STAT406- Methods of Statistical Learning Lecture 12

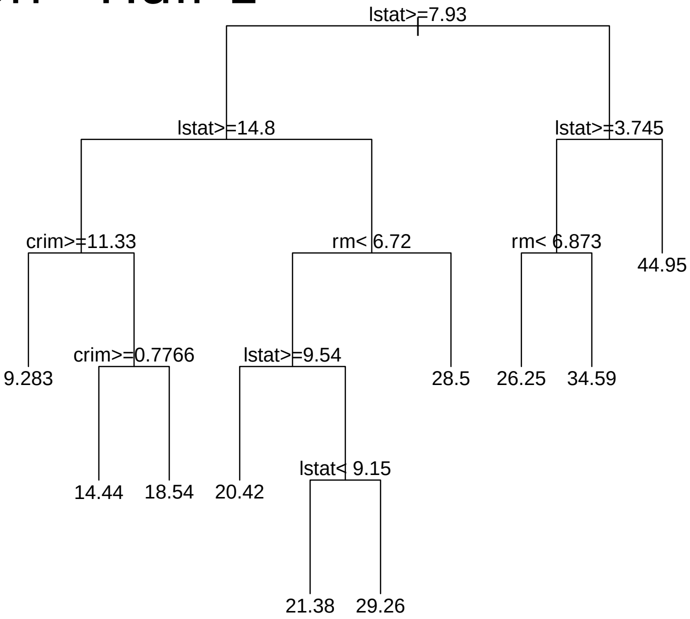
Matias Salibian-Barrera

UBC - Sep / Dec 2017

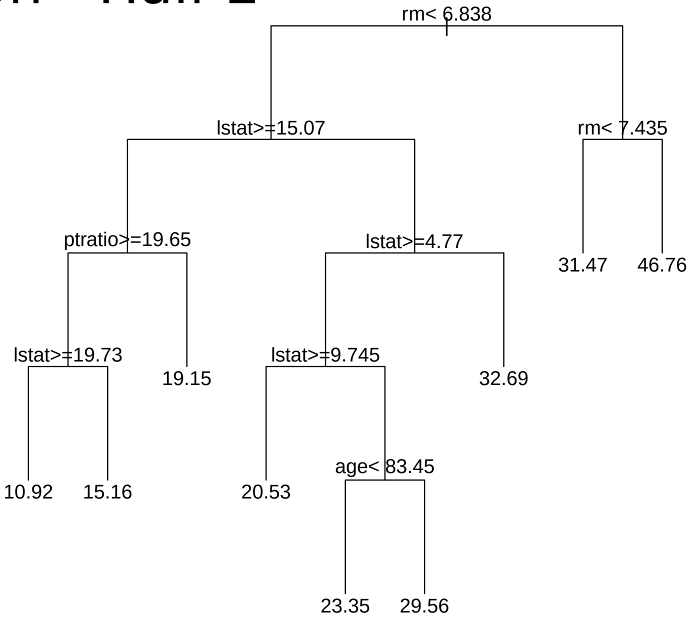
Bagging

- 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

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

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

- $B = 1$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 16.44972
```

- $B = 5$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 15.12332
```

- $B = 100$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 12.30543
```

- $B = 500$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 12.32504
```

Bagging

- $B = 2000$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 11.8116
```

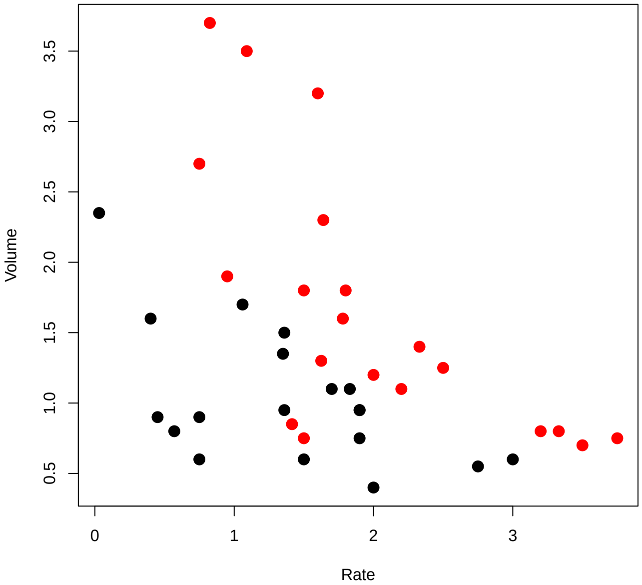
- $B = 5000$

