The Mathematics of Machine Learning 2026

Lecture 6

Tim van Erven

University of Amsterdam

Outline

Decision Tree Estimation with CART

1. General Setup
2. Growing
3. Pruning

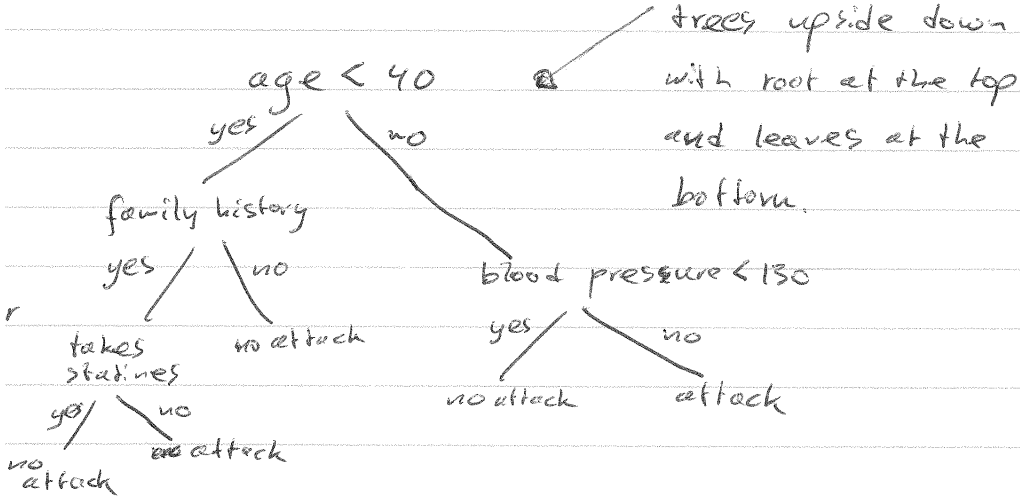
Bias-Variance Trade-off in Regression

Bagging and Random Forests

4. Bagging
5. Random Forests

General Setup

Decision tree for classification of heart attack next year:

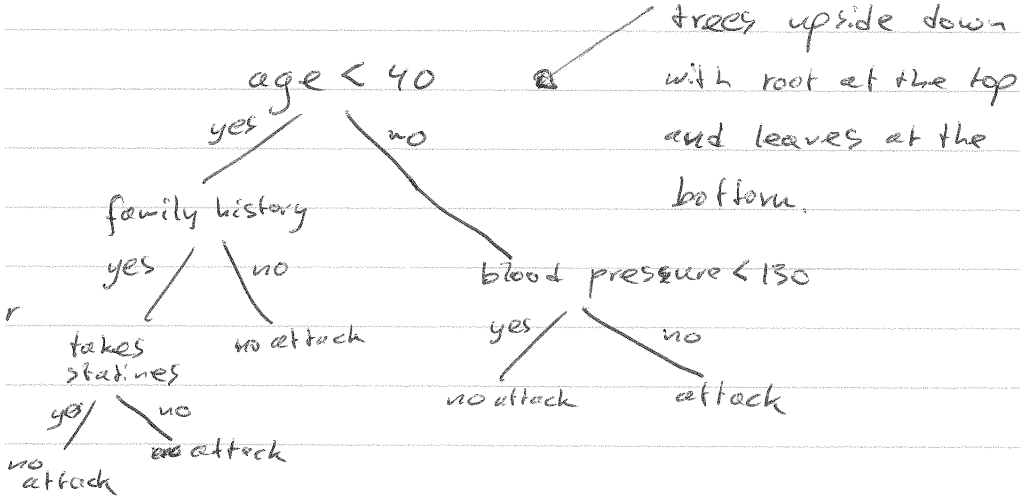


Advantages:

- Easy to interpret (if tree is not too large)
- Nice algorithms

General Setup

Decision tree for classification of heart attack next year:



Advantages:

- Easy to interpret (if tree is not too large)
- Nice algorithms

Disadvantages: (will be addressed later)

- Not competitive with logistic regression/kernel methods in terms of prediction accuracy
- Instability of training procedure:
  - Small changes in data lead to very different tree
  - So even if we can interpret a specific tree, we should not overinterpret it to be the only possible tree

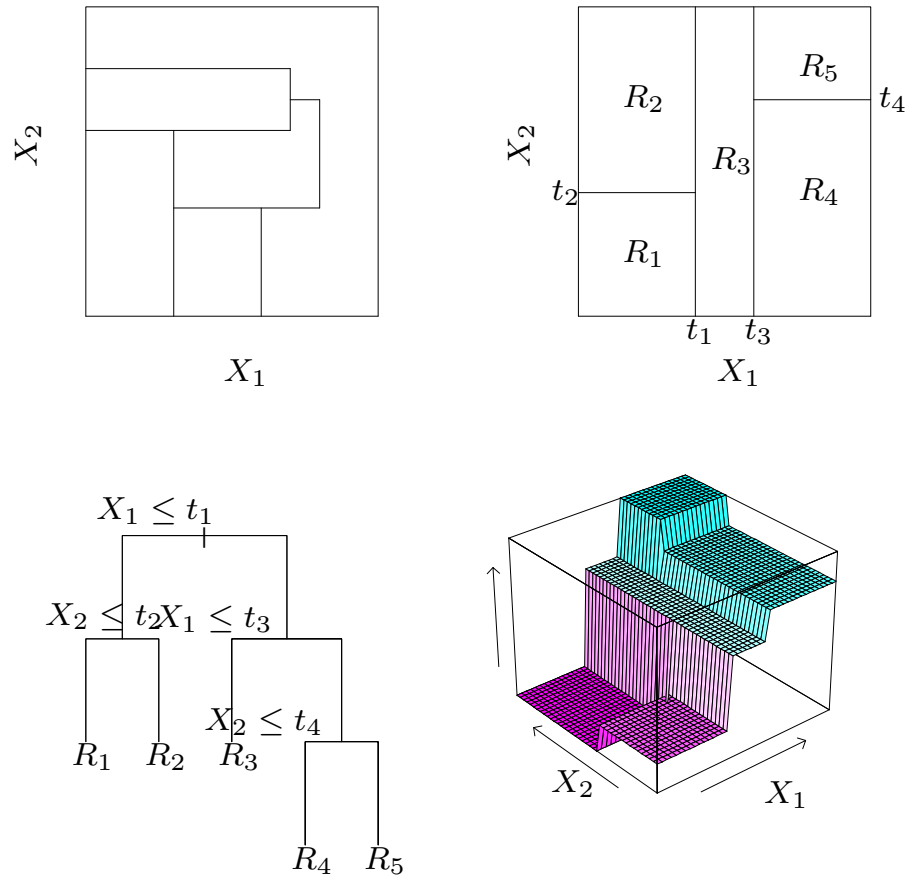
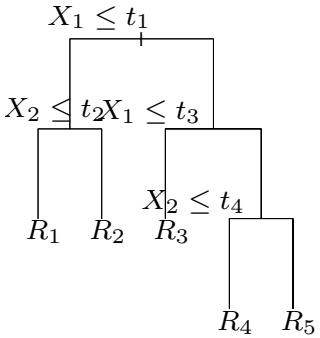


