ISE 5103 Intelligent Data Analytics

Decision Tree Methods

Charles Nicholson, Ph.D.
cnicholson@ou.edu

University of Oklahoma
Gallogly College of Engineering
School of Industrial and Systems Engineering

Outline

A decision tree is hierarchical structure which consists of a set of rules for dividing a large heterogeneous population into smaller, more homogeneous groups with respect to a particular target variable

decision trees

- Useful for data with a lot of attributes of unknown importance
- Make no prior assumptions about the data
  - Insensitive to normalization issues
  - Tolerant to correlated and noisy attributes
- Relatively easy to understand
- Powerful/popular for *both* classification and prediction

decision trees

Issues:

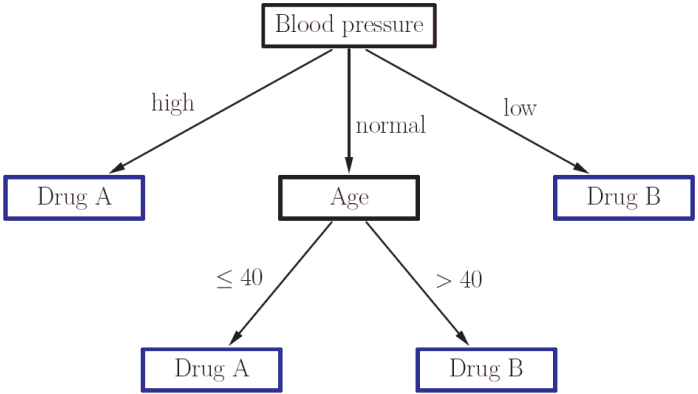
- Decision tree algorithms are unstable
- Trees created from numeric datasets can be complex

decision trees

- Interpretation
- Construction

simple decision tree

Assignment of a drug to a patient:



classification with decision trees

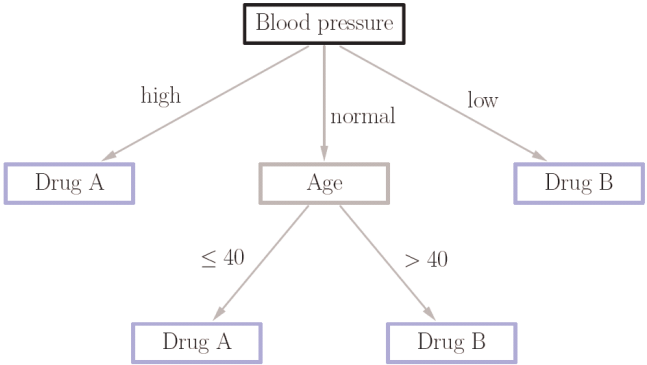
Recursive Descent:

- Start at the root node.
- If the current node is a **leaf node**:
  - Return the class assigned to the node.
- If the current node is an **inner node**:
  - Test the attribute associated with the node.
  - Follow the branch labeled with the outcome of the test.
  - Apply the algorithm recursively.

Intuitively: Follow the path corresponding to the case to be classified.

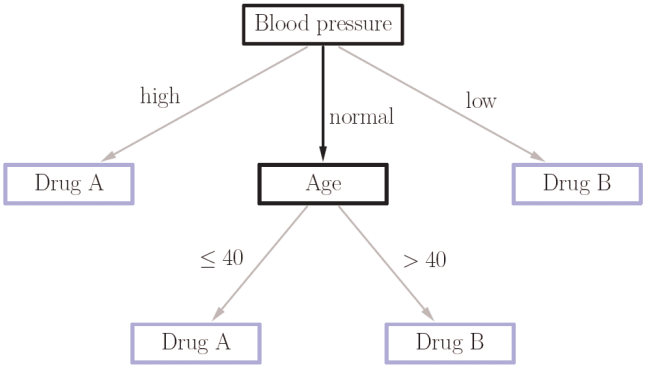
classification with decision trees

Drug assignment to 30 year old patient with normal blood pressure:



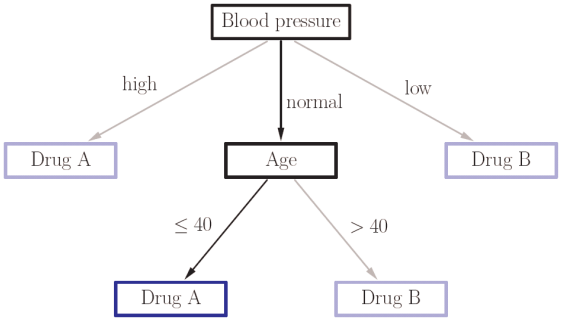
classification with decision trees

Drug assignment to 30 year old patient with normal blood pressure:



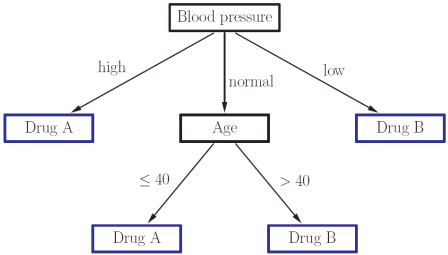
classification with decision trees

Drug assignment to 30 year old patient with normal blood pressure:



classification with decision trees

- Drug A $\Leftrightarrow$ Blood pressure $=$ high
  $\vee$ Blood pressure $=$ normal $\wedge$ Age ≤ 40
- Drug B $\Leftrightarrow$ Blood pressure $=$ low
  $\vee$ Blood pressure $=$ normal $\wedge$ Age > 40



decision trees

The construction of a tree involves the following three elements:

1. selection of the splits
2. decisions when to declare a node terminal or to continue splitting it
3. assignment of each terminal node to a class

induction of decision trees

Plan to construct a tree on sample dataset S.

- top-down approach: build tree T from root to leaves
- selection of the splits
  - compute evaluation measure for predictors
  - select predictor (and value) with best evaluation
- recursive application
  - divide the data according to the splits: e.g. $S \to \{S_1, S_2\}$
  - apply the procedure recursively to the subsets.
    e.g. $S_1 \to \{S_3, S_4\}$ and $S_2 \to \{S_5, S_6, S_7\}$
- decide if node should be split or if it is terminal
- assign class to all terminal nodes

  Let $\tilde{T}$ denote the leaf nodes of tree T

class assignment

A class assignment rule assigns a class $j = \{1, ..., K\}$ to every terminal node $t \in \tilde{T}$.

The class assigned to node $t \in \tilde{T}$ is denoted by $\kappa(t)$.

