ML (cont.): DECISION TREES

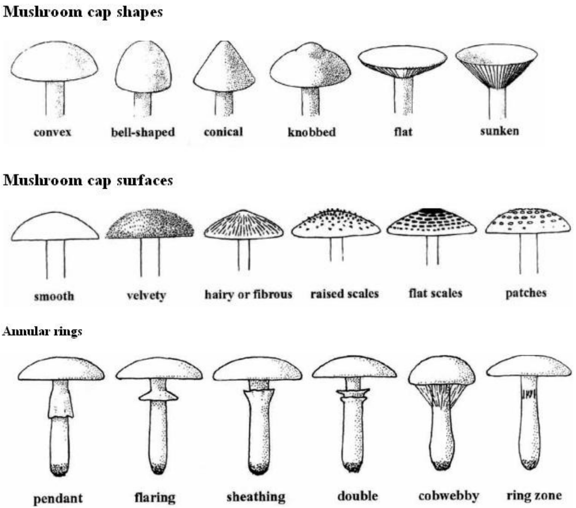
CS540 Bryan R Gibson University of Wisconsin-Madison

Slides adapted from those used by Prof. Jerry Zhu, CS540-1
Some slides from Andrew Moore http://www.cs.cmu.edu/~awm/tutorials and Chuck Dyer

$\mathbf{x}$: Review

- The input (aka example, point, instance, item)
- Usually represented by a feature vector
  - Composed of features (aka attributes)
  - For decision trees (DTs), we focus on discrete features
  - (continuous features are possible, see end of slides)

Example: Mushrooms



Example: Mushroom features

1. **cap-shape**: b=bell, c=conical, x=flat, k=knobbed, s=sunken
2. **cap-surface**: f=fibrous, g=grooves, y=scaly , s=smooth
3. **cap-color**: n=brown, b=buff, c=cinnamon, g=gray, r=green, p=pink, u=purple, e=red, w=white, y=yellow
4. **bruises?**: t=bruises, f=no
5. **odor**: a=almond, l=anise, c=creosote, y=fishy, f=foul, m=musty, n=none, p=pungent, s=spicy
6. **gill-attachment**: a=attached, d=descending, f=free, n=notched
7. ...

for example : $\mathbf{x}_1 =$ [b,g,r,t,f,n,...]

y: Review

- The output (aka label, target, goal)
- It can be . . .
  - Continuous $\rightarrow$ Regression (e.g. population prediction)
  - Discrete $\rightarrow$ Classification (e.g. is mushroom $\mathbf{x}$ edible or poisonous?)

Example: Two Mushrooms

- $\mathbf{x}_1 =$ [x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u]
  $y =$ p
- $\mathbf{x}_2 =$ [x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g]
  $y =$ e

1. **cap-shape**: b=bell, c=conical, x=flat, k=knobbed, s=sunken
2. **cap-surface**: f=fibrous, g=grooves, y=scaly , s=smooth
3. **cap-color**: n=brown, b=buff, c=cinnamon, g=gray, r=green, p=pink, u=purple, e=red, w=white, y=yellow
4. **bruises?**: t=bruises, f=no
5. ...

Supervised Learning: Review

- Training set: n pairs of example,label: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
- A function (aka hypotheses) $f : \mathbf{x} \mapsto y$
- Hypothesis space (subset of function family): e.g. the set of dth order polynomials
- Goal: find the best function in the hypothesis space that generalizes well
- Performance measure:
  - MSE for regression,
  - accuracy or error rate for classification

Evaluating Classifiers: Review

- During training
  - Train classifier from a training set: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$.
- During testing
  - For new test data $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+m}$, classifier generates predicted labels: $\hat{y}_{n+1}, \ldots, \hat{y}_{n+m}$.
- Test set accuracy
  - Need to know the true test labels: $y_{n+1}, \ldots, y_{n+m}$.
  - Test set accuracy: $\text{acc} = \frac{1}{m} \sum_{i=n+1}^{n+m} \mathbb{1}\{y_i = \hat{y}\}$
  - Test set error rate: $1 - \text{acc}$

