STAT406- Methods of Statistical Learning Lecture 10

Matias Salibian-Barrera

UBC - Sep / Dec 2018

Curse of dimensionality

- Suppose we have $n = 100$ observations uniformly distributed on the interval $[0, 1]$.
- How many do we expect to find in $[0.25, 0.75]$?

$$0.25 \leq X_i \leq 0.75$$

$$|X_i - 0.5| \leq 0.25$$

Curse of dimensionality

- Suppose we have $n = 100$ observations uniformly distributed on the square $[0, 1] \times [0, 1]$.
- How many do we expect to find in the square $[0.25, 0.75] \times [0.25, 0.75]$?

Curse of dimensionality

- Suppose we have $n = 100$ observations uniformly distributed on the hypercube $[0, 1]^{10}$.
- How many do we expect to find in the hypercube $[0.25, 0.75]^{10}$?

Curse of dimensionality

- How many observations uniformly distributed on the hypercube $[0,1]^{20}$ are needed to expect to find at least 50 observations in the hypercube $[0.25,0.75]^{20}$?
- Ans:

Curse of dimensionality

- Suppose we have $n = 10,000$ observations uniformly distributed on the hypercube $[0, 1]^{20}$
- How large should a be so that we can expect to find at least 50 observations in the hypercube $[0.5 - a, 0.5 + a]^{20}$?

What can we do?

- How can we build flexible predictors when there are many covariates available?
- Approximate the regression function by a piecewise constant function
- Use an iterative algorithm to build the piecewise function
- Suboptimal, but feasible

Regression trees

- Consider data $(Y_i, \mathbf{X}_i)$, $i = 1, \ldots, n$ with $\mathbf{X}_i \in \mathbb{R}^p$
- Find regions R_1, R_2, ..., R_K that minimize

$$\sum_{j=1}^{K} \sum_{i \in R_j} \left(Y_i - \hat{\mu}_j\right)^2$$

where $\hat{\mu}_j$ is the average of the Y_i's for which $\mathbf{X}_i \in R_j$

Regression trees

- A simpler search
- Find a feature X_j and a threshold a such that

$$\sum_{i \in R_L} (Y_i - \hat{\mu}_L)^2 + \sum_{i \in R_R} (Y_i - \hat{\mu}_R)^2$$

is minimized, where

$$R_L = \{X_j < a\} \quad R_R = \{X_j \geq a\}$$

Regression trees

- Recursively split the regions R_L and R_R
- Stopping criteria?
- Regions have few observations
- The gain in RSS is below a threshold

Regression trees

- It is relatively easy to find the optimal splits
- Trees are easy to explain and visualize
- In some cases trees are interpretable