The class assignment rule is:

$$\kappa(t) = \arg\max_j p(j|t)$$

criterion for predictor selection

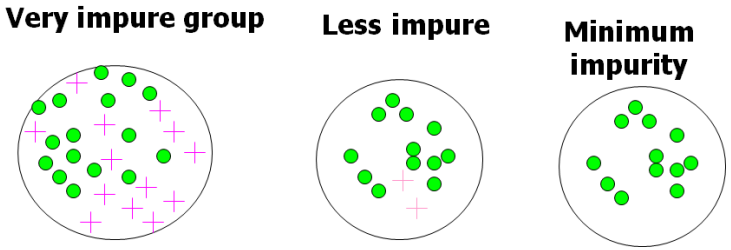
- Which is the best attribute?
  - the one which results in smallest tree (NP hard problem!)
  - heuristic: choose attribute that produces “purest” nodes
- Need a measure of impurity
  - maximal when node is equally split for all classes
  - should be zero if the node is all one class

impurity function

Definition: An *impurity function* is a function ϕ defined on the set of all K-tuples of numbers $(p_1, \ldots, p_K)$ satisfying $p_j \geq 0, j = 1, \ldots, K, \sum_j p_j = 1$ with the properties:

1. ϕ is a maximum only at the point $(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$
2. ϕ achieves its minimum only at the points $(1, 0, ..., 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, ..., 0, 1)$.
3. ϕ is a symmetric function of $(p_1, \ldots, p_K)$, i.e., if you permute p_j, ϕ remains constant.

measures of impurity



There are different measures of impurity, including:

- misclassification rate
- information gain (a.k.a entropy)
- gini index

tree induction example

Split criteria:
misclassification rate

Patient database

- 12 example cases
- 3 descriptive attributes
- 1 class attribute

Assignment of drug
(without patient attributes)
always drug A or always drug B:
50% correct (in 6 of 12 cases)

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

tree induction example

Sex of the patient

- division w.r.t. male/female

Assignment of drug

male:	50% correct	(in 3 of 6 cases)
female:	50% correct	(in 3 of 6 cases)
total:	**50% correct**	(in 6 of 12 cases)

No	Sex	Drug
1	male	A
6	male	A
12	male	A
4	male	B
8	male	B
9	male	B
3	female	A
5	female	A
10	female	A
2	female	B
7	female	B
11	female	B

tree induction example

Blood pressure of the patient

- Division w.r.t. high/normal/low

Assignment of drug

high:	A	100% correct	(in 3 of 3 cases)
normal:		50% correct	(in 3 of 6 cases)
low:	B	100% correct	(in 3 of 3 cases)
total:		**75% correct**	(in 9 of 12 cases)

No	Blood pr.	Drug
3	high	A
5	high	A
12	high	A
1	normal	A
6	normal	A
10	normal	A
2	normal	B
7	normal	B
9	normal	B
4	low	B
8	low	B
11	low	B

tree induction example

Age of the patient

- Sort according to age.
- Find best age split.
  here: ca. 40 years

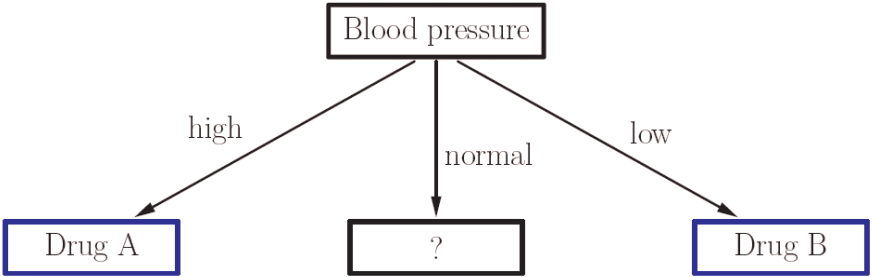
Assignment of drug

≤ 40:	A	67% correct	(in 4 of 6 cases)
> 40:	B	67% correct	(in 4 of 6 cases)
total:		**67% correct**	(in 8 of 12 cases)

No	Age	Drug
1	20	A
11	26	B
6	29	A
10	30	A
4	33	B
3	37	A
8	42	B
5	48	A
7	52	B
12	54	A
9	61	B
2	73	B

tree induction example

Current decision tree:



tree induction example

Blood pressure and sex

- Only patients with normal blood pressure.
- Division w.r.t. male/female.

Assignment of drug

male:	A	67% correct	(2 of 3)
female:	B	67% correct	(2 of 3)
total:		**67% correct**	(4 of 6)

No	Blood pr.	Sex	Drug
3	high		A
5	high		A
12	high		A
1	normal	male	A
6	normal	male	A
9	normal	male	B
2	normal	female	B
7	normal	female	B
10	normal	female	A
4	low		B
8	low		B
11	low		B

tree induction example

Blood pressure and age

- Only patients with normal blood pressure.
- Sort according to age.
- Find best age split: 40 years

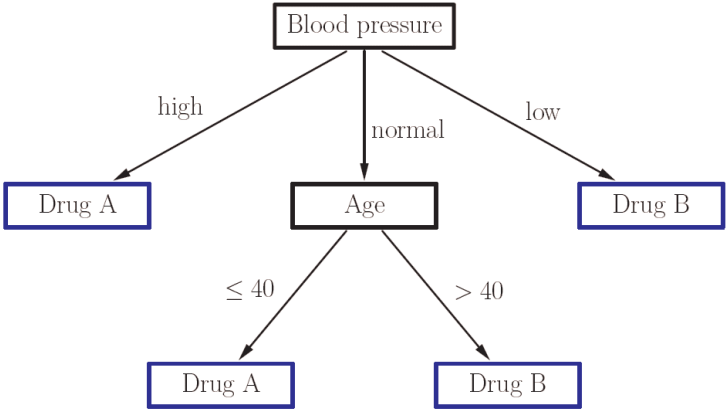
Assignment of drug

≤ 40: A	100% correct	(3 of 3)
> 40: B	100% correct	(3 of 3)
total:	**100% correct**	(6 of 6)

No	Blood pr.	Age	Drug
3	high		A
5	high		A
12	high		A
1	normal	20	A
6	normal	29	A
10	normal	30	A
7	normal	52	B
9	normal	61	B
2	normal	73	B
11	low		B
4	low		B
8	low		B

tree induction example

Resulting decision tree:

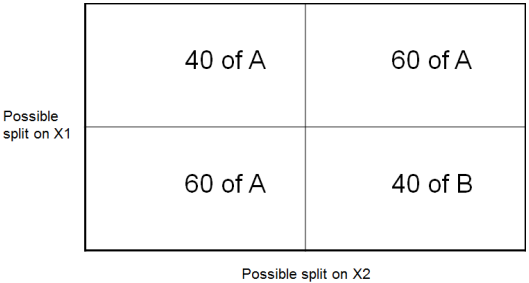


impurity: misclassification

In practice, **misclassification rate** is not used for the following reasons:

- Situations can occur where no split improves the misclassification rate
- Misclassification rate can be equal when one option is clearly better for the next step

problems with misclassifcation rate



Overall accuracy
Assign A: $\frac{160}{200} = 0.8$

Assignment of drug

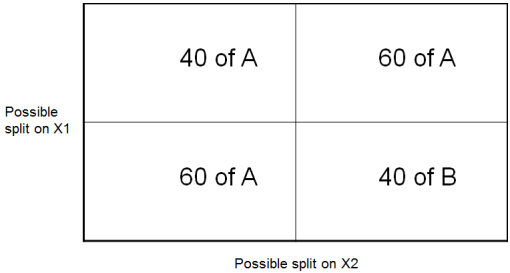
Split on $\mathbf{X_1}$, left node:	A	100% correct	(100 of 100)
Split on $\mathbf{X_1}$, right node:	A	60% correct	(60 of 100)
total:		**80% correct**	(160 of 200)