Decision Trees

- Another kind of classifier (SL)
  - The tree
  - Algorithm
  - Mutual Information of questions
  - Overfitting and Pruning
  - Extension: real-valued features, tree $\mapsto$ rules, pro/con

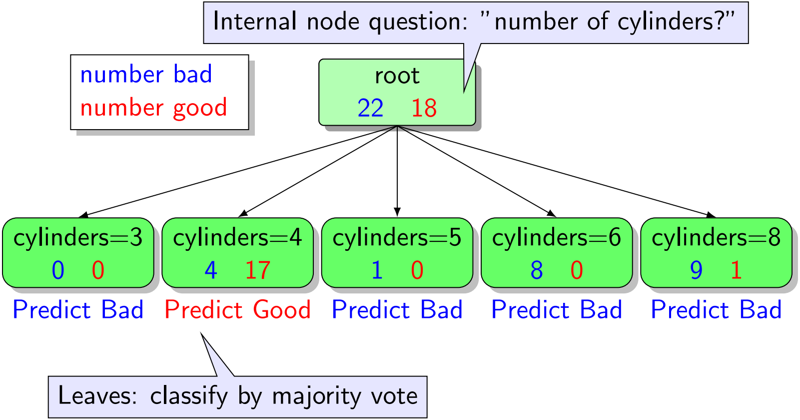
Decision Trees (cont.)

- A decision tree has 2 kinds of nodes:
  - leaf node: has a class label determined by majority vote of training examples reaching that leaf
  - internal node: a question on features. Branches out according to the answers