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

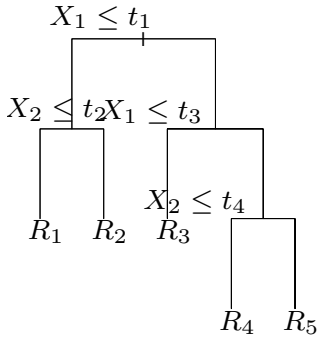
Estimation in the Leaves of the Tree



Estimate class or regression value independently per region R_m with ERM:

$$\hat{c}_m = \arg\min_c \sum_{i:X_i \in R_m} L(Y_i, c)$$

Estimation in the Leaves of the Tree



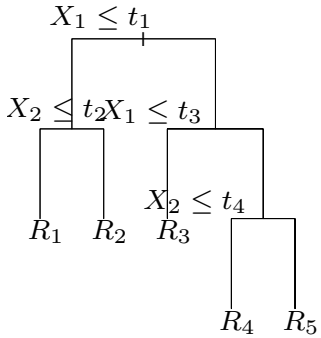
Estimate class or regression value independently per region R_m with ERM:

$$\hat{c}_m = \arg\min_c \sum_{i:X_i \in R_m} L(Y_i, c)$$

Regression:

$L(Y, c) = (Y - c)^2$ "squared error" $\Rightarrow$ $\hat{c}_m$ is average of Y_i in R_m

Estimation in the Leaves of the Tree



Estimate class or regression value independently per region R_m with ERM:

$$\hat{c}_m = \arg\min_c \sum_{i:X_i \in R_m} L(Y_i, c)$$

Regression:

$L(Y, c) = (Y - c)^2$ "squared error" $\Rightarrow$ $\hat{c}_m$ is average of Y_i in R_m

Classification:

1. **Deterministic:** 0/1-loss $\Rightarrow$ $\hat{c}_m$ is most common class in R_m
2. **Probabilistic** (for two classes for simplicity), $\hat{c}_m$ estimates $P(Y = +1 \mid X \in R_m)$:

$$\text{log-loss:} \qquad L(Y, c) = \begin{cases} -\log c & \text{if } Y = +1 \\ -\log(1 - c) & \text{if } Y = -1 \end{cases}$$

Probabilistic Estimates and Entropy

Probabilistic for two classes, $\hat{c}_m$ estimates $P(Y = +1 \mid X \in R_m)$:

$$
\begin{aligned}
\hat{c}_m &= \arg\min_c \sum_{i:X_i \in R_m} L(Y_i, c) \\
&= \arg\min_c \sum_{i:X_i \in R_m, Y_i = +1} -\log c + \sum_{i:X_i \in R_m, Y_i = -1} -\log(1-c) \\
&= \arg\min_c -n_{+1} \log c - n_{-1} \log(1-c) \\
&= \arg\min_c -\hat{p}_{m,+1} \log c - \hat{p}_{m,-1} \log(1-c) \\
&= \hat{p}_{m,+1}
\end{aligned}
$$

where

$$
n_k = |\{i : X_i \in R_m, Y_i = k\}|, \qquad \hat{p}_{m,k} = \frac{n_k}{n_{-1} + n_{+1}}
$$

Probabilistic Estimates and Entropy

Probabilistic for two classes, $\hat{c}_m$ estimates $P(Y = +1 \mid X \in R_m)$:

$$\hat{c}_m = \arg\min_c -\hat{p}_{m,+1} \log c - \hat{p}_{m,-1} \log(1 - c) = \hat{p}_{m,+1}$$

where

$$n_k = |\{i : X_i \in R_m, Y_i = k\}|, \qquad \hat{p}_{m,k} = \frac{n_k}{n_{-1} + n_{+1}}$$

Entropy:

$$-\hat{p}_{m,+1} \log \hat{p}_{m,+1} - \hat{p}_{m,-1} \log(1 - \hat{p}_{m,+1})$$

Probabilistic Estimates and Entropy

Probabilistic for two classes, $\hat{c}_m$ estimates $P(Y = +1 \mid X \in R_m)$:

$$\hat{c}_m = \arg\min_c -\hat{p}_{m,+1} \log c - \hat{p}_{m,-1} \log(1 - c) = \hat{p}_{m,+1}$$

where

$$n_k = |\{i : X_i \in R_m, Y_i = k\}|, \qquad \hat{p}_{m,k} = \frac{n_k}{n_{-1} + n_{+1}}$$