Assignment of drug

problems with misclassifcation rate

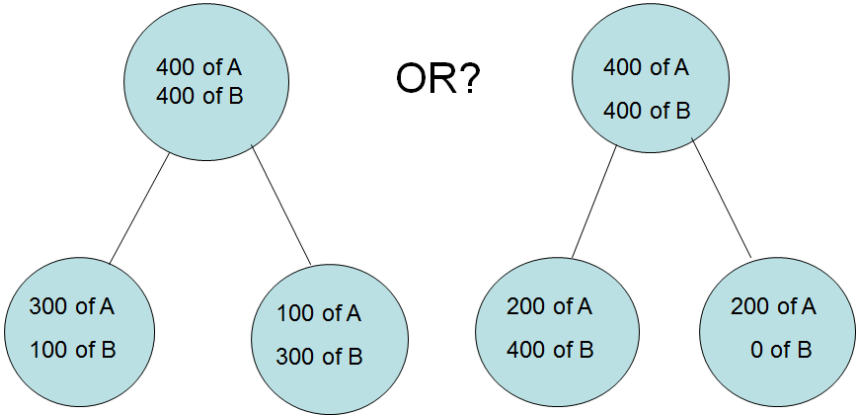
Assignment of drug

Split on X_1, left node:	A	100% correct	(100 of 100)
Split on X_1, right node:	A	60% correct	(60 of 100)
total:		**80% correct**	(160 of 200)
Split on X_2, left node:	A	100% correct	(100 of 100)
Split on X_2, right node:	A	60% correct	(60 of 100)
total:		**80% correct**	(160 of 200)



Neither improves misclassification rate, but together give perfect classification!

problems with misclassifcation rate



Split accuracy in both cases: $\frac{600}{800} = 0.75$

entropy

Entropy: a common way to measure impurity

$$H = -\sum_{i=1}^{K} p_i \log_2 p_i$$

where p_i is the probability of class i

(computed as the proportion of class i in the set)

and $0 \log 0 \equiv 0$

information gain (Kullback/Leibler 1951, Quinlan 1986)

- Goal: to determine which predictor is most useful for discriminating between the target classes
- Information gain uses the entropy measure tells us how important a given predictor is

Information Gain = H(parent) - [expected H(children)]

information gain

$$I_{\text{gain}}(C, A) = \overbrace{H(C)}^{} - \overbrace{H(C|A)}^{}$$

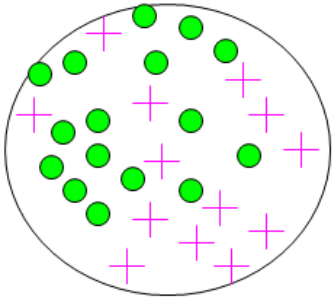
$$= \overbrace{-\sum_{i=1}^{K} p_{i.} \log_2 p_{i.}}^{} - \overbrace{\sum_{j=1}^{n_A} p_{.j} \left(-\sum_{i=1}^{K} p_{i|j} \log_2 p_{i|j} \right)}^{}$$

$H(C)$ Entropy of the class distribution (C: class attribute)

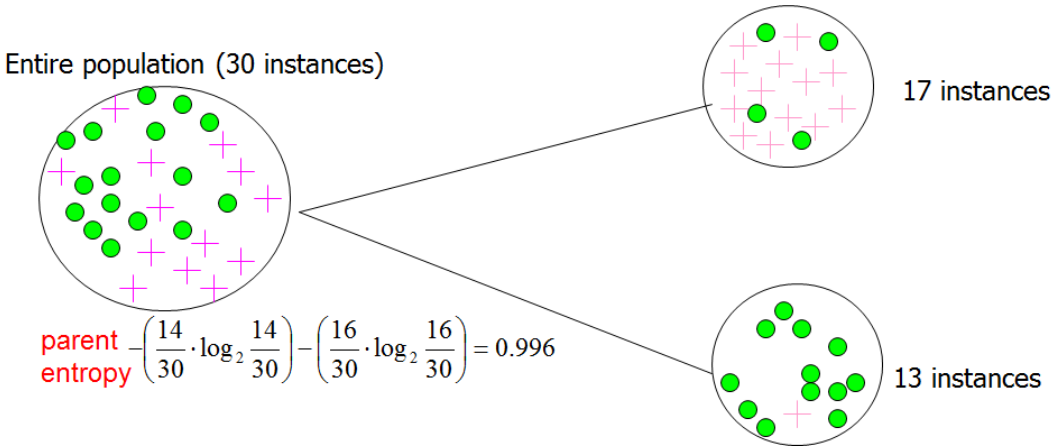
$H(C|A)$ *Expected entropy* of the class distribution if the value of the attribute A becomes known

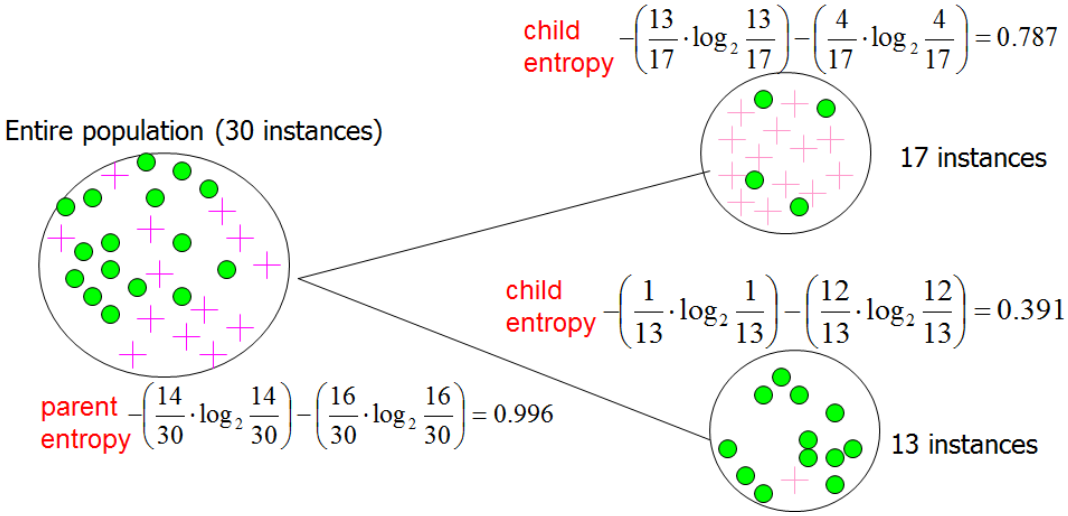
$H(C) - H(C|A)$ Expected entropy reduction or *information gain*