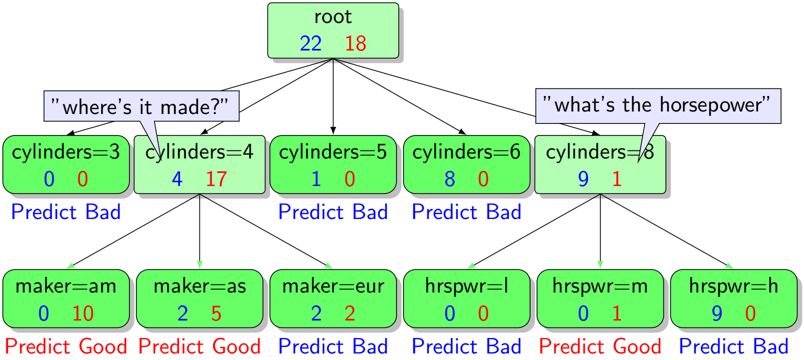
Automobile Miles-Per-Gallon Prediction

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

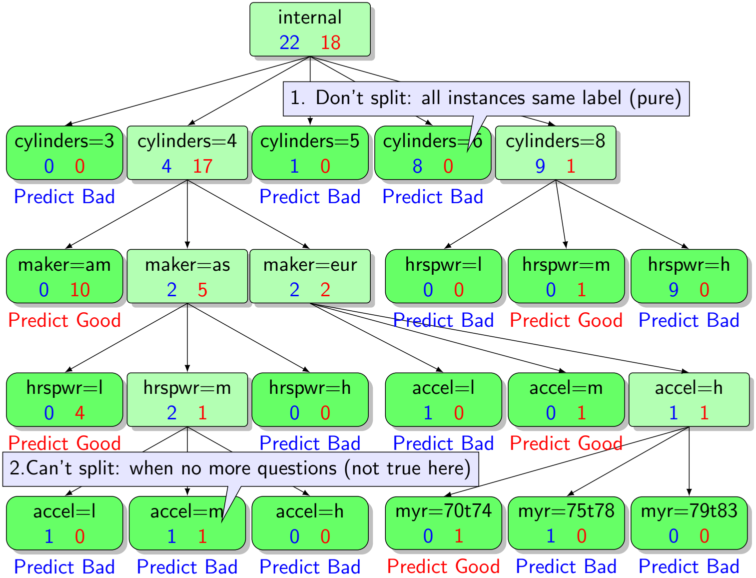
A Small Decision Tree



A Bigger DT


root
22 18
"where's it made?"
"what's the horsepower"
cylinders=3
0 0
Predict Bad
cylinders=4
4 17
cylinders=5
1 0
Predict Bad
cylinders=6
8 0
Predict Bad
cylinders=8
9 1
maker=am
0 10
Predict Good
maker=as
2 5
Predict Good
maker=eur
2 2
Predict Bad
hrspwr=l
0 0
Predict Bad
hrspwr=m
0 1
Predict Good
hrspwr=h
9 0
Predict Bad

The Full DT


internal
22 18
1. Don't split: all instances same label (pure)
cylinders=3
0 0
Predict Bad
cylinders=4
4 17
cylinders=5
1 0
Predict Bad
cylinders=6
8 0
Predict Bad
cylinders=8
9 1
maker=am
0 10
Predict Good
maker=as
2 5
maker=eur
2 2
hrspwr=l
0 0
Predict Bad
hrspwr=m
0 1
Predict Good
hrspwr=h
9 0
Predict Bad
hrspwr=l
0 4
Predict Good
hrspwr=m
2 1
hrspwr=h
0 0
Predict Bad
accel=l
1 0
Predict Bad
accel=m
0 1
Predict Good
accel=h
1 1
2.Can't split: when no more questions (not true here)
accel=l
1 0
Predict Bad
accel=m
1 1
Predict Bad
accel=h
0 0
Predict Bad
myr=70t74
0 1
Predict Good
myr=75t78
1 0
Predict Bad
myr=79t83
0 0
Predict Bad

The Decision Tree Algorithm

root.buildTree(*examples*,*questions*,*default*)

```
/* examples: a list of training examples
   questions: set of candidate questions (e.g.‘‘value of feature i?’’)
   default: default label prediction (e.g. over-all majority vote) */

IF empty(examples) THEN this.setPrediction(default)
IF (examples all same label y) THEN this.setPrediction(y)
IF empty(questions) THEN this.setPrediction(examples maj. vote)

q = bestQuestion(examples, questions)

For j = 1..n answers to q
  node = new Node;
  this.addChild(node)
  node.buildTree({examples|q=answer j},{questions\q},default)
```

The Best Question

- What we want: pure leaf nodes
- pure means all examples have (almost) the same label y
- A good question $\rightarrow$ splits examples into pure child nodes
- How to measure how much "purity" results from a question?
- One possiblity (called Max-Gain in the book):

  Mutual Information or Information Gain
  (a quantity from information theory)

- But this is a measure of change of purity
- We still need to find a measure of purity/impurity first ...

The Best Question: Entropy

- Imagine, at a node there are $n = n_1 + \ldots + n_k$ examples:
  - n_1 examples have label y_1
  - n_2 examples have label y_2
  - $\ldots$
  - n_k examples have label y_k
- What's the impurity of the node?
- Imagine this game:
  - I put all of the examples in a bag ...
  - then pull one out at random.
  - What is the probability the example has label y_i?
- p_i!
- but how do we calculate p_i?

The Best Question: Entropy (cont.)

- We'll estimate p_i from our examples:
  - with probability $p_1 = \frac{n_1}{n}$, the example has label y_1
  - with probability $p_2 = \frac{n_2}{n}$, the example has label y_2
  - ...
  - with probability $p_k = \frac{n_k}{n}$, the example has label y_k
- so that $p_1 + p_2 + \ldots + p_k = 1$
- The "outcome" of the draw is a random variable y with probability $(p_1, p_2, \ldots, p_k)$
- "What's the impurity of the node" is the same as asking: "What's the uncertainty of y in a random drawing"

The Best Question: Entropy Defined

$$
\begin{aligned}
H(Y) &= \sum_{i=1}^{k} -Pr(Y = y_i) \log_2 Pr(Y = y_i) \\
&= \sum_{i=1}^{k} -p_i \log_2 p_i
\end{aligned}
$$

