# Decision Trees

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Eric Eaton (UPenn), who assembled the original slides, David Kauchak (Pomona), whose slides are also heavily used, and the many others who made their course materials freely available online.



# Decision Tree Basics

Learning Goals

- Define Decision Tree
- State expressiveness of decision trees (what data can it classify)

# Sample Dataset

- Columns denote features $X_i$
- Rows denote labeled instances $(x^{(i)}, y^{(i)})$
- Class label denotes whether tennis game was played

| | Predictors | | | | Response |
|---|---|---|---|---|---|
| | **Outlook** | **Temperature** | **Humidity** | **Wind** | **Class** |
| | Sunny | Hot | High | Weak | No |
| | Sunny | Hot | High | Strong | No |
| | Overcast | Hot | High | Weak | Yes |
| | Rain | Mild | High | Weak | Yes |
| | Rain | Cool | Normal | Weak | Yes |
| $(x^{(i)}, y^{(i)})$ | Rain | Cool | Normal | Strong | No |
| | Overcast | Cool | Normal | Strong | Yes |
| | Sunny | Mild | High | Weak | No |
| | Sunny | Cool | Normal | Weak | Yes |
| | Rain | Mild | Normal | Weak | Yes |
| | Sunny | Mild | Normal | Strong | Yes |
| | Overcast | Mild | High | Strong | Yes |
| | Overcast | Hot | Normal | Weak | Yes |
| | Rain | Mild | High | Strong | No |

sunny, cool, normal, weak?

overcast, cool, high, weak?

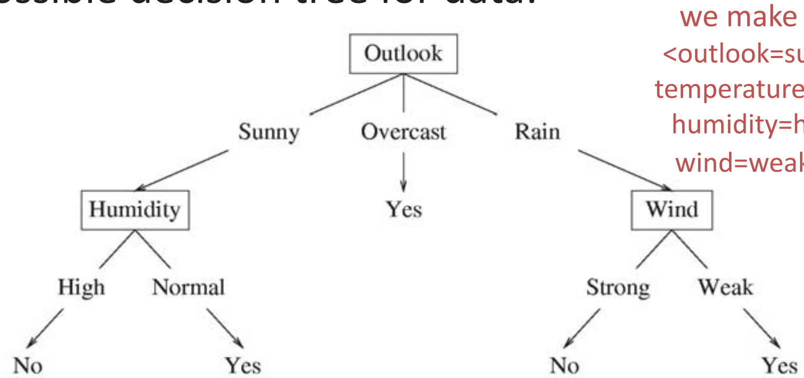
Can you describe a "model" that could be used to make decisions in general?

Based on slides by Eric Eaton and David Kauchak

# Decision Tree

- A possible decision tree for data:

What prediction would we make for <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

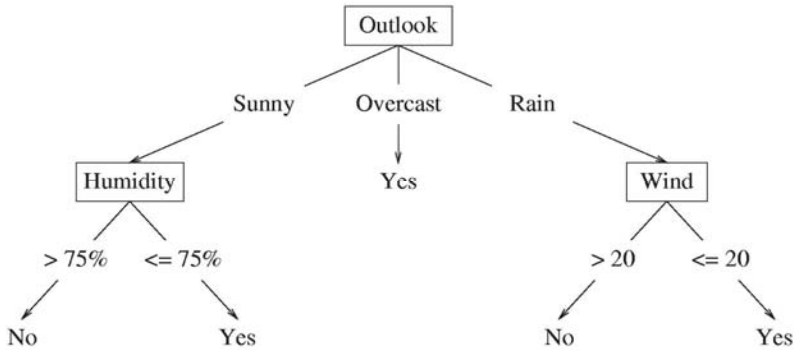



- Each internal node: test one feature $X_i$
- Each branch from node: selects one value for $X_i$
- Each leaf node: predict $Y$ [or $p(Y \mid X \in \text{leaf})$]

Based on slide by Eric Eaton
(originally by Tom Mitchell)

# Decision Tree

- If features are continuous, internal nodes can test value of feature against threshold

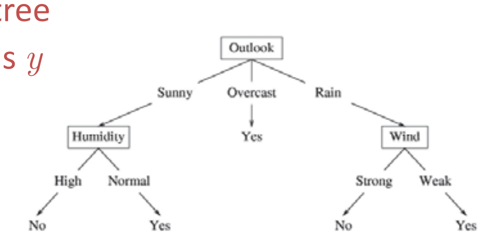



Slide credit: Eric Eaton

# Decision Tree Learning

**Problem Setting**

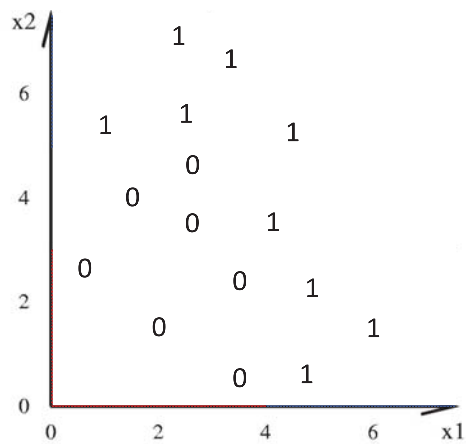
- Set of possible instances $\mathcal{X}$
  - each instance $x$ in $\mathcal{X}$ is feature vector
  - e.g. <humidity=low, wind=weak, outlook=rain, temp=hot>
- Set of possible labels $\mathcal{Y}$
  - $\mathcal{Y}$ is discrete-valued
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$
  - each hypothesis $h$ is a decision tree
  - trees sort $x$ to leaf, which assigns $y$

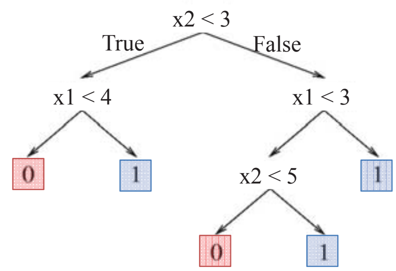



Based on slide by Tom Mitchell

# Decision Tree – Decision Boundary

- Decision trees divide feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or probability distribution over labels



