## Decision Trees

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting


## A Decision Tree



## When to Consider Decision Trees

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data


## Examples

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences


## Decision Trees

## Decision tree representation

- Each internal node tests an attribute
- Each branch corresponds to an attribute value
- Each leaf node assigns a classification

How would you represent:

- $\wedge, \vee$, XOR
- $(\mathrm{A} \wedge \mathrm{B}) \vee(\mathrm{C} \wedge \neg \mathrm{D} \wedge \mathrm{E})$
- M of N

CS 8751 ML \& KDD

## Top-Down Induction of Decision Trees

Main loop:

1. A = the "best" decision attribute for next node
2. Assign A as decision attribute for node
3. For each value of A, create descendant of node
4. Divide training examples among child nodes
5. If training examples perfectly classified, STOP Else iterate over new leaf nodes

Which attribute is best?




## Information Gain

$\operatorname{Gain}(S, A)=$ expected reduction in entropy due to sorting on $A$
$\operatorname{Gain}(S, A) \equiv \operatorname{Entropy}(S)-\sum_{v \in \operatorname{Values}(A)} \frac{\left|S_{v}\right|}{|S|} \operatorname{Entropy}\left(S_{v}\right)$

$$
\text { Entropy }([29+, 35-])=-\frac{29}{64} \log _{2}\left(\frac{29}{64}\right)-\frac{35}{64} \log _{2} \frac{35}{64}=0.994
$$

Entropy $([21+, 5-])=-\frac{21}{26} \log _{2}\left(\frac{21}{26}\right)-\frac{5}{26} \log _{2} \frac{5}{26}=0.706$ Entropy $([8+, 30-])=0.742$
$\operatorname{Gain}(S, A 1)=0.994-\left(\frac{26}{64} \operatorname{Entropy}([21+, 5-])+\right.$
$\left.\frac{38}{64} \operatorname{Entropy}([8+, 30-])\right)=0.266$
Entropy $([18+, 33-])=0.937$
Entropy $([11+, 2-])=0.619$
$\operatorname{Gain}(S, A 2)=0.121$
CS 8751 ML \& KDD
Decision Trees

## Entropy

Entropy $(S)=$ expected number of bits need to encode class (+ or -) of randomly drawn member of $S$ (using an optimal, shortest-length code)
Why?
Information theory: optimal length code assigns $-\log _{2} p$ bits to message having probability $p$
So, expected number of bits to encode + or - of random member of $S$ :
$-p_{+} \log _{2} p_{+}-p_{-} \log _{2} p_{-}$

Entropy $(S) \equiv-p_{+} \log _{2} p_{+}-p_{-} \log _{2} p_{-}$
CS 8751 ML \& KDD Decision Trees

|  |  |  | 1 Exa | mples |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Color | Type | Doors | Tires |  | Ss |
|  | Red | SUV | 2 | Whitewall | + | 0 |
|  | Blue | Minivan | 4 | Whitewall | - | $\xrightarrow{\square}$ |
|  | Green | Car | 4 | Whitewall | - | $\bigcirc$ |
|  | Red | Minivan | 4 | Blackwall | - | 0 |
|  | Green | Car | 2 | Blackwall | $+$ |  |
|  | Green | SUV | 4 | Blackwall | - | $1 \square_{0}$ |
|  | Blue | SUV | 2 | Blackwall | - |  |
| 11 | Blue | Car | 2 | Whitewall | $+$ |  |
| 1 | Red | SUV | 2 | Blackwall | - |  |
|  | Blue | Car | 4 | Blackwall | - |  |
|  | Green | SUV | 4 | Whitewall | + | $\square_{0}$ |
|  | Red | Car | 2 | Blackwall | + |  |
|  | Green | SUV | 2 | Blackwall | - | - |
|  | Green | Minivan | 4 | Whitewall | - | $\bigcirc$ |
|  | CS 8751 ML \& KDD |  | Decision Trees |  | 10 |  |



## Selecting Next Attribute



Resulting Tree


Hypothesis Space Search by ID3


## Hypothesis Space Search by ID3

- Hypothesis space is complete!
- Target function is in there (but will we find it?)
- Outputs a single hypothesis (which one?)
- Cannot play 20 questions
- No back tracing
- Local minima possible
- Statistically-based search choices
- Robust to noisy data
- Inductive bias: approximately "prefer shortest tree"


## Inductive Bias in ID3

Note $H$ is the power set of instances $X$
Unbiased?
Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a preference for some hypotheses, rather than a restriction of hypothesis space $H$
- Occam's razor: prefer the shortest hypothesis that fits the data


## Occam's Razor

Why prefer short hypotheses?
Argument in favor:

- Fewer short hypotheses than long hypotheses
- short hyp. that fits data unlikely to be coincidence
- long hyp. that fits data more likely to be coincidence

Argument opposed:

- There are many ways to define small sets of hypotheses
- e.g., all trees with a prime number of nodes that use attributes beginning with " $Z$ "
- What is so special about small sets based on size of hypothesis?