Entropy:

$$-\hat{p}_{m,+1} \log \hat{p}_{m,+1} - \hat{p}_{m,-1} \log(1 - \hat{p}_{m,+1}) = -\hat{p}_{m,+1} \log \hat{p}_{m,+1} - \hat{p}_{m,-1} \log \hat{p}_{m,-1}$$

Probabilistic Estimates and Entropy

Probabilistic for two classes, $\hat{c}_m$ estimates $P(Y = +1 \mid X \in R_m)$:

$$\hat{c}_m = \arg\min_c -\hat{p}_{m,+1} \log c - \hat{p}_{m,-1} \log(1 - c) = \hat{p}_{m,+1}$$

where

$$n_k = |\{i : X_i \in R_m, Y_i = k\}|, \qquad \hat{p}_{m,k} = \frac{n_k}{n_{-1} + n_{+1}}$$

Entropy:

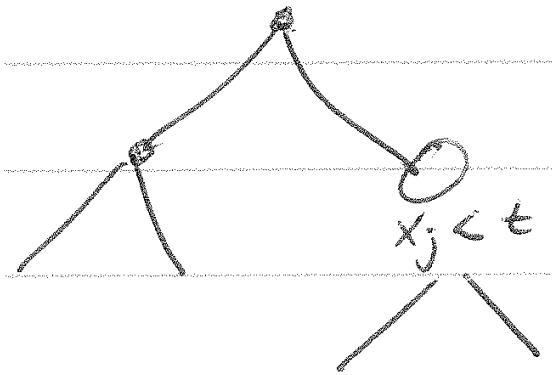
$$-\hat{p}_{m,+1} \log \hat{p}_{m,+1} - \hat{p}_{m,-1} \log(1 - \hat{p}_{m,+1}) = -\hat{p}_{m,+1} \log \hat{p}_{m,+1} - \hat{p}_{m,-1} \log \hat{p}_{m,-1}$$

In general, entropy of a discrete probability distribution:

$$H(P) = -\sum_k P(k) \log_2 P(k)$$

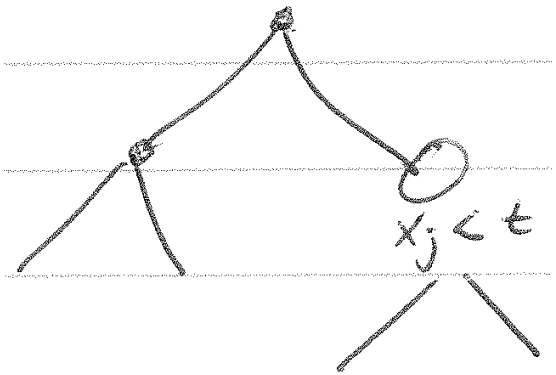
measures **expected number of bits** to encode random sample from P
(base of log is only a scale factor, so does not matter for most purposes)

Growing Trees



How to choose feature X_j + threshold t?

Growing Trees

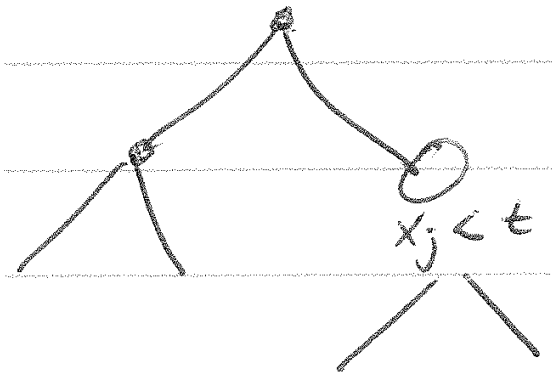


How to choose feature X_j + threshold t?