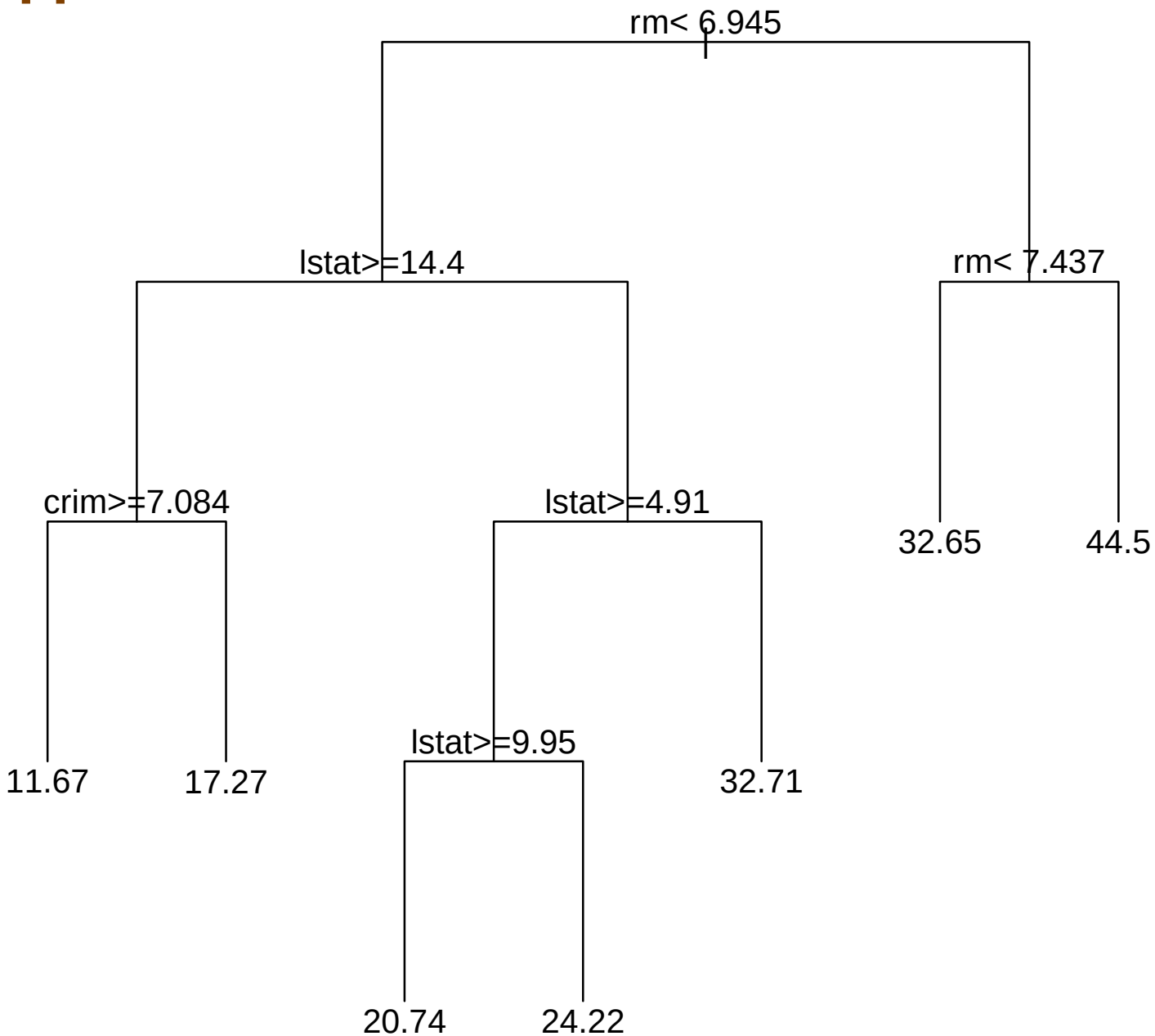
Regression trees - Example

- Consider the Boston data set
- $n = 506$, $p = 14$
- Create a training and test set ($n = 380$ and $n = 126$)
- Build a regression tree