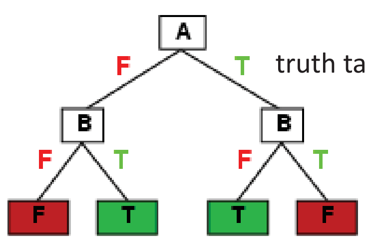




Slide credit: Eric Eaton

# Expressiveness

DTs can represent any boolean function of input features

| A | B | A xor B |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |




truth table row → path to leaf

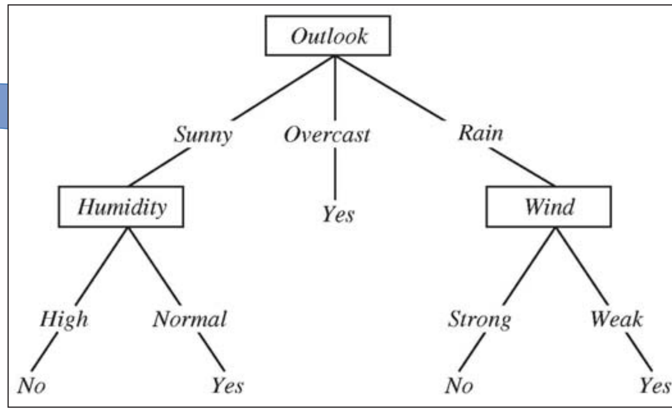
In worst case, exponentially many nodes needed

Decision trees have variable-sized hypothesis space

- As #nodes (or depth) increases, hypothesis space grows (can express more complex functions)

Slide credit: Eric Eaton

# Converting Decision Trees to Rules




| | |
|---|---|
| IF | (*Outlook* = *Sunny*) AND (*Humidity* = *High*) |
| THEN | *PlayTennis* = *No* |
| IF | (*Outlook* = *Sunny*) AND (*Humidity* = *Normal*) |
| THEN | *PlayTennis* = *Yes* |
| . . . | |

Slide credit: Eric Eaton
Based on slide by Pedro Domingos

# Decision Tree Algorithm

Learning Goals

- Define heuristic for optimal decision trees
- Describe ID3 heuristic algorithm
- Define entropy, conditional entropy, information gain
- Build a decision tree by hand

# Another Example: Restaurant Domain (Russell & Norvig)

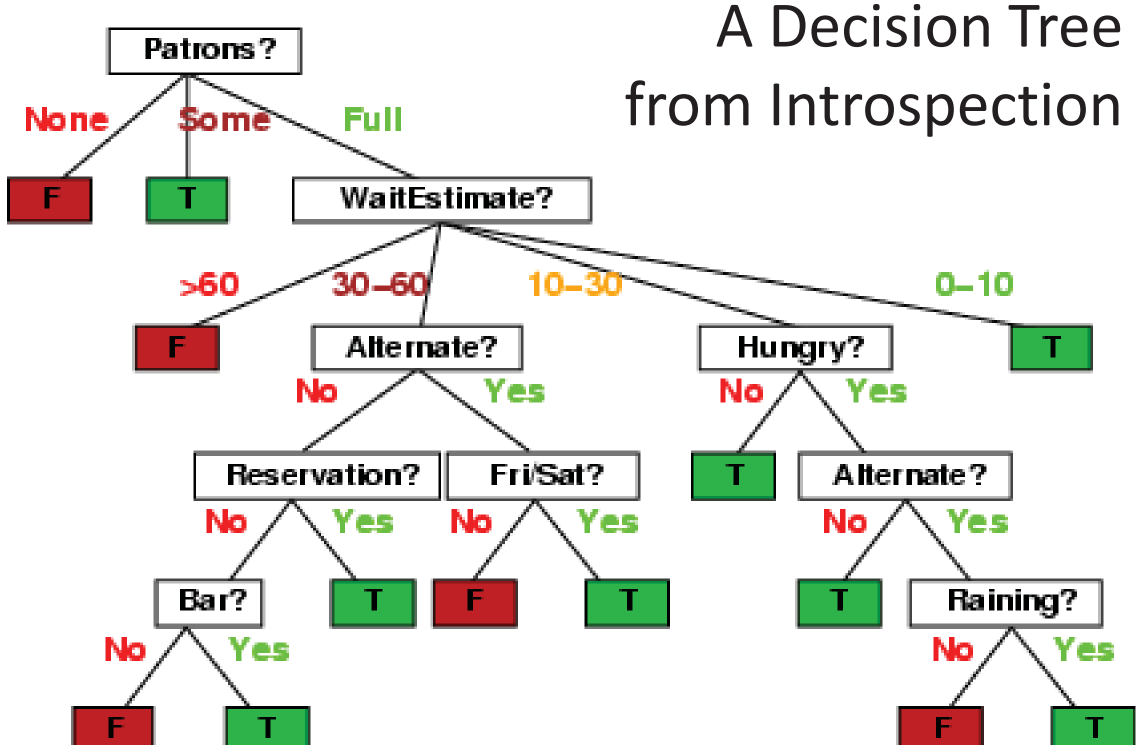
Model patron's decision of whether to wait for table at restaurant

| Example | Attributes | | | | | | | | | | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *Wait* |
| $X_1$ | T | F | F | T | Some | \$\$\$ | F | T | French | 0–10 | T |
| $X_2$ | T | F | F | T | Full | \$ | F | F | Thai | 30–60 | F |
| $X_3$ | F | T | F | F | Some | \$ | F | F | Burger | 0–10 | T |
| $X_4$ | T | F | T | T | Full | \$ | F | F | Thai | 10–30 | T |
| $X_5$ | T | F | T | F | Full | \$\$\$ | F | T | French | >60 | F |
| $X_6$ | F | T | F | T | Some | \$\$ | T | T | Italian | 0–10 | T |
| $X_7$ | F | T | F | F | None | \$ | T | F | Burger | 0–10 | F |
| $X_8$ | F | F | F | T | Some | \$\$ | T | T | Thai | 0–10 | T |
| $X_9$ | F | T | T | F | Full | \$ | T | F | Burger | >60 | F |
| $X_{10}$ | T | T | T | T | Full | \$\$\$ | F | T | Italian | 10–30 | F |
| $X_{11}$ | F | F | F | F | None | \$ | F | F | Thai | 0–10 | F |
| $X_{12}$ | T | T | T | T | Full | \$ | F | F | Burger | 30–60 | T |