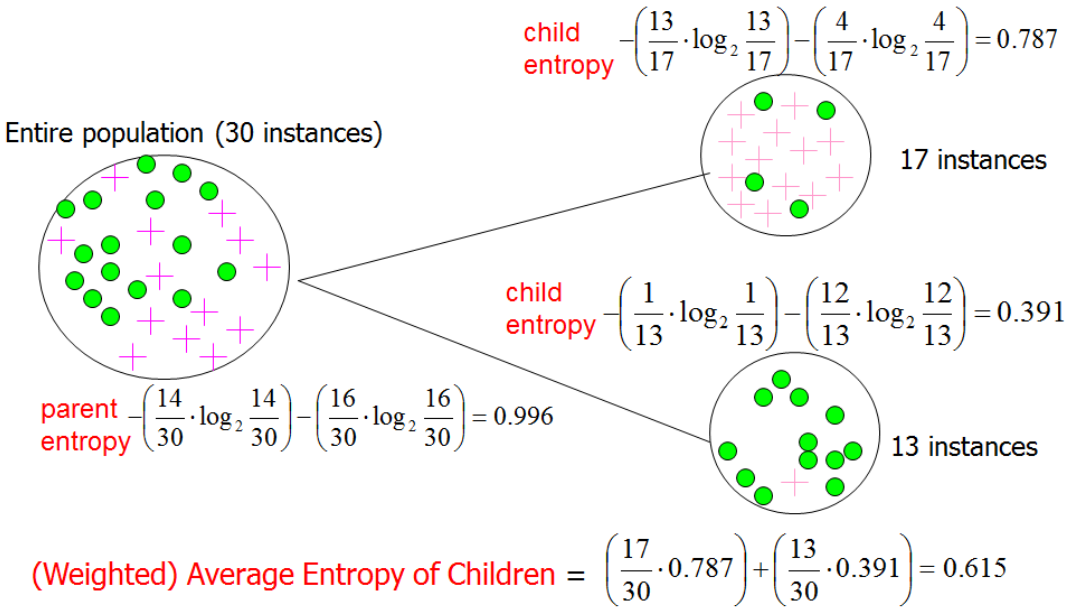
Entire population (30 instances)



parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$


Entire population (30 instances)
17 instances
13 instances
parent entropy
$-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$


child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$
Entire population (30 instances)
17 instances
child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$
parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$
13 instances

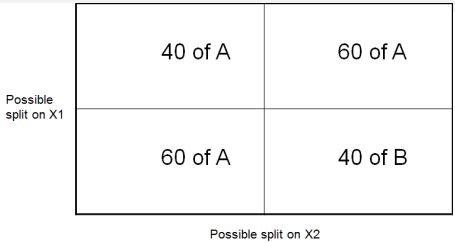


(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

misclassification vs. entropy

Does *entropy* help us with the problems associated with misclassification rate?

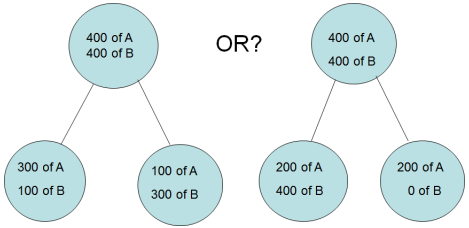


Entropy of parent: $-0.8 \log_2 0.8 - .2 \log_2 0.2 = 0.7219$
Split on X1: Entropy of children

$$-1 \log_2 1 - 0 \log_2 0 = \quad -0.6 \log_2 0.6 - 0.4 \log_2 0.4 =$$
$$0 - 0 = 0 \quad 0.4422 + 0.5288 = 0.9710$$

Weighted average: $\frac{100}{200} \times 0 + \frac{100}{200} \times 0.9710 = 0.4855$

misclassification vs. entropy



Split accuracy in both cases:

$$\frac{600}{800} = 0.75$$

But entropy is different...

Entropy of parent: $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Entropy of children:	weighted average
$-0.75 \log_2 0.75 - 0.25 \log_2 0.25 = 0.8113$ $-0.25 \log_2 0.25 - 0.75 \log_2 0.75 = 0.8113$	$\Rightarrow 0.8113$
$-0.33 \log_2 0.33 - 0.67 \log_2 0.67 = 0.9183$ $-1 \log_2 1 - 0 \log_2 0 = 0$	$\Rightarrow 0.6887$

example: classifying gender

ID	Height	Weight	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

ID	**Height**	Weight	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}) = -\left(\tfrac{4}{10}\log_2\left(\tfrac{4}{10}\right) + \tfrac{6}{10}\log_2\left(\tfrac{6}{10}\right)\right) \approx 0.9710$$

ID	**Height**	Weight	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}|\text{Height}) = -\tfrac{4}{10}\left(\tfrac{3}{4}\log_2\left(\tfrac{3}{4}\right) + \tfrac{1}{4}\log_2\left(\tfrac{1}{4}\right)\right) - \ldots$$

ID	**Height**	Weight	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}|\text{Height}) = \ldots - \tfrac{3}{10}\left(\tfrac{2}{3}\log_2\left(\tfrac{2}{3}\right) + \tfrac{1}{3}\log_2\left(\tfrac{1}{3}\right)\right) - \ldots$$

ID	**Height**	Weight	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}|\text{Height}) = \ldots - \tfrac{3}{10}\left(\tfrac{1}{3}\log_2\left(\tfrac{1}{3}\right) + \tfrac{2}{3}\log_2\left(\tfrac{2}{3}\right)\right)$$

$$
\begin{aligned}
H(\text{Sex}|\text{Height}) &= -\frac{4}{10}\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right)+\frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) \\
&\quad -\frac{3}{10}\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right)+\frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) \\
&\quad -\frac{3}{10}\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right)+\frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) \\
&\approx 0.8755
\end{aligned}
$$

ID	Height	**Weight**	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}|\text{Weight}) = -\tfrac{3}{10} \cdot 0 - \ldots$$

ID	Height	**Weight**	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}|\text{Weight}) = \ldots - \tfrac{5}{10}\left(\tfrac{3}{5}\log_2\left(\tfrac{3}{5}\right) + \tfrac{2}{5}\log_2\left(\tfrac{2}{5}\right)\right) - \ldots$$

ID	Height	**Weight**	Long hair	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$H(\text{Sex}|\text{Weight}) = \ldots - \frac{2}{10} \cdot 0$

$$
\begin{aligned}
H(\text{Sex}|\text{Weight}) &= -\frac{3}{10} \cdot 0 \\
&\quad -\frac{5}{10}\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) \\
&\quad -\frac{2}{10} \cdot 0 \\
&\approx 0.4855
\end{aligned}
$$

ID	Height	Weight	**Long hair**	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

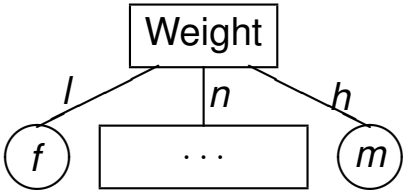
$H(\text{Sex}|\text{Long hair}) = -\frac{4}{10} \cdot 0 - \ldots$

ID	Height	Weight	**Long hair**	Sex
1	m	n	n	m
2	s	l	y	f
3	t	h	n	m
4	s	n	y	f
5	t	n	y	f
6	s	l	n	f
7	s	h	n	m
8	m	n	n	f
9	m	l	y	f
10	t	n	n	m

$$H(\text{Sex}|\text{Long hair}) = \ldots - \frac{6}{10}\left(\frac{2}{6}\log_2\left(\frac{2}{6}\right) + \frac{4}{6}\log_2\left(\frac{4}{6}\right)\right)$$

$$
\begin{aligned}
H(\text{Sex}|\text{Long hair}) &= -\frac{4}{10} \cdot 0 \\
&\quad -\frac{6}{10}\left(\frac{2}{6}\log_2\left(\frac{2}{6}\right) + \frac{4}{6}\log_2\left(\frac{4}{6}\right)\right) \\
&\approx 0.5510
\end{aligned}
$$

The attribute *Weight* yields the largest reduction of entropy.