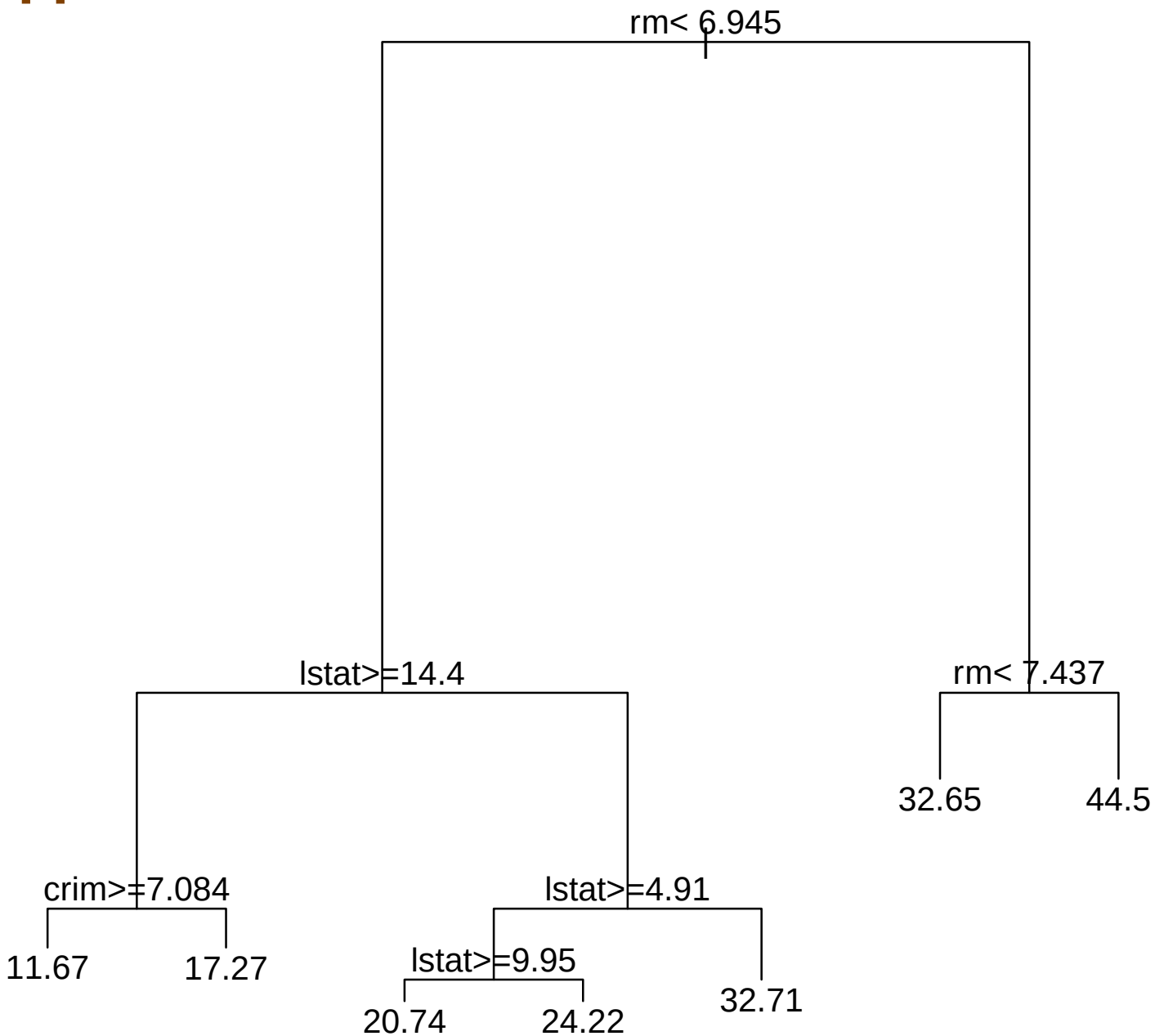
Regression trees - Example

```
data(Boston, package='MASS')

set.seed(123456)
n <- nrow(Boston)
ii <- sample(n, floor(n/4))
dat.te <- Boston[ ii, ]
dat.tr <- Boston[ -ii, ]

bos.t <- rpart(medv ~ ., data=dat.tr,
                    method='anova')
plot(bos.t, uniform=FALSE)
text(bos.t, pretty=TRUE)
```

Boston



Boston



Boston Example

Compare prediction errors with those of a standard linear regression model

```
> # predictions on the test set
> pr.t <- predict(bos.t, newdata=dat.te,
    type='vector')
> mean((dat.te$medv - pr.t)^2)
[1] 24.43552
>
> # full linear model
> bos.lm <- lm(medv ~ ., data=dat.tr)
> pr.lm <- predict(bos.lm, newdata=dat.te)
> mean((dat.te$medv - pr.lm)^2)
[1] 26.60311
```