## Overfitting in Decision Trees

Consider adding a noisy training example:
<Green,SUV,2,Blackwall> +
What happens to decision tree below?


CS 8751 ML \& KDD
Decision Trees

## Overfitting

Consider error of hypothesis $h$ over

- training data : error $_{\text {train }}(h)$
- entire distribution $D$ of data : error $_{D}(h)$

Hypothesis $h \in H$ overfits the training data if there is an alternative hypothesis $h^{\prime} \in H$ such that

$$
\text { error }_{\text {train }}(h)<\text { error }_{\text {train }}\left(h^{\prime}\right)
$$

and
$\operatorname{error}_{D}(h)>\operatorname{error}_{D}\left(h^{\prime}\right)$

## Overfitting in Decision Tree Learning



CS 8751 ML \& KDD

## Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, the post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation set (examples from the training set that are put aside)
- MDL: minimize
size(tree) + size(misclassifications(tree)


## Reduced-Error Pruning

Split data into training and validation set
Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy

- Produces smallest version of most accurate subtree
- What if data is limited?

Effect of Reduced-Error Pruning


## Decision Tree Post-Pruning

- A standard method in C4.5, C5.0
- Construct a complete tree
- For each node estimate what the error might be with and without the node (needs a conservative estimate of error since based on training data)
- Prune any node where the expected error stays the same or drops
- Greatly influenced by method for estimating likely errors


## Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

## Converting a Tree to Rules

IF (Type=Car) AND (Doors=2) THEN + IF (Type=SUV) AND (Tires=Whitewall) THEN + IF (Type=Minivan) THEN -


CS 8751 ML \& KDD
Decision Trees 27

## Continuous Valued Attributes

Create one (or more) corresponding discrete attributes based on continuous
$-($ EngineSize $=325)=$ true or false
$-($ EngineSize $<=330)=\mathrm{t}$ or f ( 330 is "split" point)
How to pick best "split" point?

1. Sort continuous data
2. Look at points where class differs between two values
3. Pick the split point with the best gain


CS 8751 ML \& KDD Decision Trees
28

## Attributes with Many Values

Problem:

- If attribute has many values, Gain will select it
- Imagine if cars had PurchaseDate feature - likely all would be different
One approach: use GainRatio instead
$\operatorname{GainRatio}(S, A) \equiv \frac{\operatorname{Gain}(S, A)}{\operatorname{SplitInformation}(S, A)}$
SplitInformation $(S, A) \equiv-\sum_{i=1}^{c} \frac{\left|S_{i}\right|}{|S|} \log _{2} \frac{\left|S_{i}\right|}{|S|}$
where $S_{i}$ is subset of $S$ for which $A$ has value $v_{i}$
CS 8751 ML \& KDD


## Attributes with Costs

## Consider

- medical diagnosis, BloodTest has cost $\$ 150$
- robotics, Width_from_1ft has cost 23 second

How to learn consistent tree with low expected cost?
Approaches: replace gain by
Tan and Schlimmer (1990) $\frac{\operatorname{Gain}^{2}(S, A)}{\operatorname{Cost}(A)}$
Nunez (1988)
$\frac{2^{\operatorname{Gain}(S, A)}-1}{(\operatorname{Cost}(A)+1)^{w}}$
where $w \in[0,1]$ and determines importance of cost

## Unknown Attribute Values

What if some examples missing values of $A$ ? "?" in C4.5 data sets
Use training example anyway, sort through tree

- If node $n$ tests $A$, assign most common value of $A$ among other examples sorted to node $n$
- assign most common value of $A$ among other examples with same target value
- assign probability $p_{i}$ to each possible value $v_{i}$ of $A$ - assign fraction $p_{i}$ of example to each descendant in tree

Classify new examples in same fashion