The remaining data table to be considered in the node $\boxed{\cdots}$:

ID	Height	Long hair	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

ID	Height	Long hair	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

$$H(\text{Sex}|\text{Weight=n}) = -\left(\tfrac{2}{5}\log_2\left(\tfrac{2}{5}\right) + \tfrac{3}{5}\log_2\left(\tfrac{3}{5}\right)\right) \approx 0.9710$$

ID	**Height**	Long hair	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

$$H(\text{Sex}|\text{Weight=n, Height}) = -\frac{1}{5} \cdot 0 - \ldots$$

ID	**Height**	Long hair	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

$$H(\text{Sex}|\text{Weight=n, Height}) = \ldots - \tfrac{2}{5}\left(\tfrac{1}{2}\log_2\left(\tfrac{1}{2}\right) + \tfrac{1}{2}\log_2\left(\tfrac{1}{2}\right)\right) - \ldots$$

ID	**Height**	Long hair	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

$$H(\text{Sex}|\text{Weight=n, Height}) = \ldots - \tfrac{2}{5}\left(\tfrac{1}{2}\log_2\left(\tfrac{1}{2}\right) + \tfrac{1}{2}\log_2\left(\tfrac{1}{2}\right)\right) = 0.8$$

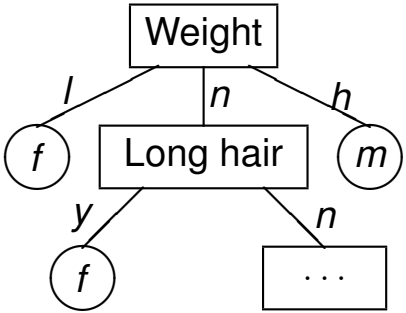
ID	Height	**Long hair**	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

$H(\text{Sex}|\text{Weight=n, Long hair}) = -\frac{2}{5} \cdot 0 - \ldots$

ID	Height	**Long hair**	Sex
1	m	n	m
4	s	y	f
5	t	y	f
8	m	n	f
10	t	n	m

$H(\text{Sex}|\text{Weight=n, Long hair}) = \ldots - \frac{3}{5}\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) \approx 0.5510$

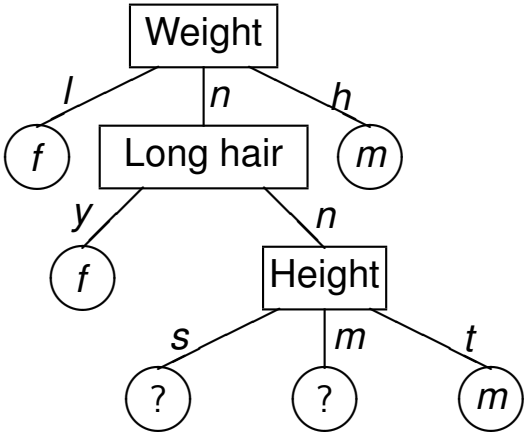
The attribute *long hair* yields the largest reduction of entropy.



For the remaining node, only the attribute *Height* is left with the remaining data table:

ID	Height	Sex
1	m	m
8	m	f
10	t	m

Therefore, the resulting decision tree is:


Weight
l
n
h
f
Long hair
m
y
n
f
Height
s
m
t
?
?
m

- Select the split that most decreases the entropy
  - evaluated over all possible variables and values for a split
- Keep splitting until terminal nodes have very few cases or are all pure
  - this is an unsatisfactory answer to when to stop growing the tree...
  - but, the best approach is to grow a larger tree than required and then to prune it!

information gain ratio

- Information gain is biased towards many-valued attributes i.e., if two attributes have about the same information content it tends to select the one having more values.
- Normalization reduces this bias.

Information Gain Ratio (Quinlan 1986 / 1993)

$$I_{\text{gr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{\textit{Split Info}(A)} = \frac{I_{\text{gain}}(C, A)}{-\sum_{j=1}^{n_A} p_{\cdot j} \log_2 p_{\cdot j}}$$

where n_A is the number of splits on predictor A.

example: information gain vs. gain ratio

From the previous example, using *information gain*,

- The first and most important split was based on **Weight**, split in 3 levels: 30% light, 50% normal, 20% heavy.
  Information gain: $0.971 - 0.4855 = 0.4855$.
- Competing split **Long hair** has 2 levels: 60% no and 40% yes.
  Information gain: $0.971 - 0.551 = 0.420$.

Gain ratio for **Weight**:

$$\frac{0.971 - 0.4855}{-0.3 \log_2 0.3 - 0.5 \log_2 0.5 - 0.2 \log_2 0.2} = \frac{0.4855}{1.485} = 0.327$$

Gain ratio for **Long hair**:

$$\frac{0.971 - 0.551}{-0.6 \log_2 0.6 - 0.4 \log_2 0.4} = \frac{0.420}{0.9709} = 0.433$$

Gini index

The gini index is used by CART (one of several DT algorithms)

$$I_{Gini}(C) = 1 - \sum_{i=1}^{n_C} p_i^2$$

$$\begin{aligned} I_{\text{GiniGain}}(C, A) &= \overbrace{I_{Gini}(C)}^{} - \overbrace{I_{Gini}(C|A)}^{} \\ &= 1 - \sum_{i=1}^{K} p_i^2 - \sum_{j=1}^{n_A} p_{\cdot j} \left(1 - \sum_{i=1}^{K} p_{i|j}^2\right) \end{aligned}$$

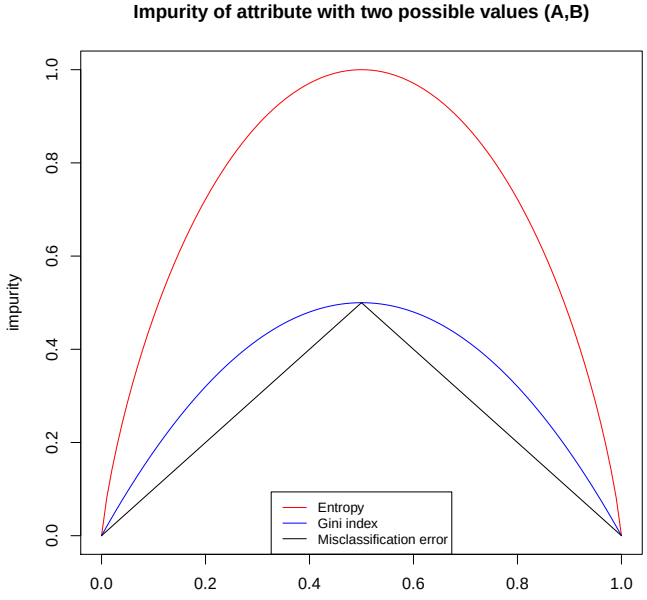
comparison of impurity measures

- **Misclassification rate**: Don't use!