Boston Example

Use stepwise to get a better linear model

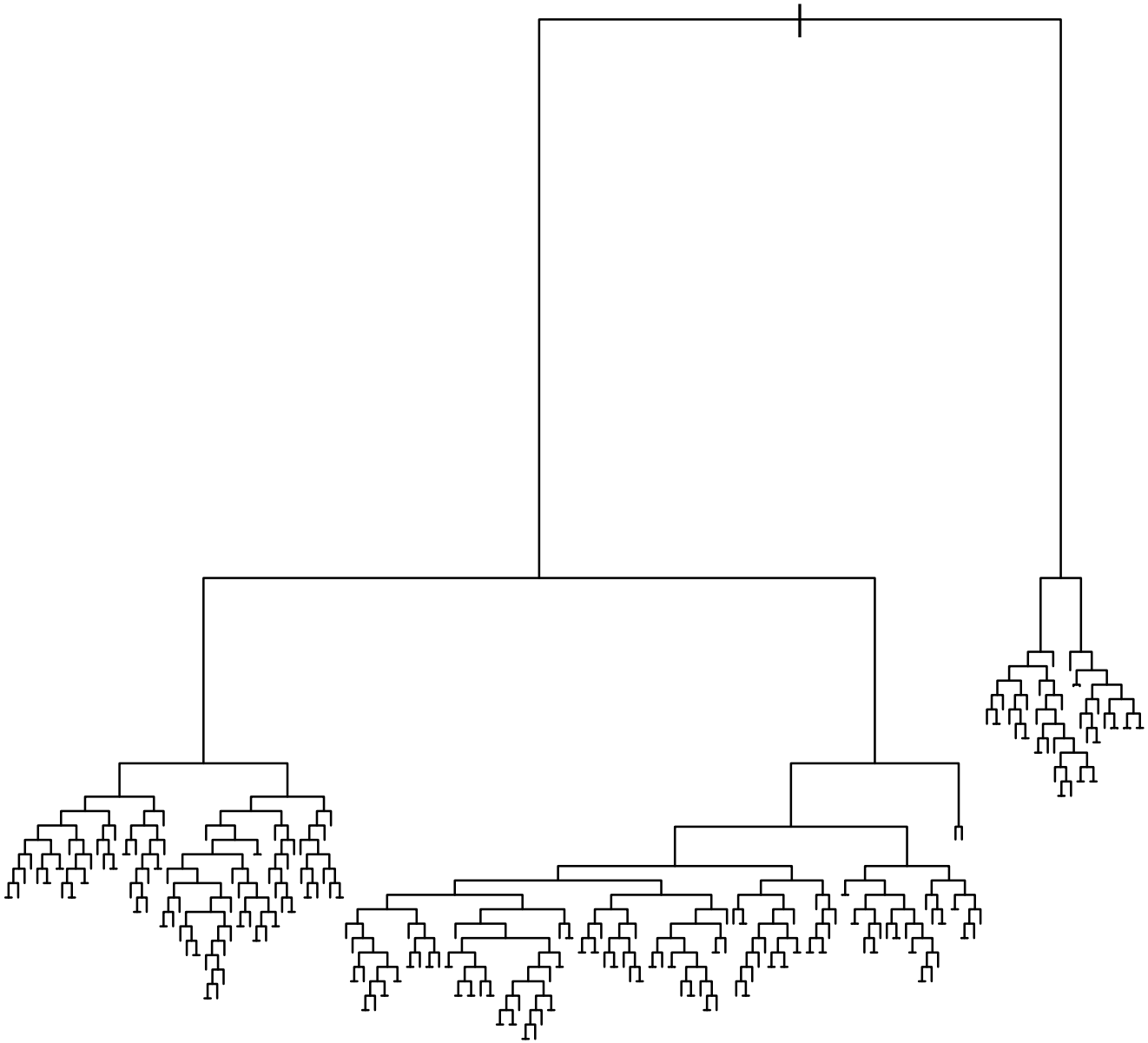
```
> # try to make it better
> null <- lm(medv ~ 1, data=dat.tr)
> full <- lm(medv ~ ., data=dat.tr)
> bos.aic <- stepAIC(null,
    scope=list(lower=null, upper=full),
    trace=FALSE)
> pr.aic <- predict(bos.aic,
    newdata=dat.te)
> with(dat.te, mean( (medv - pr.aic)^2 ) )
[1] 25.93452
```

Boston Example

Use LASSO

```
> set.seed(123)
> bos.la <- cv.glmnet(x=x.tr, y=y.tr,
     alpha=1)
> x.te <- as.matrix(dat.te[,-14])
> pr.la <- predict(bos.la,
     s='lambda.1se', newx=x.te)
> with(dat.te, mean((medv - pr.la)^2))
[1] 29.20216
```

Overfitting...