Greedy Approach: split region R_m into $R_{m,1}$ and $R_{m,2}$ that minimize

$$\text{Loss}(R_{m,1}) + \text{Loss}(R_{m,2}) = \min_c \sum_{i:X_i \in R_{m,1}} L(Y_i, c) + \min_c \sum_{i:X_i \in R_{m,2}} L(Y_i, c)$$

Growing Trees



How to choose feature X_j + threshold t?

Greedy Approach: split region R_m into $R_{m,1}$ and $R_{m,2}$ that minimize

$$\text{Loss}(R_{m,1}) + \text{Loss}(R_{m,2}) = \min_c \sum_{i:X_i \in R_{m,1}} L(Y_i, c) + \min_c \sum_{i:X_i \in R_{m,2}} L(Y_i, c)$$

Regression: always use squared error loss

Classification:

- Use entropy or "Gini index" while growing
- 0/1-loss to estimate class in each region

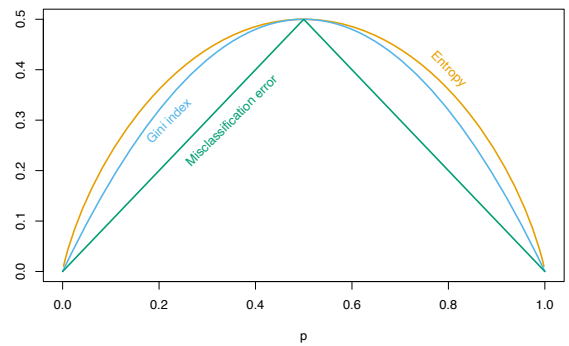


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through $(0.5, 0.5)$.*

Pruning Trees

Larger tree $\rightarrow$ more regions $\rightarrow$ better fit training data

- Too large $\rightarrow$ overfitting, because too little data per region R_m
- Too small $\rightarrow$ underfitting

Pruning Trees

Larger tree $\rightarrow$ more regions $\rightarrow$ better fit training data

- Too large $\rightarrow$ overfitting, because too little data per region R_m
- Too small $\rightarrow$ underfitting

Controlling Tree Complexity:

- Challenge: difficult to know when to stop growing tree
- Solution: grow very big tree $\mathcal{T}$, then prune to smaller subtree $\mathcal{T}_\alpha \subseteq \mathcal{T}$ whose size depends on hyperparameter α

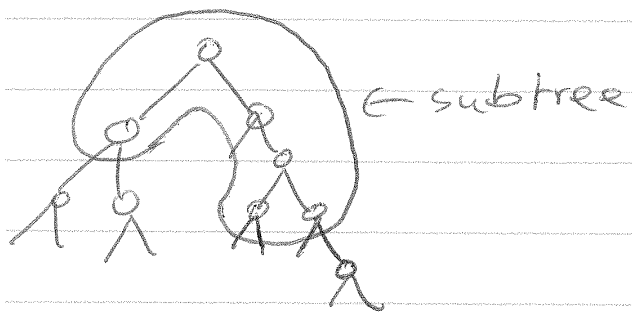
Pruning Trees

Larger tree $\rightarrow$ more regions $\rightarrow$ better fit training data

- Too large $\rightarrow$ overfitting, because too little data per region R_m
- Too small $\rightarrow$ underfitting

Controlling Tree Complexity:

- Challenge: difficult to know when to stop growing tree
- Solution: grow very big tree $\mathcal{T}$, then prune to smaller subtree $\mathcal{T}_\alpha \subseteq \mathcal{T}$ whose size depends on hyperparameter α



$$\mathcal{T}_\alpha \;=\; \underset{\mathcal{S}\subseteq\mathcal{T}}{\arg\min} \sum_{i=1}^{N} L\big(Y_i, \mathcal{S}(X_i)\big) + \alpha|\mathcal{S}| \qquad (\alpha \geq 0),$$

where $|\mathcal{S}|$ denotes the nr. of leaf-nodes in the tree

Outline

Decision Tree Estimation with CART

1. General Setup
2. Growing
3. Pruning

Bias-Variance Trade-off in Regression

Bagging and Random Forests

4. Bagging
5. Random Forests

Bias-Variance Trade-off in Regression

Understand balance between underfitting and overfitting in regression:

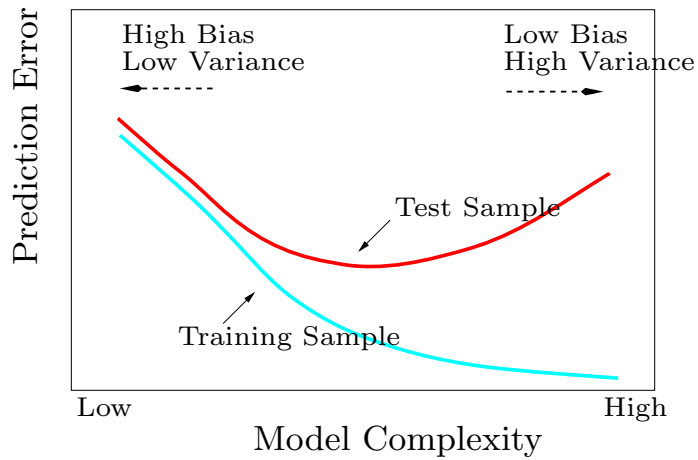
Theorem (Bias-Variance Decomposition)

Let $\bar{f}(X) = \mathbb{E}_T[\hat{f}(X)]$. *Then*

$$\begin{aligned}
\mathop{\mathbb{E}}_{T}[\mathrm{EPE}(\hat{f})] = \mathop{\mathbb{E}}_{X}[\mathrm{Var}(Y \mid X)] & \quad \leftarrow \mathrm{EPE}(f_{\mathrm{B}}) \\
+ \mathop{\mathbb{E}}_{X}\left[\left(f_{\mathrm{B}}(X) - \bar{f}(X)\right)^2\right] & \quad \leftarrow \textit{bias} \\
+ \mathop{\mathbb{E}}_{T,X}\left[\left(\hat{f}(X) - \bar{f}(X)\right)^2\right] & \quad \leftarrow \textit{variance}
\end{aligned}$$

For estimator $\hat{f}$ based on model $\mathcal{F}$:

- Bias typically smaller for larger $\mathcal{F}$
- Variance typically larger for larger $\mathcal{F}$



Outline

Decision Tree Estimation with CART

1. General Setup
2. Growing
3. Pruning

Bias-Variance Trade-off in Regression

Bagging and Random Forests

4. Bagging
5. Random Forests

Bagging

Motivation: Suppose we have a high variance method (e.g. decision trees without pruning). How do we reduce variance?

Bagging

Motivation: Suppose we have a high variance method (e.g. decision trees without pruning). How do we reduce variance?

Bagging = Bootstrap + Aggregation

- High variance = very different estimates (e.g. different trees) when run on new data
- Simulate many data sets by bootstrap and "average out" the variance

Bagging

Bagging = Bootstrap + Aggregation

- Simulate many data sets by bootstrap and "average out" the variance

Bootstrap:

$$T = \begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} Y_N \\ X_N \end{pmatrix}$$

For $b = 1, \ldots, B$:

- Let Z_b be a sample of size N, sampled **with replacement** from T
- Train estimator $\hat{f}_b$ on Z_b

Bagging

Bagging = Bootstrap + Aggregation

- Simulate many data sets by bootstrap and "average out" the variance

Bootstrap:

$$T = \begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} Y_N \\ X_N \end{pmatrix}$$

For $b = 1, \ldots, B$:

- Let Z_b be a sample of size N, sampled **with replacement** from T
- Train estimator $\hat{f}_b$ on Z_b

Aggregation:

- **Regression:** $\hat{f}(X) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(X)$ is average
- **Classification:**
  - Majority vote among $\hat{f}_1, \ldots, \hat{f}_B$
  - Or, if classifiers provide probability estimates $\hat{P}_1(Y \mid X), \ldots, \hat{P}_B(Y \mid X)$, then average those.