~7,000 possible cases

Slide credit: Eric Eaton

# A Decision Tree from Introspection




Is this the best decision tree?

Slide credit: Eric Eaton

# Preference bias: Ockham's Razor

Principle stated by William of Ockham (1285-1347)

"*non sunt multiplicanda entia praeter necessitatem*"

- entities are not to be multiplied beyond necessity
- AKA Occam's Razor, Law of Economy, or Law of Parsimony

**Idea**: The simplest consistent explanation is the best

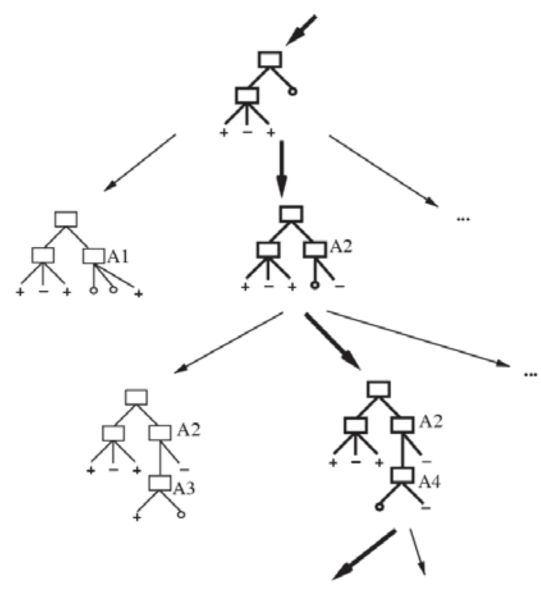
Therefore, the smallest decision tree that correctly classifies all of training examples is best

- finding provably smallest decision tree is NP-hard
- ... so instead of constructing the absolute smallest tree consistent with training examples, construct one that is pretty small

Slide credit: Eric Eaton

# Hypothesis Search Space

- Heuristic search through space of decision trees
  - search space is exponential
- Stops at smallest acceptable tree (ID3)
  - do not get globally optimal tree




Based on slide by Tom Mitchell

# Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

*node* = root of decision tree

Main loop:

1. $\boxed{A \leftarrow \text{"best" decision attribute (feature) for next node.}}$
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recur over new leaf nodes.

How do we choose which attribute is best?

Based on slide by Eric Eaton
[originally by Tom Mitchell]

This slide intentionally blank.

# Choosing the Best Attribute

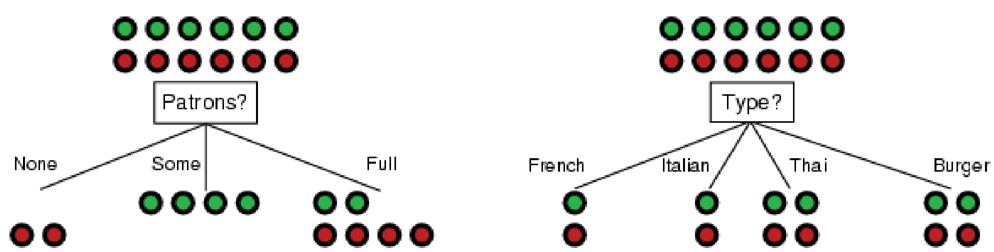
**Key problem**: choosing which attribute to split given set of examples

- Some possibilities are:
  - **Random:** select any attribute at random
  - **Least-Values:** choose the attribute with the smallest number of possible values
  - **Most-Values:** choose the attribute with the largest number of possible values
  - **Max-Gain:** choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- ID3 algorithm uses Max-Gain method of selecting the best attribute

Slide credit: Eric Eaton

# Choosing an Attribute

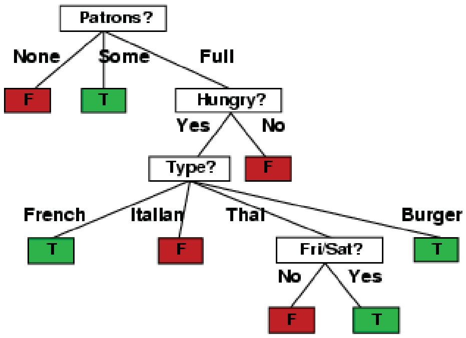
**Idea**: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



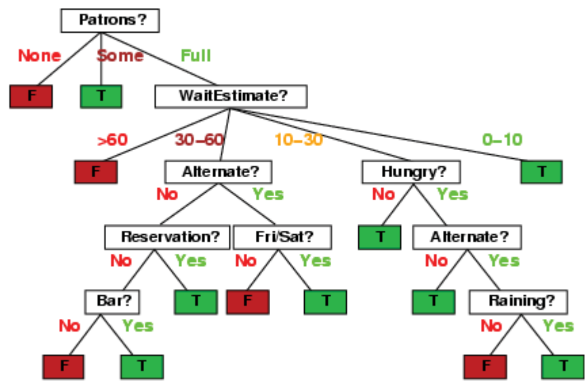

Which split is more informative: *Patrons?* or *Type?*

Slide credit: Eric Eaton
Based on slide by M. desJardins & T. Finin

# ID3-induced Decision Tree




compare to introspection




Based on slide by Eric Eaton
(originally by M. desJardins & T. Finin)

# Let's build a tree

| Movie | Type | Length | Director | Famous actors | Liked? |
|---|---|---|---|---|---|
| m1 | Comedy | Short | Adamson | No | Yes |
| m2 | Animated | Short | Lasseter | No | No |
| m3 | Drama | Medium | Adamson | No | Yes |
| m4 | Animated | Long | Lasseter | Yes | No |
| m5 | Comedy | Long | Lasseter | Yes | No |
| m6 | Drama | Medium | Singer | Yes | Yes |
| m7 | Animated | Short | Singer | No | Yes |
| m8 | Comedy | Long | Adamson | Yes | Yes |
| m9 | Drama | Medium | Lasseter | No | Yes |

Features: Type, Length, Director, Famous actors — Labels: Liked?