Boston Example

Not surprisingly, when we overfit...

```
> pr.to <- predict(bos.to,
      newdata=dat.te,
      type='vector')
> with(dat.te, mean((medv - pr.to)^2))
[1] 36.51097
```

Pruning...

- Cost pruning

$$\min_{T \subset T_0} \sum_{m=1}^{|T|} \sum_{\mathbf{x}_i \in R_m} (y_i - \hat{\mu}_m)^2 + \alpha |T|$$

- We can compute the solution for all α
- Compare each subtree in this sequence using CV
- Pick the best subtree

Pruning...

- More specifically:
- Let $T_\ell \subset T_0$ be the solution to

$$\min_{T \subset T_0} \sum_{m=1}^{|T|} \sum_{\mathbf{x}_i \in R_m} (y_i - \hat{\mu}_m)^2 + \alpha |T|$$

when

$$\alpha \in [\alpha_\ell, \alpha_{\ell+1}) \subseteq [0, +\infty) \quad \ell = 1, 2, \ldots, L$$

Pruning...

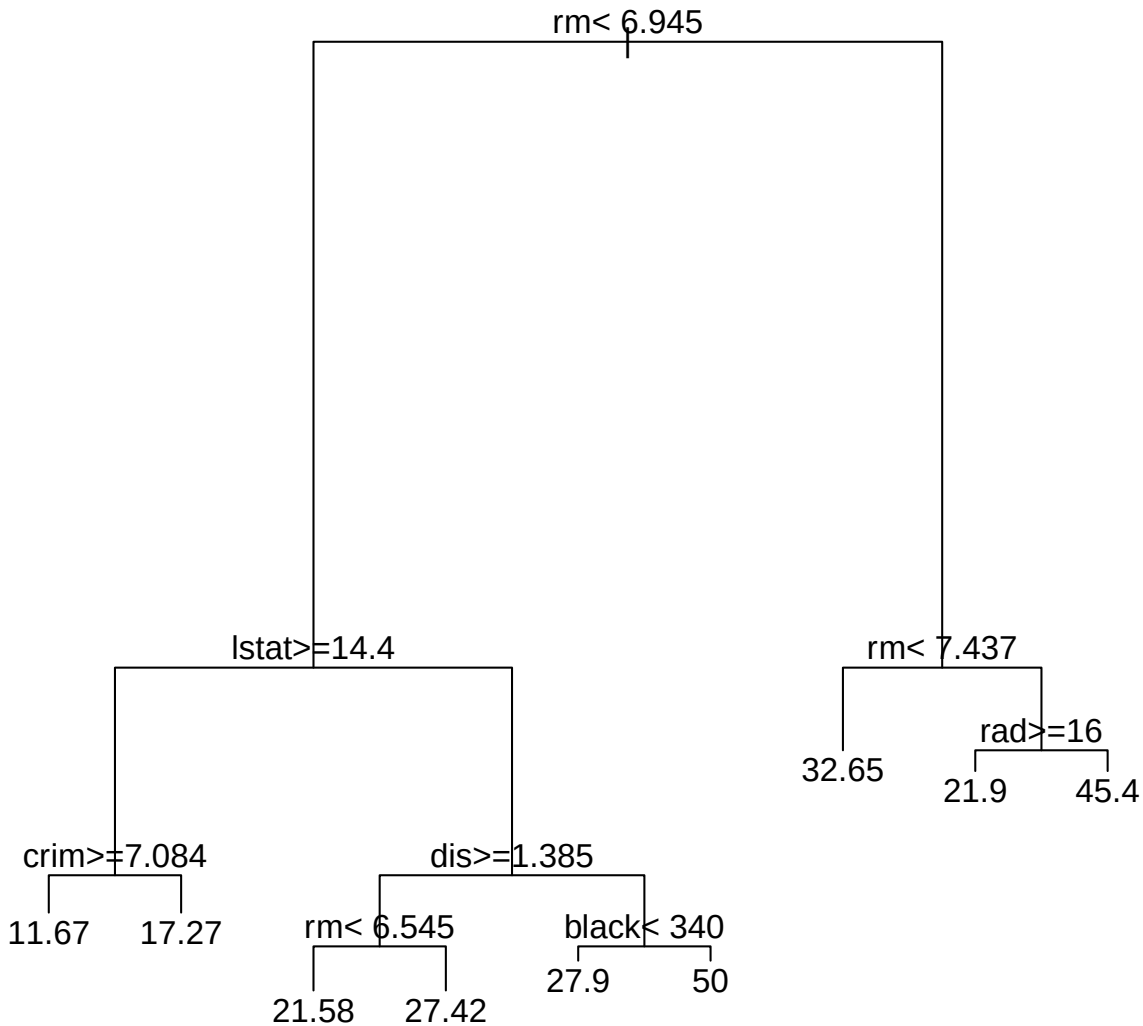
- Split the data into K folds
- For $j = 1, \ldots, K$
  - Build a tree without using the j-th fold
  - Prune it with penalties α_ℓ, $\ell = 1, \ldots, L$
  - Use these L trees to predict the j-th fold
  - Record the prediction errors.
- Sum or average over the folds.
- We obtain K-fold CV-estimated