The remaining three measures give good results in general, but each has some particular characteristics:

- **Information Gain**
  - biased towards multivalued attributes
- **Gain Ratio**
  - tends to prefer unbalanced splits (i.e. one partition is much smaller than the other)
- **Gini index**
  - biased towards multivalued attributes
  - has difficulties when the number of classes is large
  - favors equal-sized partitions and purity in both partitions

comparison of impurity measures



Outline

well known tree algorithms

- CART
  - Classification and Regression Trees (Breiman et al., 1984)
  - uses gini index for impurity; produces binary trees
- ID3
  - Iterative Dichotomizer 3, (Quinlan, 1986)
  - uses information gain for impurity
  - cannot handle numeric attributes or missing values
- CHAID
  - Chi-squared automatic interaction detection (Kass, 1980)
  - uses hypothesis testing; likely to produce multi-way splits
- C4.5
  - An improvement on ID3 (Quinlan, 1993)
  - Uses gain ratio. Can handle numeric attributes
- C5.0
  - Improvement on C4.5 by Quinlan. Publicly available in 2011
  - Faster, less memory, supports “boosting”

Useful tree functions in R

Good packages for decision tree modeling in R:

- `rpart` : "recursive partitioning"
  - uses the gini index by default (can be set to information gain)
  - default is very easy to use: `fit <- rpart(class ~ X1 + X2 + X3, data = mydata)`
  - has several control functions
  - works well with the plotting package `party` and `partykit`
  - `Rattle` package has a `fancyRpartPlot` method that produces nice results
- C5
  - uses information gain (or Gain Ratio)
  - one of the lastest and greatest techniques

Some comments about trees

1. monotone transformations of predictors have no effect on tree
2. trees deal with continuous variables by converting them into categorical variables
3. make almost no assumptions about the data
4. especially useful with large data and many predictors
5. can find complex structure in data sets, structure that cannot be detected with regression models
6. due to how trees are formulated (greedy), the overall tree can be far from optimal even though the tree obtained at each local step is optimal
7. trees are unstable: the induced structure is highly dependent on the sample; accuracy may be robust, but the rules are not

Some comments about trees

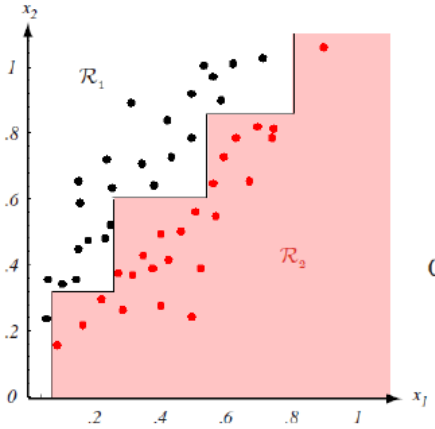
How can trees be used in conjunction with ordinary parametric models?

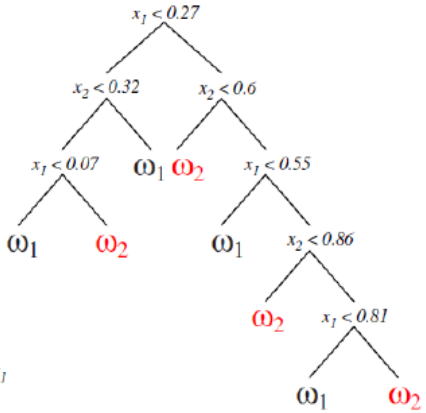
1. to help with feature selection and suggest variable importance for use in parametric models
2. to suggest possible interactions
   - if two (or more) variables are used to split a tree at different points along same branch, it can be seen as an interaction effect
3. to suggest possible non-linear effects
   - if one variable appears repeatedly along a branch, this suggests a possible nonlinear relation between that variable and the response
4. to evaluate residuals
   - having fit a parametric model, one can run the residuals from the model through a tree to see if there is any additional structure that has been overlooked

feature engineering still important

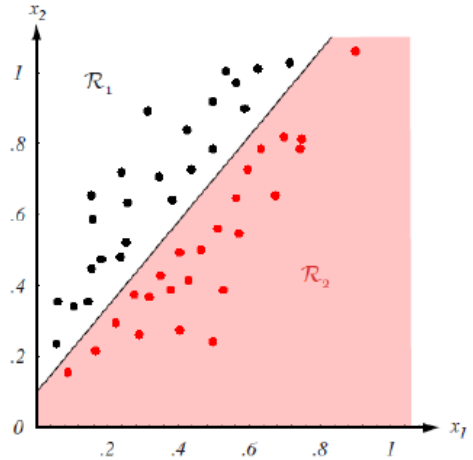
- As with most methods, decision trees work best if proper features are used.
- Specifically preprocessing by PCA can be effective because it finds “important” axes.

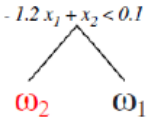
feature engineering still important





feature engineering still important





surrogate splits

- Certain variables are missing in some training or test samples
  - Suppose each variable has 5% chance being missing independently. Then for a training sample with 50 variables, the probability of missing some variables is as high as 92.3%
- A query sample to be classified may have missing variables
- Find *surrogate splits*

  Suppose the best split for node t is s which involves a predictor X_m. Find another split s' on a variable $X_j, j \neq m$, which is most similar to s in a certain sense. Similarly, the second best surrogate split, the third, and so on, can be found

- example data
  - 10 observations
  - 3 predictors
- assume the primary split is: $x_1 < 5.5$ and $x_1 \geq 5.5$
- A surrogate is defined based on the similarity of the results of another rule (splitting on a different variable)

x1	x2	x3
0	7	8
1	8	9
2	9	0
4	1	1
5	2	2
3	3	3
6	0	4
7	4	5
8	5	6
9	6	7

primary split: $x_1 < 5.5$

x1	x2	x3
0	7	8
1	8	9
2	9	0
4	1	1
5	2	2
3	3	3
6	0	4
7	4	5
8	5	6
9	6	7

first surrogate: $x_3 < 3.5$

x1	x2	x3
0	7	8
1	8	9
2	9	0
4	1	1
5	2	2
3	3	3
6	0	4
7	4	5
8	5	6
9	6	7

second surrogate: $x_2 < 3.5$

x1	x2	x3
0	7	8
1	8	9
2	9	0
4	1	1
5	2	2
3	3	3
6	0	4
7	4	5
8	5	6
9	6	7

variable importance

A variable may appear in the tree many times, either as a *primary*, or a *surrogate* variable.

An overall measure of variable importance is the sum of the "goodness" of split measures for each split for which it was the primary variable, plus "goodness" $\times$ (adjusted) agreement for all splits in which it was a surrogate.

