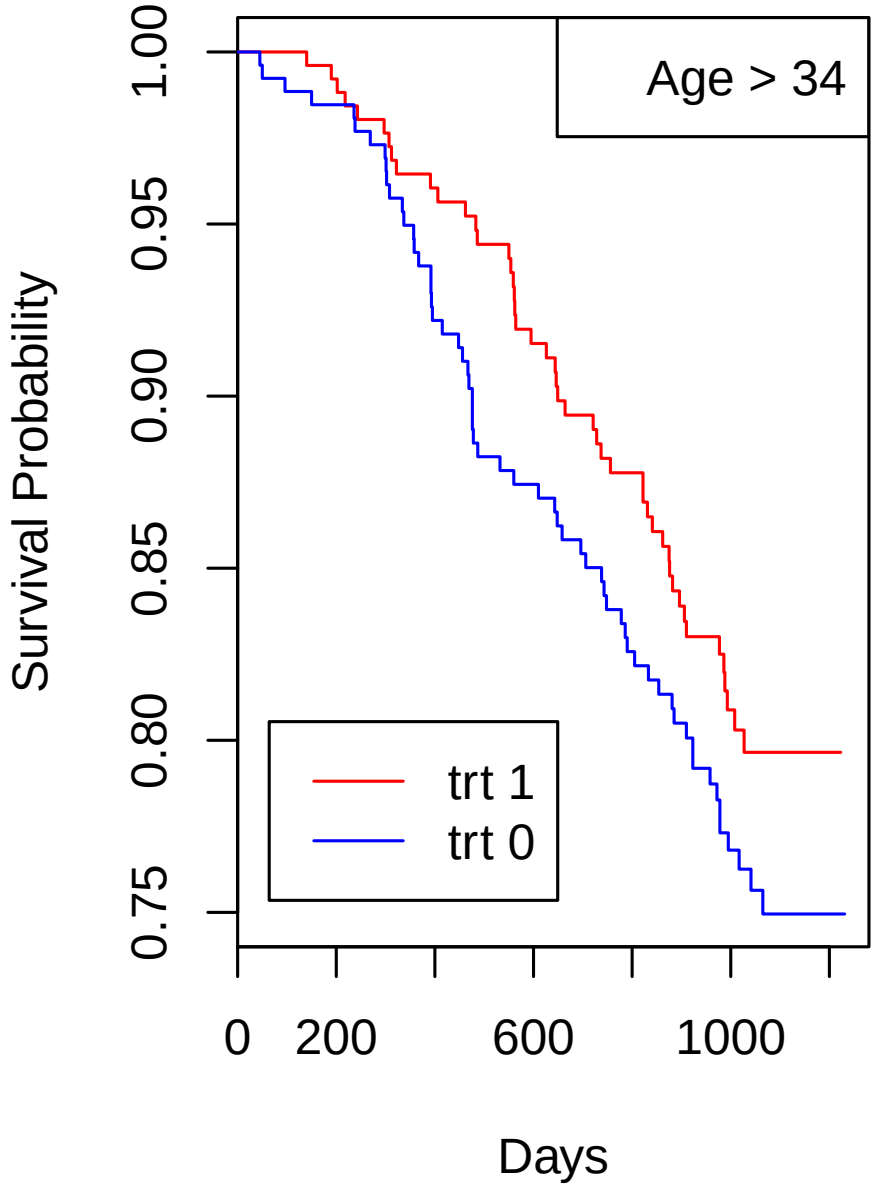
- A=1: ZDV + ddI
- A=0: ZDV + zal

	Age $\leq$ 34	Age $>$ 34	
A = 1	266	256	522
A = 0	263	261	524
	529	517	1046

Note: Age 34 is the median age for the patients

Heterogeneous Treatment Effects





Random Forest Estimation of CTE

```
library(survival)

a = read.table("C:\\Users\\wlu4\\all.dat",sep=" ")
b = read.table("C:\\Users\\wlu4\\events.dat")
ix = match(as.numeric(b[,1]), as.numeric(a[,1]))
trt = as.numeric(b[!is.na(ix),4])

###Kaplan-Meier curve
t.time = as.numeric(b[!is.na(ix),3])
delta = as.numeric(b[!is.na(ix),2])

km.fit = survfit(Surv(t.time,delta)~trt)
plot(km.fit,ylim=c(0.55,1),col = c("black","red","blue",
"green") )
legend(10, .7, c("black: ZDV only", "red: ZDV+ddI",
"blue: ZDV+zal","green: ddI only"))
```

Random Forest Estimation of CTE

```
####consider ZDV + ddI & ZDV + zal groups
ix1 = (trt == 1)|(trt==2)
Y = as.numeric(a[ix1,20])
A = as.numeric(trt[ix1]==1)
X = as.matrix(a[ix1,c(2,3,7,19,23,4,5,6,12,13,14,16)])
###random forest fit
library(randomForestSRC)
Y1 = Y[A==1]
X1 = X[A==1,]
data1 = data.frame(cbind(Y1,X1))
fit1 = rfsrc(Y1~.,data=data1)

Y0 = Y[A==0]
X0 = X[A==0,]
data0 = data.frame(cbind(Y0,X0))
fit0 = rfsrc(Y0~.,data=data0)
```