AP Breaking News

**Netflix offers $1 million prize for better movie recommendations**

By MICHAEL LIEDTKE, AP Business Writer
Sunday, October 1, 2006

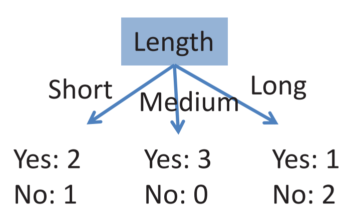
(10-01) 21:06 PDT San Francisco (AP) --

Online DVD rental pioneer Netflix Inc. wants recommendations on how to improve its movie-recommendation system so badly that it's dangling a $1 million reward as an incentive.

The prize, offered in a contest scheduled to begin Monday, is part of Netflix's effort to sharpen its competitive edge as it continues a bitter duel with Blockbuster Inc. and prepares for an anticipated onslaught of services that make it easier to download movies on to computer hard drives.

*Printable Version
*Email This Article

Business & Finance
*Get Quote:
Detailed
Submit
Symbol Lookup




How much information do we gain from this split?

Based on slide by Ziv Bar-Joseph

# Entropy

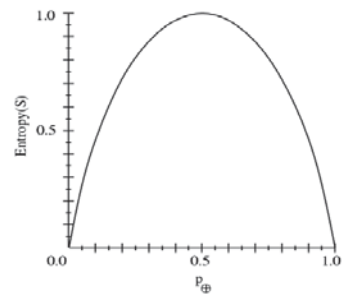
- Measures amount of **uncertainty** of random variable (with specific probability distribution)
- Higher it is, less confident we are in its outcome

**Definition**

Entropy $H(X)$ of r.v. $X$

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

($n$ = # of possible values for $X$)




Example

- coin flip ($n$ = 2)
  $H(X) = -P(X=0)\log_2 P(X=0) - P(X=1)\log_2 P(X=1)$
- if $P(X=1) = p_\oplus = 1$
  $H(X) = -1{\cdot}\log_2(1) - 0{\cdot}\log_2(0) = 0$ (no uncertainty)
- if $P(X=1) = p_\oplus = 0.5$
  $H(X) = -0.5{\cdot}\log_2(0.5) - 0.5{\cdot}\log_2(0.5) = -\log_2(0.5) = 1$

Based on slides by Ziv Bar-Joseph and Tom Mitchell

# Interpreting Entropy

Measure of **impurity**


Use **information theory**

- Assume both sender and receiver know distribution
- How many bits, on average, would it take to transmit one symbol?
  - if $P(X=1) = 1$
    - then answer is $0$ (we do not need to transmit anything)
  - if $P(X=1) = 0.5$
    - then answer is $1$ (transmit one bit for each equally likely message)
  - if $0 < P(X=1) < 0.5$ or $0.5 < P(X=1) < 1$
    - then answer is between $0$ and $1$ (why?)

Based on slides by Eric Eaton and Ziv Bar-Joseph

# Expected Bits Per Symbol

Assume $P(X = 1) = 0.8$

Then $P(X = 11) = 0.64$
$P(X = 10) = P(X = 01) = 0.16$
$P(X = 00) = 0.04$

Define the following code

- for `11`, send `0`
- for `10`, send `10`
- for `01`, send `110`
- for `00`, send `1110`

| | |
|---|---|
| so message | `01001101110001101110` |
| can be broken into | `01 00 11 01 11 00 01 10 11 10` |
| which is encoded as | `110 1110 0 110 0 1110 110 10 0 10` |

What is the expected bits / symbol?

$$(0.64)(1) + (0.16)(2) + (0.16)(3) + (0.04)(4) = 1.6 \;/\; 2 = 0.8$$

Entropy $H(X) = -\,0.8\log_2(0.8) - 0.2\log_2(0.2) = 0.7219$
(entropy is lower bound for bits / symbol)

Based on slide by Ziv Bar-Joseph

This slide intentionally blank.

# Conditional Entropy

- Entropy measures uncertainty in specific distribution
- What if we have additional information?

| Length | Liked? |
|---|---|
| Short | Yes |
| Short | No |
| Medium | Yes |
| Long | No |
| Long | No |
| Medium | Yes |
| Short | Yes |
| Long | Yes |
| Medium | Yes |

- For example, say I want to know the label ($\mathrm{Liked}$) entropy when $\mathrm{Length}$ is known
- This becomes a **conditional entropy** problem

$$H(\mathrm{Liked} \mid \mathrm{Length} = v)$$

is the entropy of $\mathrm{Liked}$ among movies with $\mathrm{Length}\ v$

Based on slide by Ziv Bar-Joseph

# Conditional Entropy : Examples for Specific Values

| Length | Liked? |
|---|---|
| Short | Yes |
| Short | No |
| Medium | Yes |
| Long | No |
| Long | No |
| Medium | Yes |
| Short | Yes |
| Long | Yes |
| Medium | Yes |