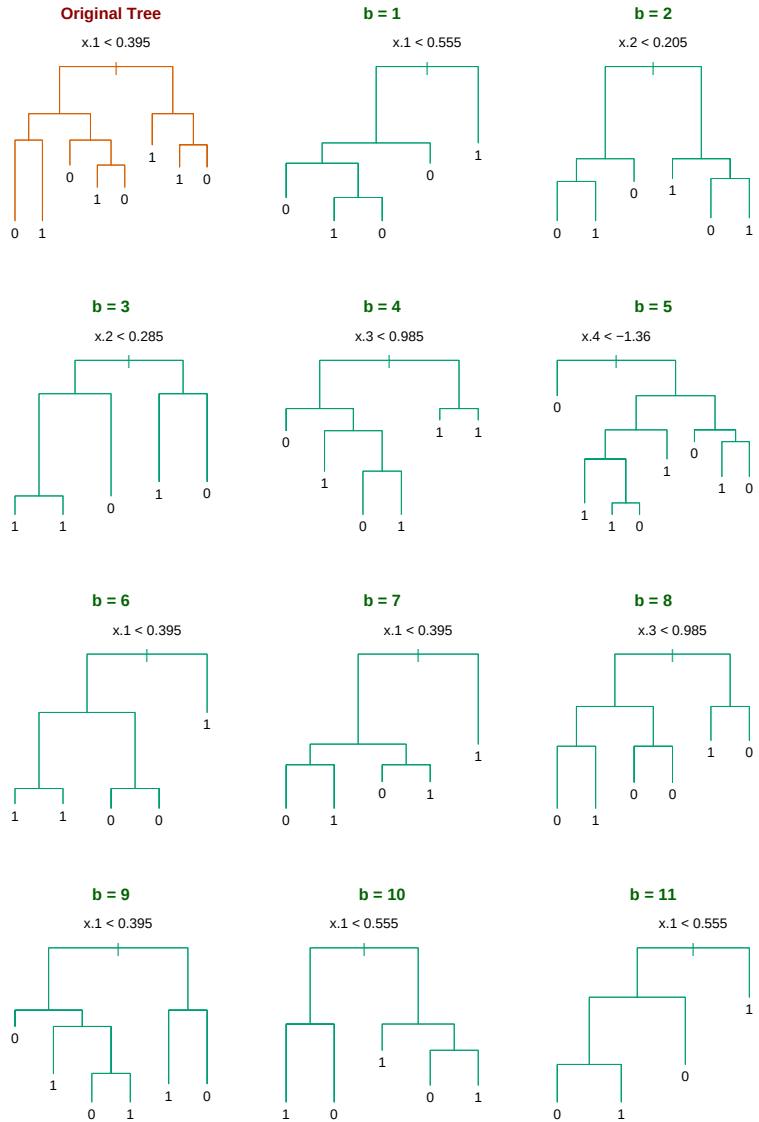
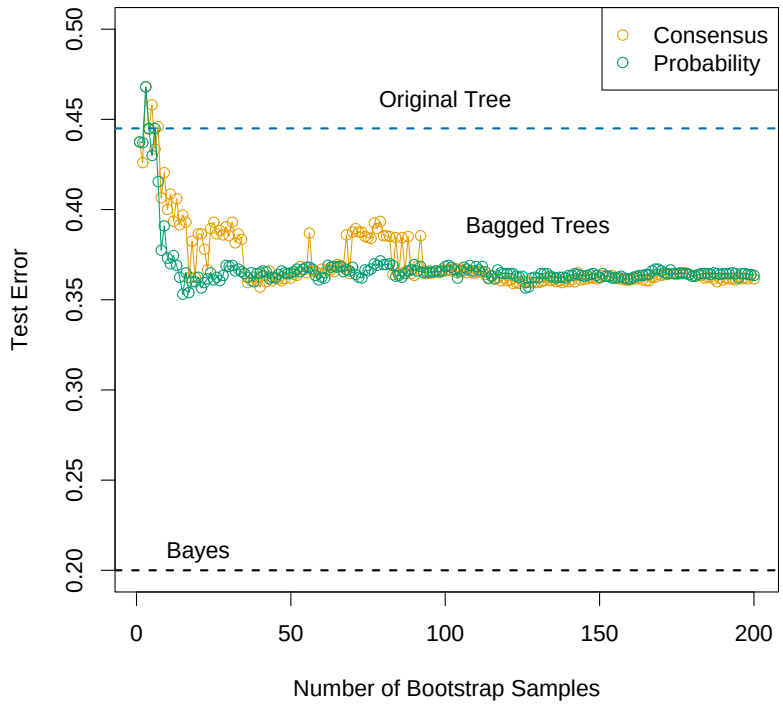