Boston Example

Pruning works...

```
> b <- ***cp with minimum xerror***
>
> bos.t3 <- prune(bos.to, cp=b)
> plot(bos.t3)
> pr.t3 <- predict(bos.t3,
          newdata=dat.te,
          type='vector')
> with(dat.te, mean((medv - pr.t3)^2))
[1] 18.96988
```

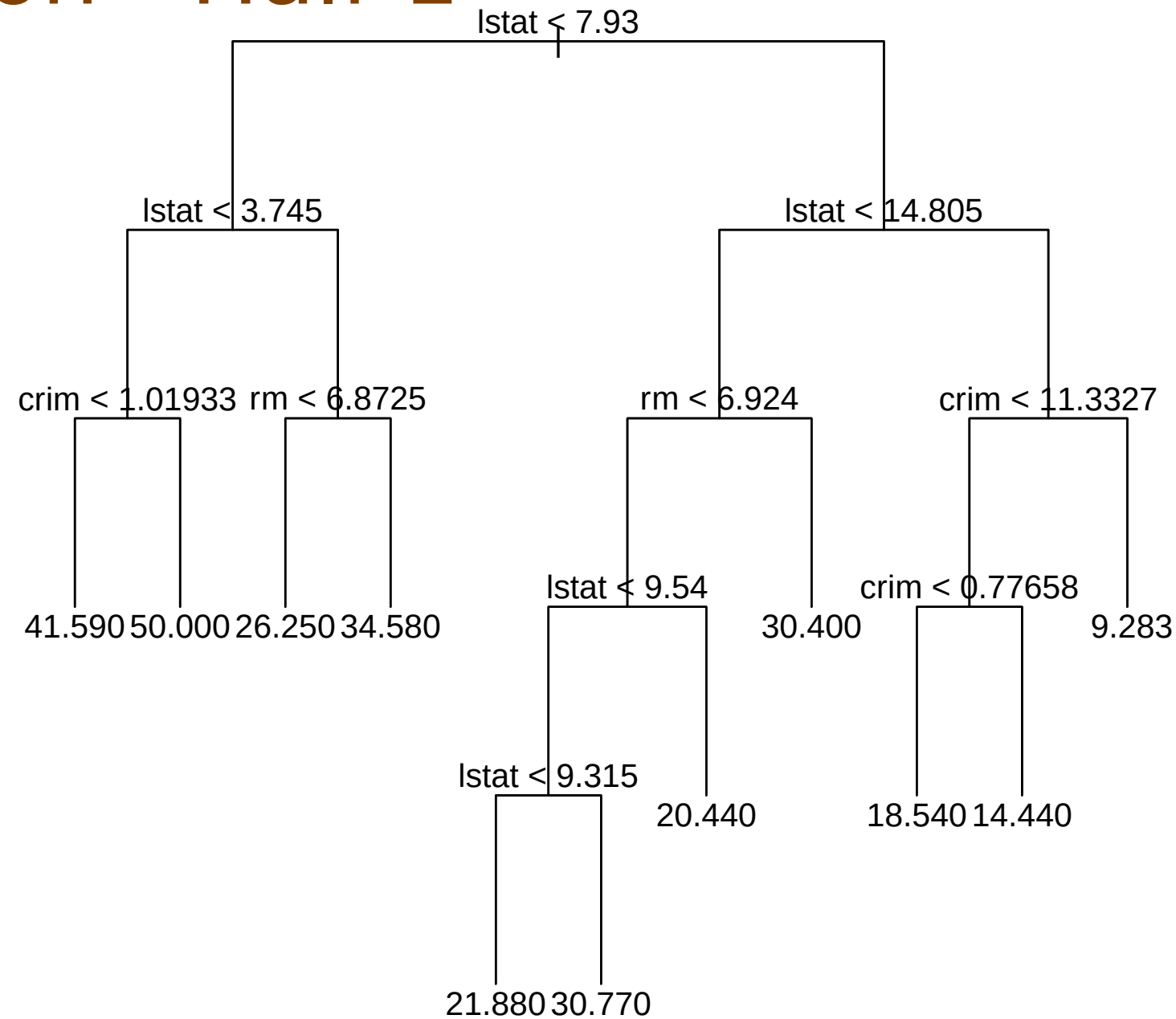
Pruned tree



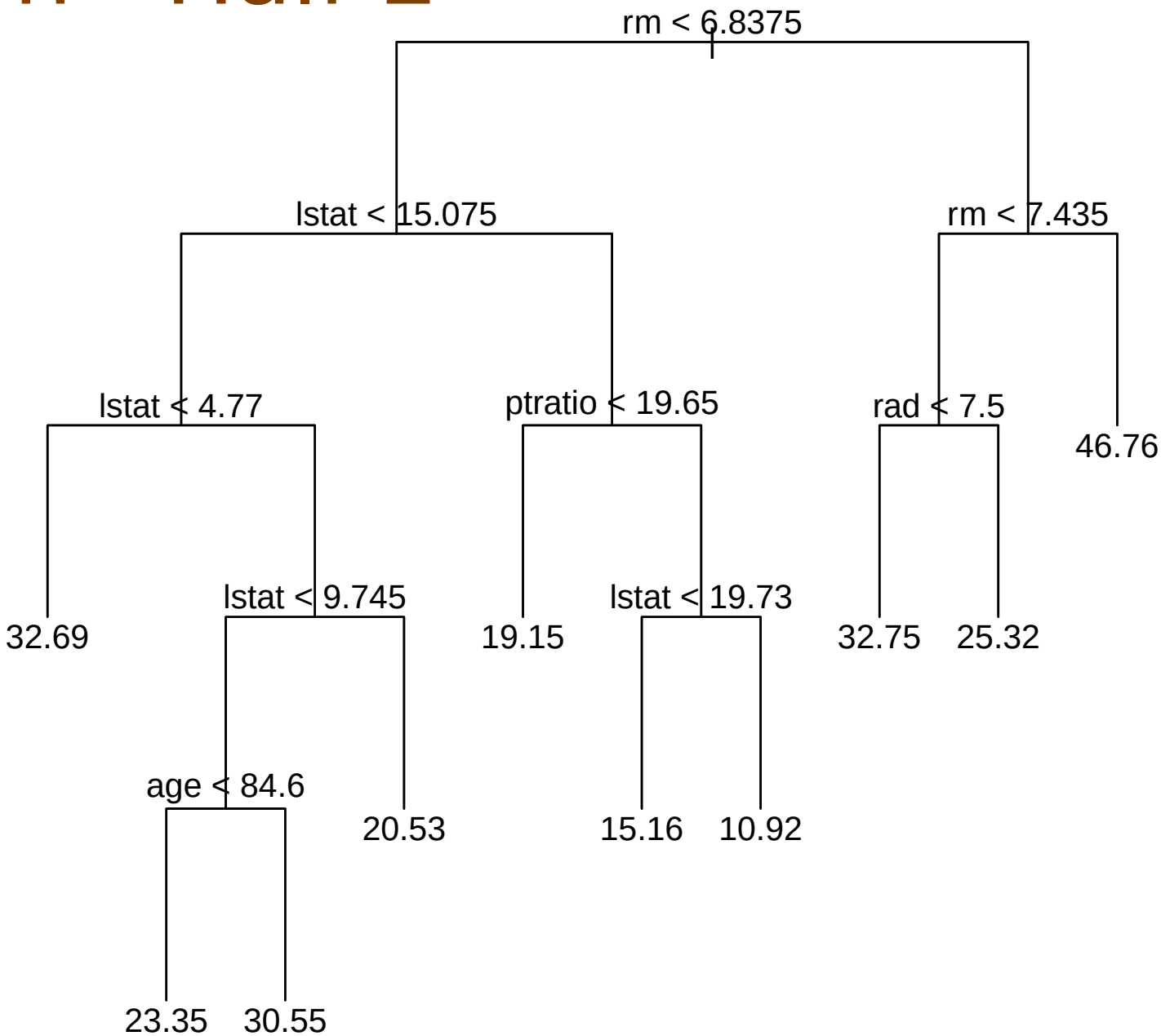
Bagging

- Trees can be highly variable
- Trees computed on samples from the sample population can be quite different from each other
- For example, we split the Boston data in two...

Boston - Half 1



Boston - Half 2



Bagging

- Linear regression, for example, is not so variable
- Estimated coefficients computed on the same two halfs

```
     (Intercept)  crim   zn indus chas
[1,]       39.21 -0.13 0.04  0.04 2.72
[2,]       33.12 -0.10 0.05 -0.01 2.80
              nox   rm age   dis  rad   tax
[1,]       -20.07 3.45   0 -1.44 0.28 -0.01
[2,]       -14.18 4.15   0 -1.46 0.34 -0.02
       ptratio black lstat
[1,]   -1.01  0.01 -0.56
[2,]   -0.90  0.01 -0.50
```

Bagging

- If we could average many trees trained on independent samples from the same population, we would obtain a predictor with lower variance