Compute $H(\mathrm{Li} \mid \mathrm{Le} = v)$

$$H(\mathrm{Li} \mid \mathrm{Le} = \mathrm{S}) =$$

$$H(\mathrm{Li} \mid \mathrm{Le} = \mathrm{M}) =$$

$$H(\mathrm{Li} \mid \mathrm{Le} = \mathrm{L}) =$$

Based on slide by Ziv Bar-Joseph

# Conditional Entropy

| Length | Liked? |
|---|---|
| Short | Yes |
| Short | No |
| Medium | Yes |
| Long | No |
| Long | No |
| Medium | Yes |
| Short | Yes |
| Long | Yes |
| Medium | Yes |

- We can generalize the conditional entropy idea to determine $H(\mathrm{Li} \mid \mathrm{Le})$
- That is, what is the expected uncertainty if we already know the value of $\mathrm{Le}$ for each of the records (samples)

**Definition**

Conditional Entropy $H(X|\,Y)$ of $X$ given $Y$

$$H(X|Y) = \sum_{v\ \in values(Y)} P(Y=v)\underbrace{H(X|Y=v)}_{\text{Specific Conditional Entropy (we explained how to compute this in previous slide)}}$$

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v)\log_2 P(X=i|Y=v)$$



# Conditional Entropy Example

| Length | Liked? |
|---|---|
| Short | Yes |
| Short | No |
| Medium | Yes |
| Long | No |
| Long | No |
| Medium | Yes |
| Short | Yes |
| Long | Yes |
| Medium | Yes |

$$H(X|Y) = \sum_{v\ \in values(Y)} P(Y=v)H(X|Y=v)$$

Compute $H(\mathrm{Li} \mid \mathrm{Le})$

# Information Gain

- How much do we gain from knowing one of the attributes?
- In other words, what is the reduction in entropy from this knowledge?

**Definition**
Mutual Information (aka Info Gain) of $X$ and $Y$

$$I(X,Y) = H(X) - H(X|Y)$$

Notes
- $I(X,Y) = H(Y) - H(Y|X)$
- $I(X,Y)$ is always $\geq 0$ (proof: Jensen's inequality)

Based on slides by Ziv Bar-Joseph and Tom Mitchell

# Information Gain in Decision Trees

Information Gain …
- is the mutual information between input attribute $A$ and target variable $Y$
- is the expected reduction in entropy of target variable $Y$ for data sample $S$, due to sorting on variable $A$

$$Gain(S, A) = I_S(A,Y) = H_S(Y) - H_S(Y|A)$$

$$[Gain = H(parent) - weighted\ average\ H(children)]$$

- tells us how important a given attribute of the feature vector is

Based on slides by Tom Mitchell

# Where We Are

- We were looking for a good criteria for selecting the best attribute for a node split
- We defined entropy, conditional entropy and information gain
- We will now use information gain as our criteria for a good split

*node* = root of decision tree

Main loop:

based on information gain

1. *A* ← "best" decision attribute for next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recur over new leaf nodes.

Based on slides by Ziv Bar-Joseph and Eric Eaton

# Example : Building a Decision Tree

$$P(\text{Li} = \text{yes}) = 2/3$$
$$H(\text{Li}) = 0.92$$

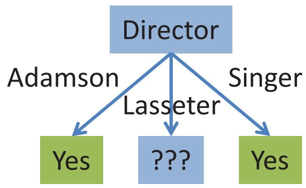
$$H(\text{Li} \mid \text{T}) = 0.61$$
$$H(\text{Li} \mid \text{Le}) = 0.61$$
$$H(\text{Li} \mid \text{D}) = 0.36$$
$$H(\text{Li} \mid \text{F}) = 0.85$$

| Movie | Type | Length | Director | Famous actors | Liked? |
|---|---|---|---|---|---|
| m1 | Comedy | Short | Adamson | No | Yes |
| m2 | Animated | Short | Lasseter | No | No |
| m3 | Drama | Medium | Adamson | No | Yes |
| m4 | Animated | Long | Lasseter | Yes | No |
| m5 | Comedy | Long | Lasseter | Yes | No |
| m6 | Drama | Medium | Singer | Yes | Yes |
| m7 | Animated | Short | Singer | No | Yes |
| m8 | Comedy | Long | Adamson | Yes | Yes |
| m9 | Drama | Medium | Lasseter | No | Yes |

$$I(\text{Li}, \text{T}) =$$
$$I(\text{Li}, \text{Le}) =$$
$$I(\text{Li}, \text{D}) =$$
$$I(\text{Li}, \text{F}) =$$

Based on slide by Ziv Bar-Joseph

# Example : Building a Decision Tree




$$P(\mathrm{Li} = \mathrm{Y}) = 0.25$$
$$H(\mathrm{Li}) = 0.81$$

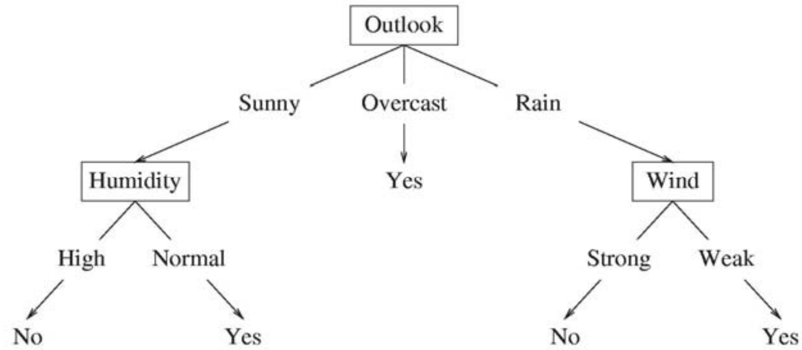
| Movie | Type | Length | Famous actors | Liked? |
|---|---|---|---|---|
| m2 | Animated | Short | No | No |
| m4 | Animated | Long | Yes | No |
| m5 | Comedy | Long | Yes | No |
| m9 | Drama | Medium | No | Yes |

we eliminated the 'director' attribute (since all samples have the same director)

Based on slide by Ziv Bar-Joseph

This slide intentionally blank.