Random Forest Estimation of CTE

```
mu11 = predict(fit1)
mu10 = predict(fit1,data0)
mu01 = predict(fit0,data1)
mu00 = predict(fit0)

## $predicted.oob: out of bag prediction
tau1 = mu11$predicted.oob - mu01$predicted
tau0 = mu10$predicted - mu00$predicted.oob

tau = numeric(length(Y))
tau[A==1] = tau1
tau[A==0] = tau0
cr = sum((tau > 0)*(Y > 0))/length(Y) #correct decision rate
```

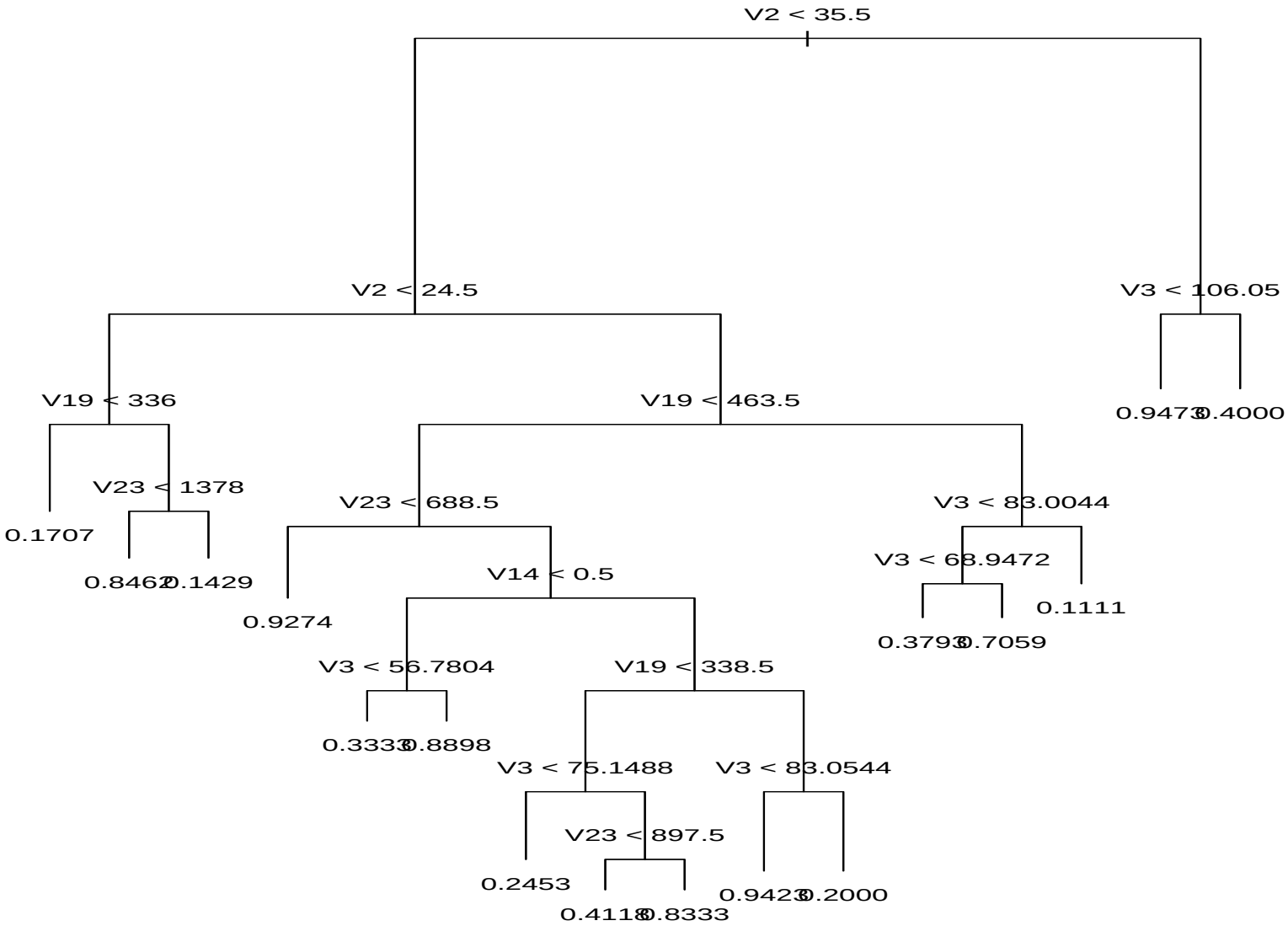
OTR Estimation Using Tree

```
## Tree classification for optimal treatment rule
library(tree)
Z = as.numeric(tau > 0)
data2 = data.frame(cbind(Z,X))

## fit the original tree with split="deviance"
tree_otr=tree(Z~., data=data2, split="deviance")
## tree summary
summary(tree_otr)
plot(tree_otr)
text(tree_otr)
```

OTR Estimation Using Tree

```
## 10-fold CV for parameter tuning
set.seed(11)
tree_cv = cv.tree(tree_otr,,prune.tree,K=10)
min_idx = which.min(tree_cv$dev)
min_idx
tree_cv$size[min_idx]

tree_otr_prune = prune.tree(tree_otr, best = 12)
plot(tree_otr_prune)
text(tree_otr_prune)
```

Original Tree Using Deviance



After Parameter Tuning based on 10-CV