Interpretation:

> *Entropy (H) is the number of yes/no questions (bits) needed* **on average** *to pin down the value of y in a random drawing*

The Best Question: Entropy, some examples

The Best Question: Conditional Entropy

$$H(Y \mid X = v) = \sum_{i=1}^{k} -Pr(Y = y_i \mid X = v) \log_2 Pr(Y = y_i \mid X = v)$$

$$H(Y \mid X) = \sum_{v \in \{\text{values of} X\}} Pr(X = v) H(Y \mid X = v)$$

- Y : label
- X : a question (e.g. a feature
- v : an answer to the question
- $Pr(Y \mid X = v)$: conditional probability

The Best Question: Information Gain

- Information Gain, or Mutual Information

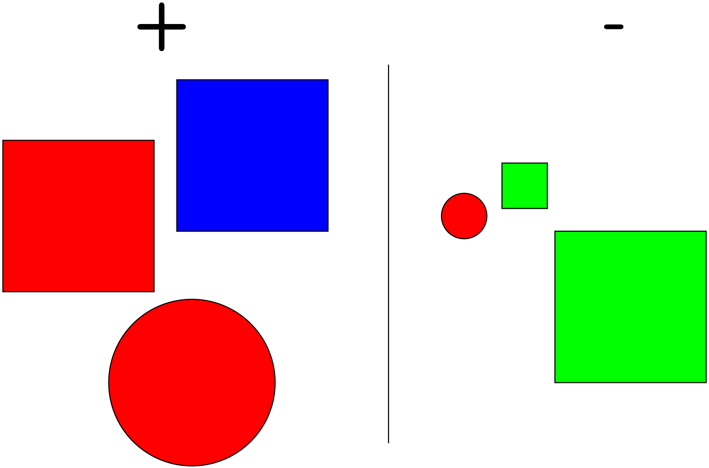
$$I(Y; X) = H(Y) - H(Y \mid X)$$

- Choose question (feature) X which maximizes $I(Y; X)$

$$\arg\max_X I(Y; X)$$

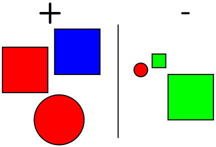
The Best Question: Example

- Features: color, shape, size
- What's the best question at root?



The Best Question: Example, Information Gain

Example	Color	Shape	Size	Class
1	Red	Square	Big	$+$
2	Blue	Square	Big	$+$
3	Red	Circle	Big	$+$
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

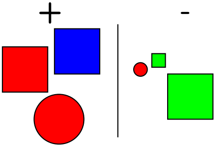


- $H(\text{class}) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1$
- $H(\text{class} \mid \textbf{color}) = \frac{3}{6}\left[H(\frac{2}{3}, \frac{1}{3})\right] + \frac{1}{6}\left[H(1,0)\right] + \frac{2}{6}\left[H(0,1)\right] = .46$

  3 out of 6 are red, 2 of those are $+$;
  1 out of 6 is blue, that one is $+$;
  2 out of 6 are green, those are -
- $I(\text{class}; \textbf{color}) = H(\text{class}) - H(\text{class} \mid \textbf{color}) = 0.54$ bits

The Best Question: Example, Information Gain (cont.)

Example	Color	Shape	Size	Class
1	Red	Square	Big	$+$
2	Blue	Square	Big	$+$
3	Red	Circle	Big	$+$
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

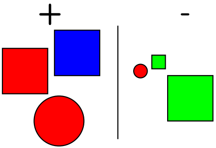


- $H(\text{class}) = 1$
- $I(\text{class}; \text{color}) = 0.54$ bits
- $H(\text{class} \mid \textbf{shape}) = \frac{4}{6}\left[H(\frac{1}{2}, \frac{1}{2})\right] + \frac{2}{6}\left[H(\frac{1}{2}, \frac{1}{2})\right] = 1$
- $I(\text{class}; \textbf{shape}) = H(\text{class}) - H(\text{class} \mid \textbf{shape}) = 0$ bits

  Shape tells us nothing about class!