# Summary of Decision Trees (so far)




Decision tree induction

- Split based on attribute values, choosing best (based on information gain)
- Build tree greedily, recursing on children of split
- Stop when all instances in child have same class

Based on slide by Eric Eaton
(originally by Pedro Domingos)

This slide intentionally blank.

# Decision Tree Overfitting

Learning Goals

- Define overfitting
- Describe how to avoid overfitting decision trees

# Noisy Data

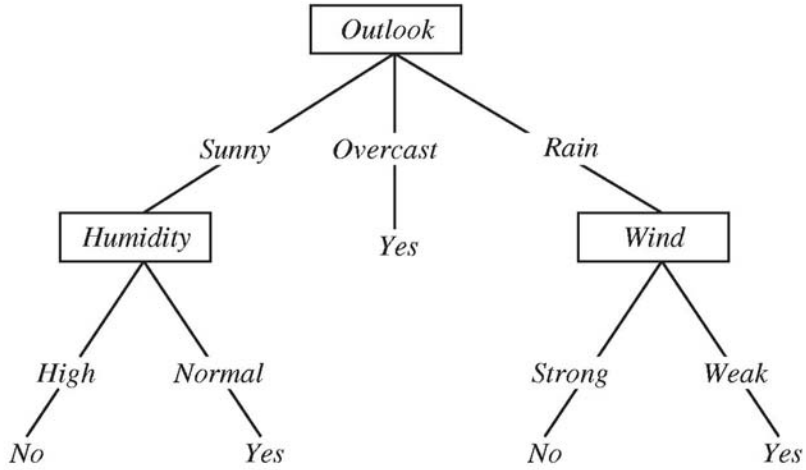
Many kinds of "**noise**" can occur in the examples

- **Erroneous** training data
  - two examples have same attribute/value pairs, but different classifications
  - feature noise
    - some values of attributes are incorrect because of errors in data acquisition process or preprocessing phase
  - label noise
    - instance was labeled incorrectly (+ instead of -)

  $\Rightarrow$ **means training error not guaranteed to be 0**

# Overfitting in Decision Trees

Consider adding a noisy training example to the following tree:

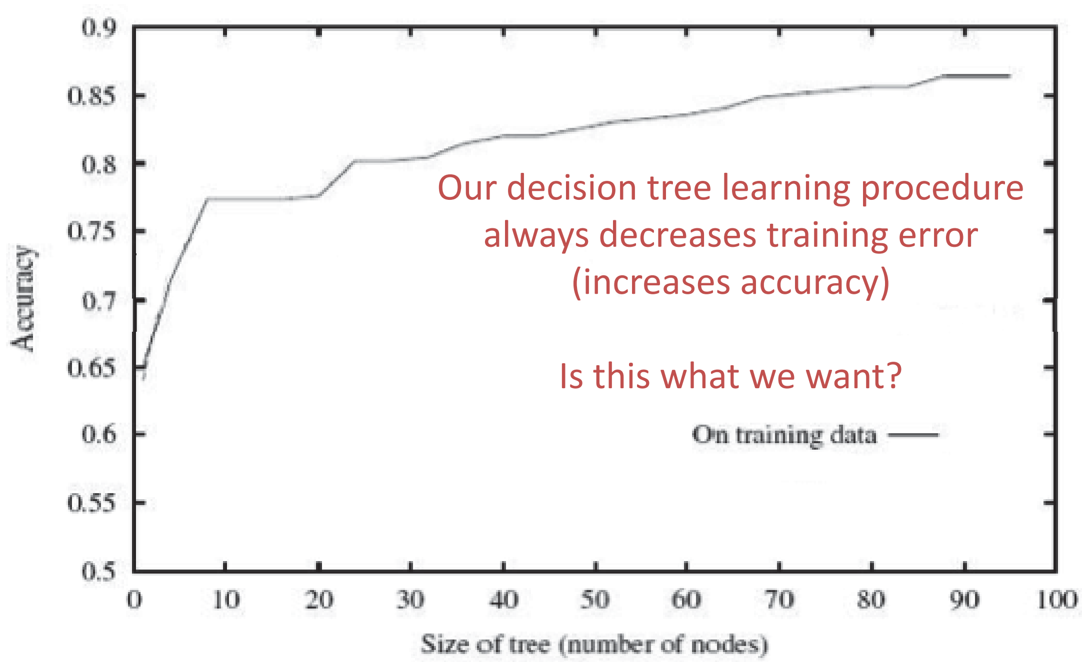



What would be the effect of adding:

(outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No) ?

Slide credit: Eric Eaton
(originally by Pedro Domingos)