```
> mean((dat.te$medv - pr.ba)^2)
[1] 11.85943
```

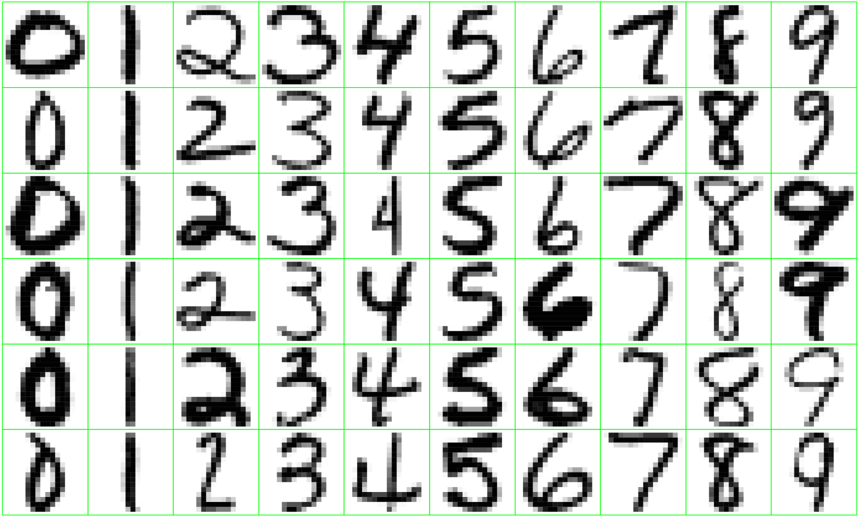
Bagging

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

Classification



Predict hand-written digits



Classification as prediction

- In general, we have n observations (training)
- $(g_1, \mathbf{x}_1)$, $(g_2, \mathbf{x}_2)$, ..., $(g_n, \mathbf{x}_n)$
- we would like to build a classifier, a function $\hat{g}(\mathbf{x})$ to predict the true class g of a future observation $(g, \mathbf{x})$ (for which g is unknown)

Classification as prediction

- In general, there are K possible classes, c_1, c_2, ..., c_K. In other words $g \in \{c_1, c_2, \ldots, c_K\}$
- Consider the following loss function

$$L(a,b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases}$$

Classification as prediction

- Find a classifier $\hat{g}(\mathbf{x})$ such that

$$E_{(G,\mathbf{X})}\left[L\left(G,\hat{g}(\mathbf{x})\right)\right] \;\leq\; E_{(G,\mathbf{X})}\left[L\left(G,h(\mathbf{X})\right)\right]$$

for any other function h

$$\begin{aligned} E_{(G,\mathbf{X})}\left[L\left(G,\hat{g}(\mathbf{X})\right)\right] &= E_{\mathbf{X}}\left\{E_{G|\mathbf{X}}\left[L\left(G,\hat{g}(\mathbf{X})\right)\right]\right\} \\ &= E_{\mathbf{X}}\left\{\sum_{j=1}^{K} L\left(c_j,\hat{g}(\mathbf{X})\right) P\left(G=c_j \,\middle|\, \mathbf{X}\right)\right\} \end{aligned}$$

Classification as prediction

- It is sufficient to find $\hat{g}(\mathbf{X})$ that minimizes

$$\sum_{j=1}^{K} L\left(c_j, \hat{g}(\mathbf{X})\right) P\left(G = c_j \middle| \mathbf{X}\right)$$

$$= \sum_{c_j \neq \hat{g}(\mathbf{X})} P\left(G = c_j \middle| \mathbf{X}\right)$$

$$= 1 - P\left(G = \hat{g}(\mathbf{X}) \middle| \mathbf{X}\right)$$

- Hence, the optimal classifier satisfies

$$P\left(G = \hat{g}(\mathbf{X}) \middle| \mathbf{X}\right) \geq P\left(G = c_j \middle| \mathbf{X}\right) \quad \text{for all } c_j$$

More than 2 groups

- In other words, $\hat{g}(\mathbf{X})$ should be the class with the highest probability

$$\hat{g}(\mathbf{X}) \; = \; \arg \max_{\mathbf{g} \in \{c_1, \ldots, c_K\}} P\Big(G = \mathbf{g} \,|\, \mathbf{X} \Big)$$

- "Assign **X** to the class with largest posterior probability given **X**"