The Best Question: Example, Information Gain (cont.)

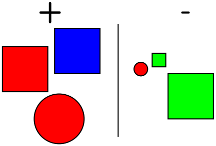
Example	Color	Shape	Size	Class
1	Red	Square	Big	$+$
2	Blue	Square	Big	$+$
3	Red	Circle	Big	$+$
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



- $H(\text{class}) = 1$
- $I(\text{class}; \text{color}) = 0.54$ bits
- $I(\text{class}; \text{shape}) = 0$ bits
- $H(\text{class} \mid \textbf{size}) = \frac{4}{6}\left[H(\frac{3}{4}, \frac{1}{4})\right] + \frac{2}{6}\left[H(0,1)\right] = 0.54$
- $I(\text{class}; \textbf{size}) = H(\text{class}) - H(\text{class} \mid \text{size}) = 0.46$ bits

The Best Question: Example, Information Gain (cont.)

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



- $H(\text{class}) = 1$
- $I(\text{class}; \text{color}) = 0.54$ bits
- $I(\text{class}; \text{shape}) = 0$ bits
- $I(\text{class}; \text{size}) = 0.46$ bits
- We select **color** as the question at root!

Overfitting Example: Predicting US Population

- Given some training data ($n = 11$)
- What will the population be in 2020?

x=Year	y=Millions
1900	75.995
1910	91.972
1920	105.71
1930	123.2
1940	131.67
1950	150.7
1960	179.32
1970	203.21
1980	226.51
1990	249.63
2000	281.42

Overfitting Example: Regression - Polynomial Fit

- The degree d (complexity of the model) is important:

$$f(x) = c_d x^d + c_{d-1} x^{d-1} + \ldots + c_1 x + c_0$$

- Want to fit (or learn) coefficients $c_d, \ldots, c_0$ to minimize Mean Squared Error (MSE) on training data:

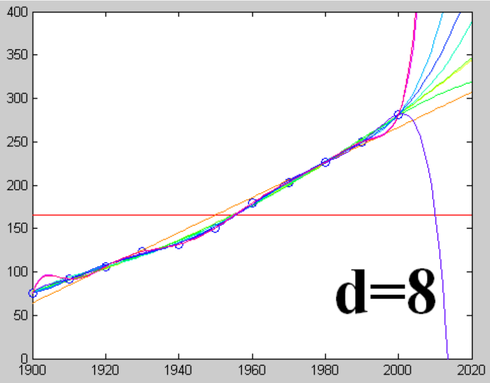
$$\mathsf{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- Matlab demo: USpopulation.m

Overfitting Example: Regression - Polynomial Fit (cont.)

As d increases, MSE on training set improves, but prediction outside data worsens

degree	MSE
0	4181.4526
1	79.6005
2	9.3469
3	9.2896
4	7.4201
5	5.3101
6	2.4932
7	2.2783
8	1.2580
9	0.0014
10	0.0000



Overfitting a Decision Tree

- construct a special training set
- feature vector of five bits
- create every possible configuration (32 configurations)
- set y = e, then randomly flip 25% of the y labels

$\underbrace{\text{32 records}}$

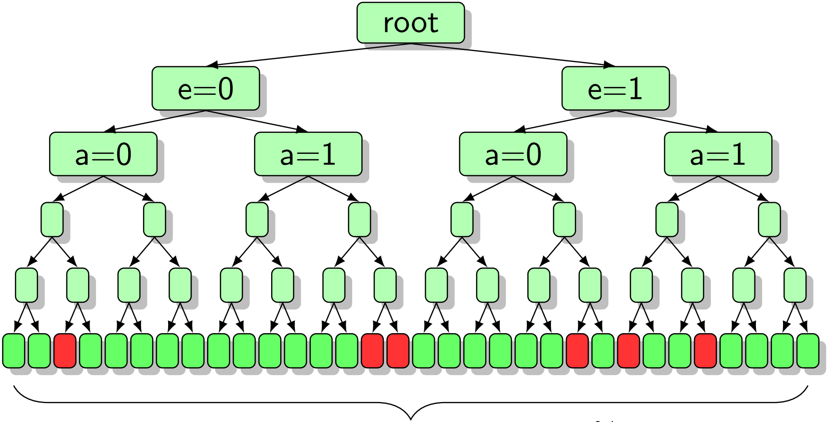
a	**b**	**c**	**d**	**e**	**y**
0	0	0	0	0	0
0	0	0	0	1	**0**
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	**1**
⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	1	1

Overfitting a Decision Tree (cont.)