Imagine two variables which were essentially duplicates of each other; if we did not count surrogates they would split the importance with neither showing up as strongly as it should.

resubstitution estimate

- The *resubstitution estimate* $r(t)$ of the probability of misclassification, given that a case falls into node t is

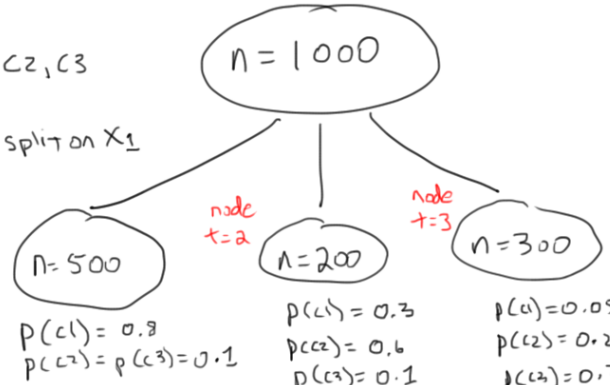
$$r(t) = 1 - \max_j p(j|t) = 1 - p(\kappa(t)|t)$$

- Denote $R(t) = r(t)p(t)$
- The resubstitution estimate for the overall misclassification rate $R(T)$ of the tree classifier T is:

$$R(T) = \sum_{t \in \tilde{T}} R(t)$$

TREE T

3 classes : c_1, c_2, c_3



resubstitution rate $r(t)$	$r(1) = 1 - 0.8 = 0.2$	$r(2) = 1 - 0.6 = 0.4$	$r(3) = 1 - 0.75 = 0.25$
node probability $p(t)$	$p(1) = 0.5$	$p(2) = 0.2$	$p(3) = 0.3$
$R(t)$	$(0.2)(0.5) = 0.1$	$(0.4)(0.2) = 0.08$	$(0.25)(0.3) = 0.075$

$$R(T) = 0.1 + 0.08 + 0.075 = 0.255$$

cost-complexity measure

For any T, define its *complexity* as $|\tilde{T}|$, the number of terminal nodes in T

Let $\alpha \geq 0$ be the complexity parameter and define the cost-complexity measure $R_\alpha(T)$ as: $R_\alpha(T) = R(T) + \alpha|\tilde{T}|$

The complexity parameter (α) is used to control the size of the decision tree.

If the cost of adding another variable to the decision tree from the current node is above the value of α, then tree building does not continue.

The error rate will always drop (or at least not increase) with every split.

This does not mean however that the error rate on Test data will improve

The best method of arriving at a suitable size for the tree is to grow an overly complex one then to prune it back.

Outline

Classification trees are adaptive and robust, but do not generalize well.

- ✓ Classification Trees
1. Bagging: Averaging Trees
2. Random Forests: Cleverer Averaging of Trees
3. Boosting: Cleverest Averaging of Trees

Ensemble methods help improve the performance of weak learners such as decision trees.

The techniques discussed here enhance their performance considerably.

Classification trees can be simple, but often produce noisy (bushy) or weak (stunted) classifiers. Trees are unstable.

- Bagging (Breiman, 1996): Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.
- Random Forests (Breiman, 1999): Fancier version of bagging.
- Boosting (Freund & Shapire, 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.

In general: Boosting $\succeq$ Random Forests $\succeq$ Bagging $\succeq$ Single Tree.

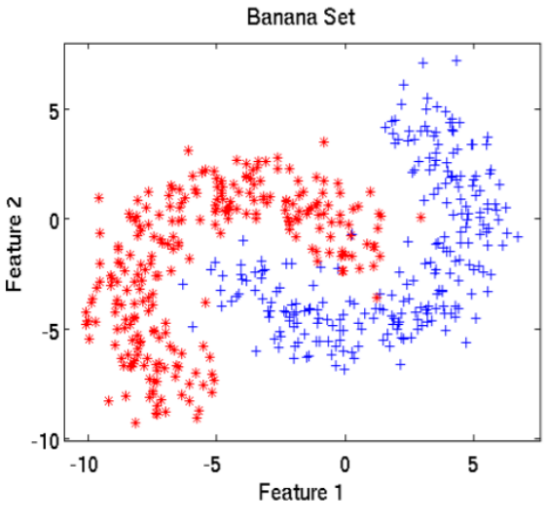
Bootstrap aggregation (a.k.a. "Bagging") is a generic technique applied to bootstrap samples to reduce variance.

- Suppose $C(S, x)$ is a tree classifier based on training data S of size N, producing a predicted class label at input point x
- To bag C, we draw B bootstrap samples with replacement $S_b^1, S_b^2, \ldots, S_b^B$ of size N from the S
- Then,

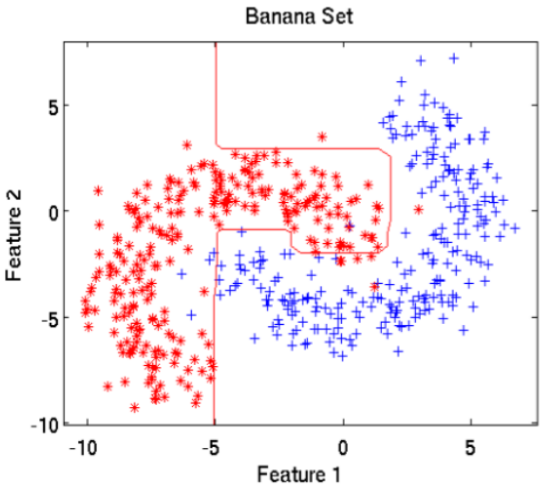
$$C^{bag}(x) = \text{Majority Vote}\{C(S_b^k, x)\}_{k=1}^{B}$$

Bagging can dramatically reduce the variance of unstable procedures (like trees), leading to improved prediction.

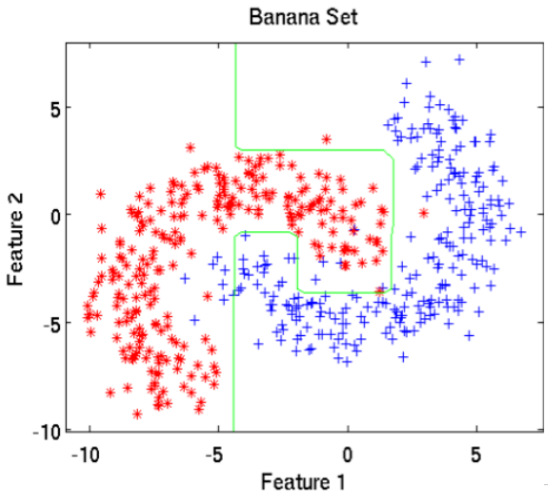
However, the simplicity of the tree classifier is lost.