FIGURE 8.9. *Bagging trees on simulated dataset.*



Bagging (potentially) reduces EPE by **reducing variance**

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 orange points correspond to the consensus vote, while the green points average the probabilities.*

Bagging **reduces variance**, but does **not help** (or very little) if model cannot express what we need to learn, i.e. has **large bias**.

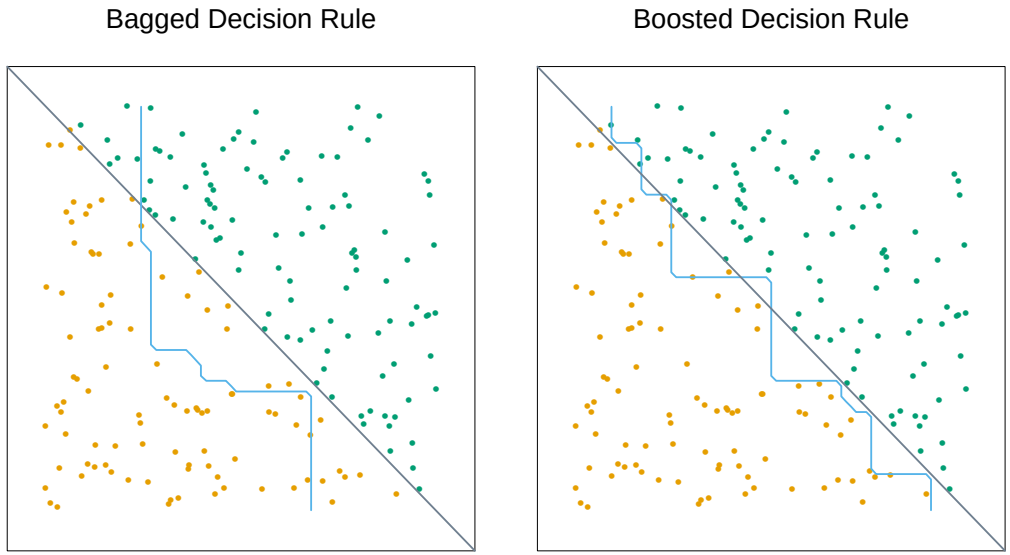


FIGURE 8.12. *Data with two features and two classes, separated by a linear boundary. (Left panel:) Decision boundary estimated from bagging the decision rule from a single split, axis-oriented classifier. (Right panel:) Decision boundary from boosting the decision rule of the same classifier. The test error rates are* 0.166, *and* 0.065, *respectively. Boosting is described in Chapter 10.*

Random Forests

Random Forests:

decision trees (without pruning) + bagging + small extra tricks (e.g. to increase diversity among generated trees)

- **Loses interpretability** of single decision tree
- But much **better predictive performance**