- If $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_B$ are B regression trees, then their average is

$$\hat{f}_{\text{av}}(\mathbf{x}) = \frac{1}{B}\sum_{j=1}^{B} \hat{f}_j(\mathbf{x})$$

Bagging

- However, we generally do not have B training sets...
- We can **bootstrap** the training set to obtain B pseudo-new-training sets
- Let $(Y_1, \mathbf{X}_1)$, $(Y_2, \mathbf{X}_2)$, $\ldots$, $(Y_n, \mathbf{X}_n)$ be the training sample, where

$$(Y_j, \mathbf{X}_j) \sim F_0$$

Bagging

- If we knew F_0, then we could generate / simulate new training sets, and average the resulting trees...
- We do not know F_0, but we have an estimate for it
- Let F_n be the empirical distribution of our only training set $(Y_1, \mathbf{X}_1)$, $(Y_2, \mathbf{X}_2)$, $\ldots$, $(Y_n, \mathbf{X}_n)$

Bagging

- We know that

$$F_n \xrightarrow[n \to \infty]{} F_0$$

  (in what sense?)

- Bootstrap generates / simulates samples from F_n
- Taking a sample of size n from F_n is the same as sampling with replacement from the training set $(Y_1, \mathbf{X}_1), (Y_2, \mathbf{X}_2), \ldots, (Y_n, \mathbf{X}_n)$

Bagging

- To apply bagging to a regression tree, take B independent samples (with replacement) from the training set
- Obtain the B trees: $\hat{f}_1^*, \hat{f}_2^*, \ldots, \hat{f}_B^*$
- and average their predictions

$$\hat{f}_{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{j=1}^{B} \hat{f}_j^*(\mathbf{x})$$

Bagging

- Generally, we apply bagging on “large” trees, without pruning them (try to retain their low-bias and reduce their variance by averaging)
- With the Boston data set, if we apply bagging to the regression tree computed on the training set, and then use it to predict on the test set, we obtain:

Bagging

- $B = 5$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 13.89539
```

- $B = 100$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 12.62785
```

- $B = 500$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 12.08049
```

Bagging

- $B = 2000$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 11.87869
```

- $B = 5000$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 11.77328
```

Bagging

- This approach applies to any predictor (not only trees)
- It will be particularly useful for low-bias / high-variance predictors