- Test set is constructed similarly
  - $y = e$, then a different 25% corrupted to $y = \neg e$
  - corruptions in training and test are independent
- Training and Test sets have many identical feature, label pairs
- Some labels different between Training and Test

Overfitting a Decision Tree (cont.)

- Building the full tree on the Training Set:

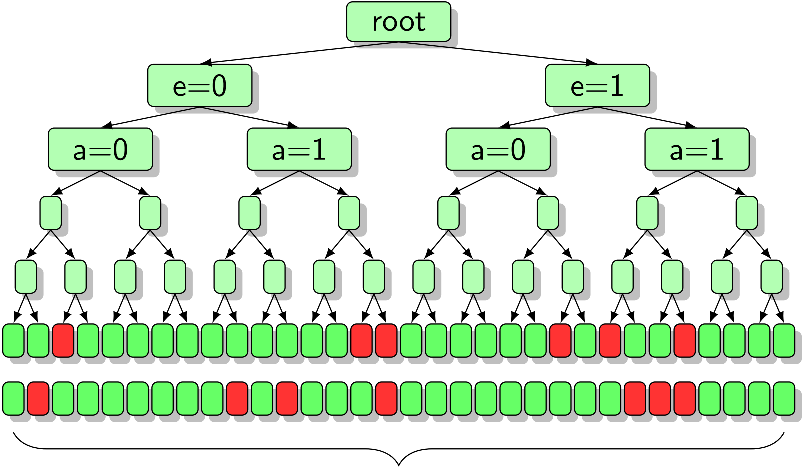


Training set accuracy = 100%

Remember: 25% of nodes are corrupted ($y \neq e$)

Overfitting a Decision Tree (cont.)

- Then classify the Test Set with the learned tree:



Different 25% corrupted - independent of the training data.

Overfitting a Decision Tree (cont.)

On average:

- $\frac{3}{4}$ of training data uncorrupted
  - $\frac{3}{4}$ of these are uncorrupted in test $\rightarrow$ correct predicted labels
  - $\frac{1}{4}$ of these are corrupted in test $\rightarrow$ incorrect predictions
- $\frac{1}{4}$ of training data corrupted
  - $\frac{3}{4}$ of these are uncorrupted in test $\rightarrow$ incorrect predictions
  - $\frac{1}{4}$ of these are also corrupted in test $\rightarrow$ correct predictions
- Test Set Accuracy $= \left(\frac{3}{4}\right)\left(\frac{3}{4}\right) + \left(\frac{1}{4}\right)\left(\frac{1}{4}\right) = \frac{5}{8} = 62.5\%$

Overfitting a Decision Tree

- but if we knew a,b,c,d were irrelevant features and didn't use them in the tree . . .

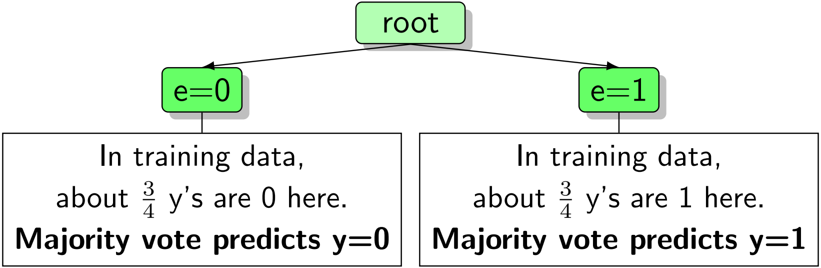
Pretend these don't exist (a, b, c, d)

32 records

a	**b**	**c**	**d**	**e**	**y**
0	0	0	0	0	0
0	0	0	0	1	**0**
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	**1**
⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	1	1

Overfitting a Decision Tree (cont.)