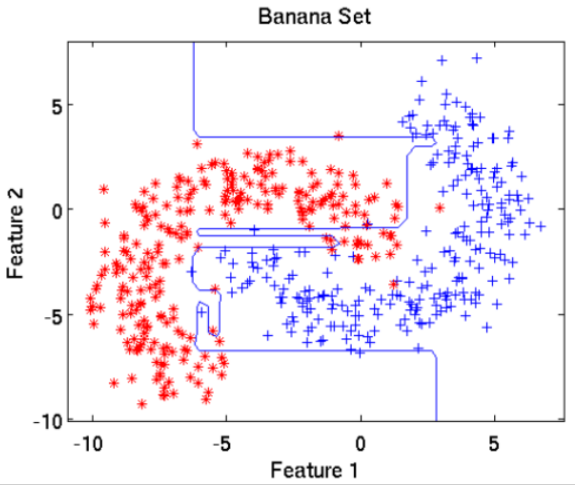
Training data



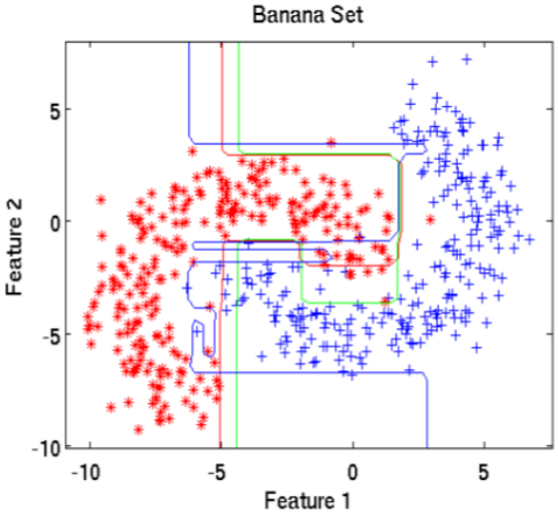
Decision boundary produced by one tree



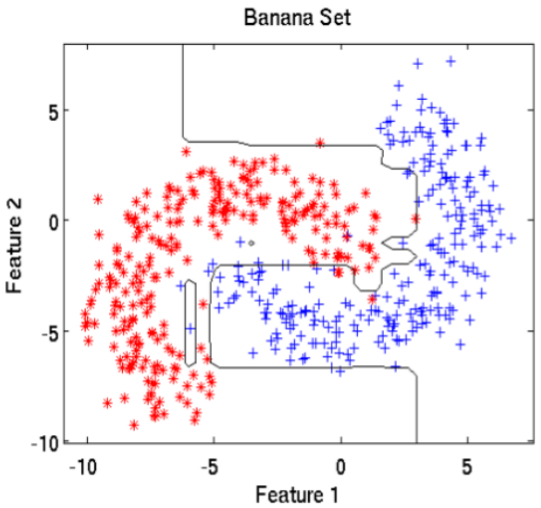
Decision boundary produced by a second tree



Decision boundary produced by a third tree



Three trees and final boundary overlaid



Final result from bagging all trees.

bagged trees

Bagged trees automatically includes a cross-validation feature:

- for each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored
- this is called the “out-of-bag” error rate

random forests

- Popular and powerful refinement of bagged trees
- Injects additional randomness into the bagging procedure on trees
- Each node is split using the best among a *randomly chosen subset of predictors* at that node, instead of the full set

random forests

- At each node, a random sample of m features are selected
- Only these m features are considered for splitting

  typically: $m = \sqrt{p}$ or $m = \log_2 p$ (p is the number of features)
- Random forests tries to improve on bagging by "de-correlating" the trees.

boosting

Boosted trees induces sequence of simple trees where each successive tree is built to be sensitive to the prediction residuals of the preceding tree.

- If the individual trees are very simple (i.e. only uses one split of one predictor), the method is called “stump boosting”
- There are multiple algorithms for boosting (e.g. AdaBoost, MadaBoost, LPBoost, Gradient Boost).
- We will consider one of the most popular one: Adaptive Boosting, a.k.a. AdaBoost (Freund and Schapire, 1997)

AdaBoost: a boosting algorithm

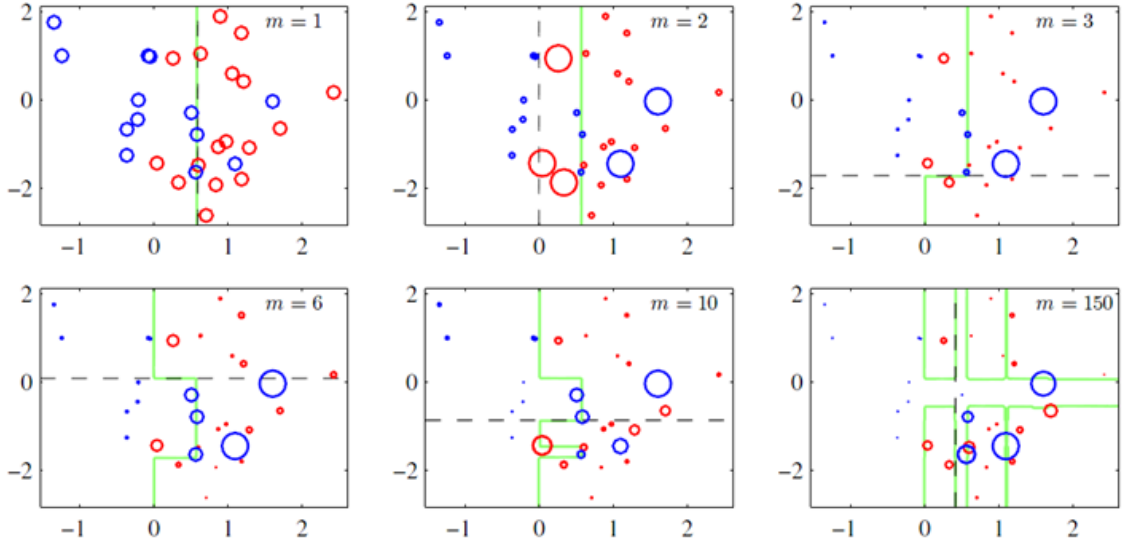
1. For binary classes: set one class to $+1$ and the other to -1
2. Initialize observation weights: $w_i = \frac{1}{N}; i = 1, 2, ..., N$
3. For $m = 1$ to M repeat steps (a)-(d):
   - **(a)** fit a classifier $C_m(x)$ to the training data using weights w_i
   - **(b)** compute weighted error of newest tree

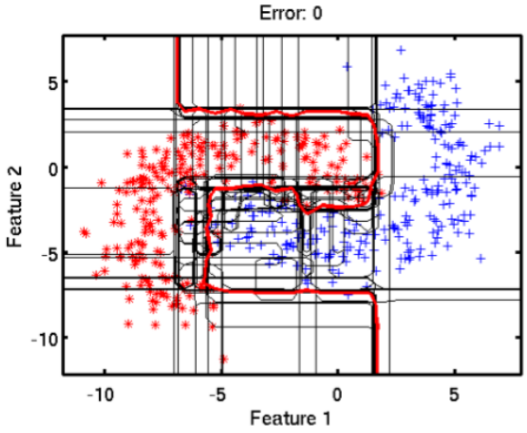
$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq C_m(x_i))}{\sum_{i=1}^{N}}$$

   - **(c)** compute $\alpha_m = \frac{\log(1-\text{err}_m)}{\text{err}_m}$
   - **(d)** update weights for $i = 1, \ldots, N$:

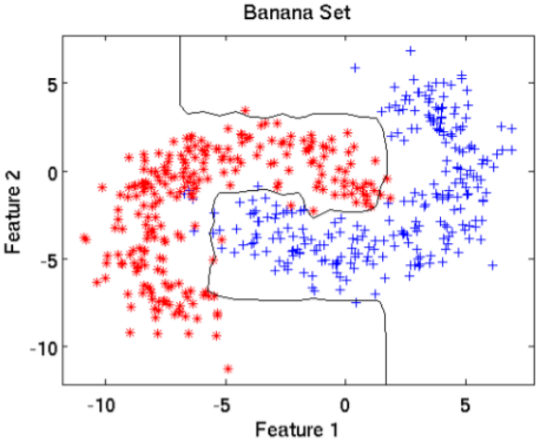
$$w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq C_m(x_i))]$$

   and renormalize w_i to sum to 1
4. Output $C(x) = \text{sign} \sum_{m=1}^{M} \alpha_m C_m(x)$


m = 1
m = 2
m = 3
m = 6
m = 10
m = 150



AdaBoost using 20 decision trees with default settings



Final output of AdaBoost with 20 decision trees