# Overfitting

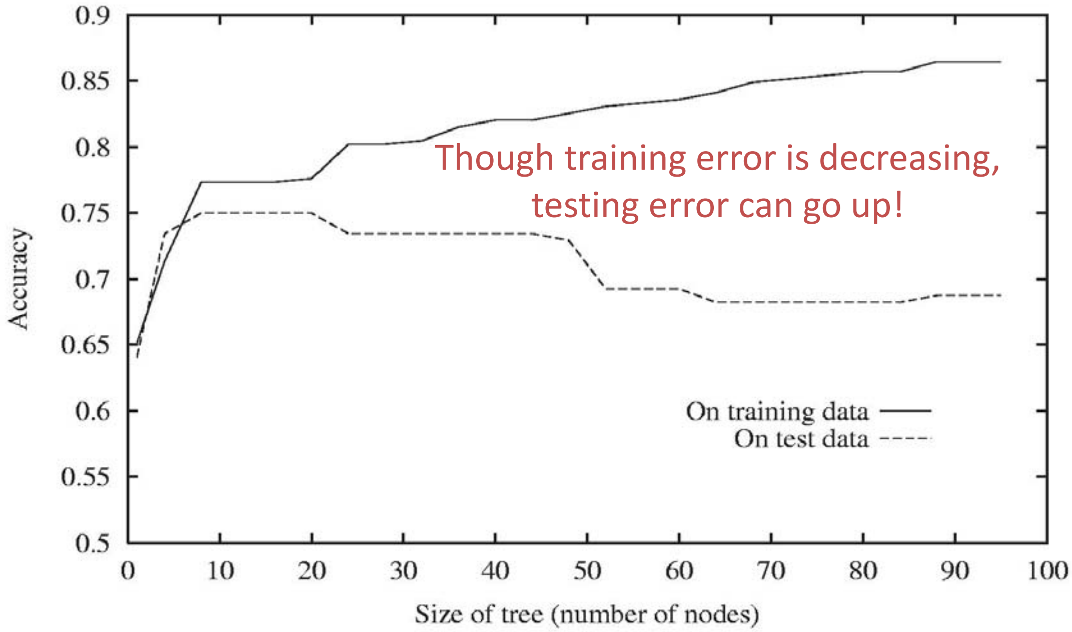



Our decision tree learning procedure always decreases training error (increases accuracy)

Is this what we want?

Based on slide by David Kauchak
(originally by Pedro Domingos)

# Overfitting in Decision Tree Learning




Based on slide by David Kauchak
(originally by Pedro Domingos)

# Overfitting

Consider error of hypothesis $h$ over

- training data : $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data : $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

**Idea**

- model biased too much towards training data
- remember, goal is to learn a **general** model that will work on the training data as well as other data (i.e. test data)

Based on slides by David Kauchak and Pedro Domingos

# Overfitting

Much more significant sources of “noise”

- Some attributes are **irrelevant** to decision-making process
  - if hypothesis space has many dimensions (large # of attributes), may find **meaningless regularity** in data that is irrelevant to true, important, distinguishing features
- Target function is **non-deterministic** in attributes
  - in general, we cannot measure all variables needed to do perfect prediction $\Rightarrow$ target function is not uniquely determined by attribute values
  - if too little training data, even a reasonable hypothesis will “overfit”

Based on slides by Eric Eaton (originally by M. desJardins & T. Finin) and Sara Sood

# Avoiding Overfitting

How can we avoid overfitting?

- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)

Decision Tree Pruning

- Prune while building tree (**stopping early**)
- Prune after building tree (**post-pruning**)

How to select “best” tree

- Statistical tests
- Measure performance over separate validation set

Slide credit: Eric Eaton
(originally by Pedro Domingos)

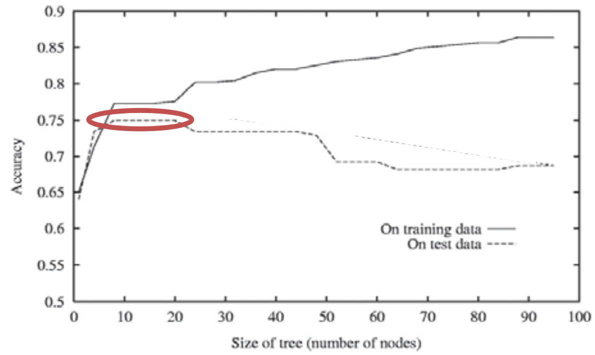
# Depth-Limited Decision Trees

Split data into *training* and *validation* sets

Grow tree with max depth *d* based on *training set*

Evaluate each tree on *validation* set

Pick tree with best performance




Slide credit: Eric Eaton
Based on slide by Pedro Domingos

# Ranking Classifiers

| MODEL | CAL | ACC | FSC | LFT | ROC | APR | BEP | RMS | MXE | MEAN | OPT-SEL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BST-DT | PLT | .843* | .779 | **.939** | **.963** | **.938** | .929* | **.880** | **.896** | **.896** | **.917** |
| RF | PLT | .872* | .805 | .934* | .957 | .931 | **.930** | .851 | .858 | .892 | .898 |
| BAG-DT | – | .846 | .781 | .938* | .962* | .937* | .918 | .845 | .872 | .887* | .899 |
| BST-DT | ISO | .826* | .860* | .929* | .952 | .921 | .925* | .854 | .815 | .885 | .917* |
| RF | – | **.872** | .790 | .934* | .957 | .931 | **.930** | .829 | .830 | .884 | .890 |
| BAG-DT | PLT | .841 | .774 | .938* | .962* | .937* | .918 | .836 | .852 | .882 | .895 |
| RF | ISO | .861* | **.861** | .923 | .946 | .910 | .925 | .836 | .776 | .880 | .895 |
| BAG-DT | ISO | .826 | .843* | .933* | .954 | .921 | .915 | .832 | .791 | .877 | .894 |
| SVM | PLT | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| ANN | – | .803 | .762 | .910 | .936 | .892 | .899 | .811 | .821 | .854 | .885 |
| SVM | ISO | .813 | .836* | .892 | .925 | .882 | .911 | .814 | .744 | .852 | .882 |
| ANN | PLT | .815 | .748 | .910 | .936 | .892 | .899 | [illegible] | [illegible] | [illegible] | [illegible] |
| ANN | ISO | .803 | .836 | .908 | .924 | .876 | .891 | [illegible] | [illegible] | [illegible] | [illegible] |
| BST-DT | – | .834* | .816 | **.939** | **.963** | **.938** | .929 | [illegible] | [illegible] | [illegible] | [illegible] |
| KNN | PLT | .757 | .707 | .889 | .918 | .872 | .872 | [illegible] | [illegible] | [illegible] | [illegible] |
| KNN | – | .756 | .728 | .889 | .918 | .872 | .872 | [illegible] | [illegible] | [illegible] | [illegible] |
| KNN | ISO | .755 | .758 | .882 | .907 | .854 | .869 | [illegible] | [illegible] | [illegible] | [illegible] |
| BST-STMP | PLT | .724 | .651 | .876 | .908 | .853 | .845 | [illegible] | [illegible] | [illegible] | [illegible] |
| SVM | – | .817 | .804 | .895 | .938 | .899 | .913 | [illegible] | [illegible] | [illegible] | [illegible] |
| BST-STMP | ISO | .709 | .744 | .873 | .899 | .835 | .840 | [illegible] | [illegible] | [illegible] | [illegible] |
| BST-STMP | – | .741 | .684 | .876 | .908 | .853 | .845 | [illegible] | [illegible] | [illegible] | [illegible] |
| DT | ISO | .648 | .654 | .818 | .838 | .756 | .778 | [illegible] | [illegible] | [illegible] | [illegible] |
| DT | – | .647 | .639 | .824 | .843 | .762 | .777 | [illegible] | [illegible] | [illegible] | [illegible] |
| DT | PLT | .651 | .618 | .824 | .843 | .762 | .777 | [illegible] | [illegible] | [illegible] | [illegible] |
| LR | – | .636 | .545 | .823 | .852 | .743 | .734 | [illegible] | [illegible] | [illegible] | [illegible] |
| LR | ISO | .627 | .567 | .818 | .847 | .735 | .742 | [illegible] | [illegible] | [illegible] | [illegible] |
| LR | PLT | .630 | .500 | .823 | .852 | .743 | .734 | [illegible] | [illegible] | [illegible] | [illegible] |
| NB | ISO | .579 | .468 | .779 | .820 | .727 | .733 | .572 | .555 | .654 | .661 |
| NB | PLT | .576 | .448 | .780 | .824 | .738 | .735 | .537 | .559 | .650 | .654 |
| NB | – | .496 | .562 | .781 | .825 | .738 | .735 | .347 | -.633 | .481 | .489 |