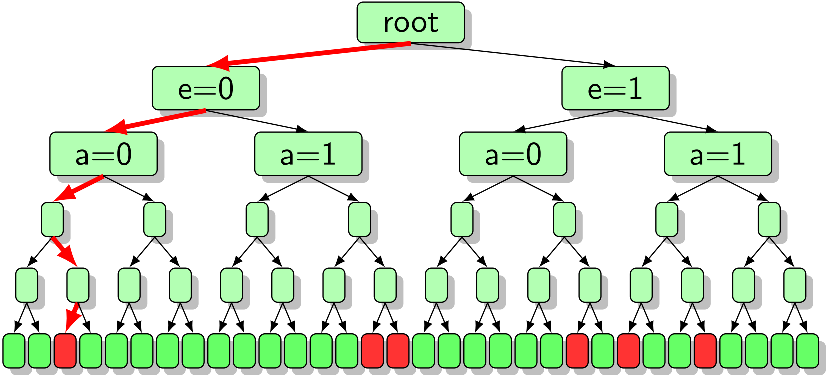
- The tree would be . . .



- In test data, $\frac{1}{4}$ of the y's are different from e.
- Test accuracy = ?
- Test accuracy = $\frac{3}{4}$ = 75%
- (better than full tree test accuracy of 62.5%!)

Overfitting a Decision Tree (cont.)

- In the full tree, we overfit by learning non-existent relations (noise)



Avoiding Overfitting: Pruning

How to prune with a tuning set:

1. Randomly split data into TRAIN and TUNE sets
   (say 70% and 30% for instance)
2. Build a full tree using **only TRAIN**
3. Prune the tree using the **TUNE set**
   (next page shows a greedy version)

Greedy Pruning Algorithm

Prune(tree T, TUNE set)

1. compute T's accuracy on TUNE, call it A(T)
2. For every internal node N in T:
   2.1 Create new tree T_N by: copy T, but prune (delete) subtree under N
   2.2 N becomes a leaf node in T_N: label is majority vote of TRAIN examples reaching N
   2.3 Calculate A(T_N) $= T_N$'s accuracy on TUNE
3. Find T^*, the tree (among T_N's and T) with largest A()
4. Set $T \leftarrow T^*$ (pruning!)
5. Repeat from step 1 until no more improvement available (A() does not increase)
6. Return T (the fully pruned tree)

Real-valued Features

- What if some (or all) of the features $x_1, x_2, \ldots, x_k$ are real-valued?
- Example: $x_1 =$ height (in inches)
- Idea 1: branch on each possible numerical value (fragments the training data, prone to overfitting, with caveat)
- Idea 2: use questions like $(x_1 > t?)$, where t is a threshold. There are fast ways to try all(?) t.

$$H(y \mid x_i > t?) = p(x_i > t) H(y \mid x_i > t) + p(x_i \leq t) H(y \mid x_i \leq t)$$
$$I(y; x_i > t?) = H(y) - H(y \mid x_i > t?)$$

What does the feature space look like?

Axis-parallel cuts

Trees as Rules

- Each path, from root to a leaf, corresponds to a rule
  - antecedent is all of decistions leading to leaf
  - consequent is classification at the leaf node
- For example:
  from the tree in color/shape/size example, we can generate:

  IF ([color]=red) AND ([size]=big) THEN +

Summary

- Decision trees are popular tools for data mining
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Overfiting might happen $\rightarrow$ pruning!
- We've used decision trees for classification
  (predict categorical output from categorical or real inputs)

What you should know

- Trees for classification
- Top-down tree construction algorithm
- Information Gain (includes Entropy and Conditional Entropy)
- Overfitting
- Pruning
- Dealing with real-valued features