Top 8 all based on various extensions of decision trees

Based on slide by Ziv Bar-Joseph
[Source: Rich Caruana & Alexandru Niculescu-Mizil,
An Empirical Comparison of Supervised Learning Algorithms, ICML 2006]

# Summary: Decision Trees

**Widely used in practice**
- works very (very) well

**Advantages**
- one of most intuitive classifiers
- no prior assumption on data
- fast and simple to implement
- can convert to rules
- handles noisy data

**Disadvantages**
- univariate splits / partitioning on only one attribute limits types of possible learners (e.g. cannot learn simple linearly separable data sets)
- large trees may be hard to understand, slow, and perform poorly
- pruning / tuning can be tricky to get right
- requires fixed-length feature vectors
- non-incremental (i.e. batch method)
- sacrifices predictive power

Based on slides by Eric Eaton, Ziv Bar-Joseph, and David Kauchak

(Extra Slides)

# Decision Tree Implementation

Learning Goals

- Describe ID3 in pseudocode
- Identify problems in and propose solutions for decision tree implementations

# Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

*node* = root of decision tree
Main loop:

1. *A* ← "best" decision attribute for next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recur over new leaf nodes.

see any problems?

what if *S* is empty?

what if all examples have same label?

what if *A* is empty or all examples have same values for attributes?

```
function BuildTree(S, A, C)          // S : training set, A : input attributes, C : class attribute
    a ← "best" decision attribute among A using S
    tree ← new tree with root node assigned attribute a
    for each value v_k of a
        S_k ← examples from S with value v_k for attribute a
        subtree ← BuildTree(S_k, A – a, C)
        add subtree as child of tree with branch labeled A = v_k
    return tree
```

recursion missing base case?

# Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

```
function BuildTree(n, A, C)          // n : training set, A : input attributes, C : class attribute
    if n is empty
        return leaf node with failure
    else if all samples in n have same label lbl for C
        return leaf node with lbl
    else if A is empty or all n have same feature values
        return leaf node with majority vote of values of C in n
    else
        a ← "best" decision attribute among A using n
        tree ← new tree with root node assigned attribute a
        for each value v_k of a
            n_k ← examples from n with value v_k for attribute a
            subtree ← BuildTree(n_k, A – a, C)
            add subtree as child of tree with branch labeled A = v_k
        return tree
```

alternatively keep track of training set at parent (if exists) and return majority vote of parent

# Gain Ratio

Problem

- Information gain biased towards attributes with many values
  - Imagine using *date = jun_3_1996* as attribute

Alternative

- Use gain ratio instead

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) = -\sum_{v \in values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

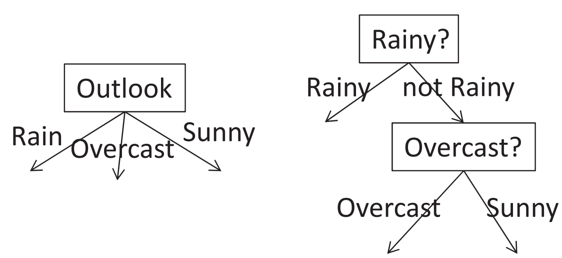
  where $S_v$ is subset of $S$ for which $A = v$
  - $SplitInformation$ measures “intrinsic value” of attribute

Based on slide by Eric Eaton
(originally by Pedro Domingos)

# Non-Boolean Features

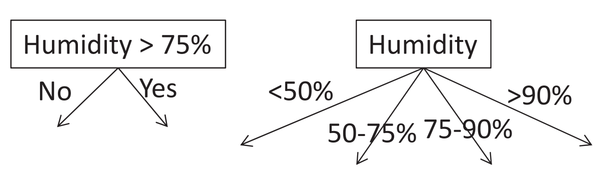
Features with multiple discrete values

- Construct multi-way splits
- Test for one value versus rest
- Group values into disjoint subsets




Real-valued features

- Use threshold split or discretize
- Use information gain or sort and identify adjacencies with different classes




Based on slides by Eric Eaton, Ziv Bar-Joseph, Pedro Domingos, and David Kauchak