

Artificial Intelligence

Machine Learning I – Decision Tree Learning

10/02/2002

What is Learning?

- Reorganization of existing knowledge
 - Typically as a result of acquiring new knowledge
- Improvement through experience
- Inductive inference:
 - Arrives at general conclusions by examining specific examples
- Set of examples is called the *training set*

Inductive Learning Scenarios

- Computer boardgame player loses game
 - We want it to learn from its mistakes
- Security robot distinguishes friend from foe
 - Learn to distinguish based on behavioral cues
- Search engine learns to distinguish useful and useless web pages
 - Possibly according to multiple criteria
- A legged robot learns to walk
 - Experiences “pain” when it falls

Online vs. Batch Learning

- Batch learning
 - System processes a large set of examples at once
 - System typically learns before deployment
- Online learning
 - Examples processed one at a time
 - System can learn during deployment
 - Online learning can implement batch learning

Supervision and Learning

- Learning programs try to infer associations between specific inputs/outputs
- Supervised learning:
 - System given correct output for each input
- Unsupervised learning
 - System not given correct output
 - Must rely upon other cues
 - Canonical example: reinforcement learning

Supervised Learning

- Training set contains input/output pairs called *examples*
- Program learns a function f that:
 - Accounts for examples seen so far
 - Makes a good guess for the outputs of inputs it has not seen
- Ability to generalize tested using a different example set

Examples of Inputs and Outputs

- Object description/Object classification
 - Pattern recognition applications
- Situation/Prediction
 - What food might this child eat?
 - If anything is green, he won't eat it!
- Situation/Action
 - If the road veers left, steer to the left

Classification/Concept Learning

- If the learned function is discrete-valued:
 - Outputs are called *classes*
 - Learning is called *classification*
- If there are only two possible outputs:
 - Learned function is called a *concept*
 - Each input either does or does not satisfy the concept

Training Examples

- Examples are logical statements in CNF
- Propositions represent “raw” data (inputs)
- Associated with each CNF is the yes/no output
- Each CNF has the same number of terms
- Each term represents the example’s value along a particular *dimension*

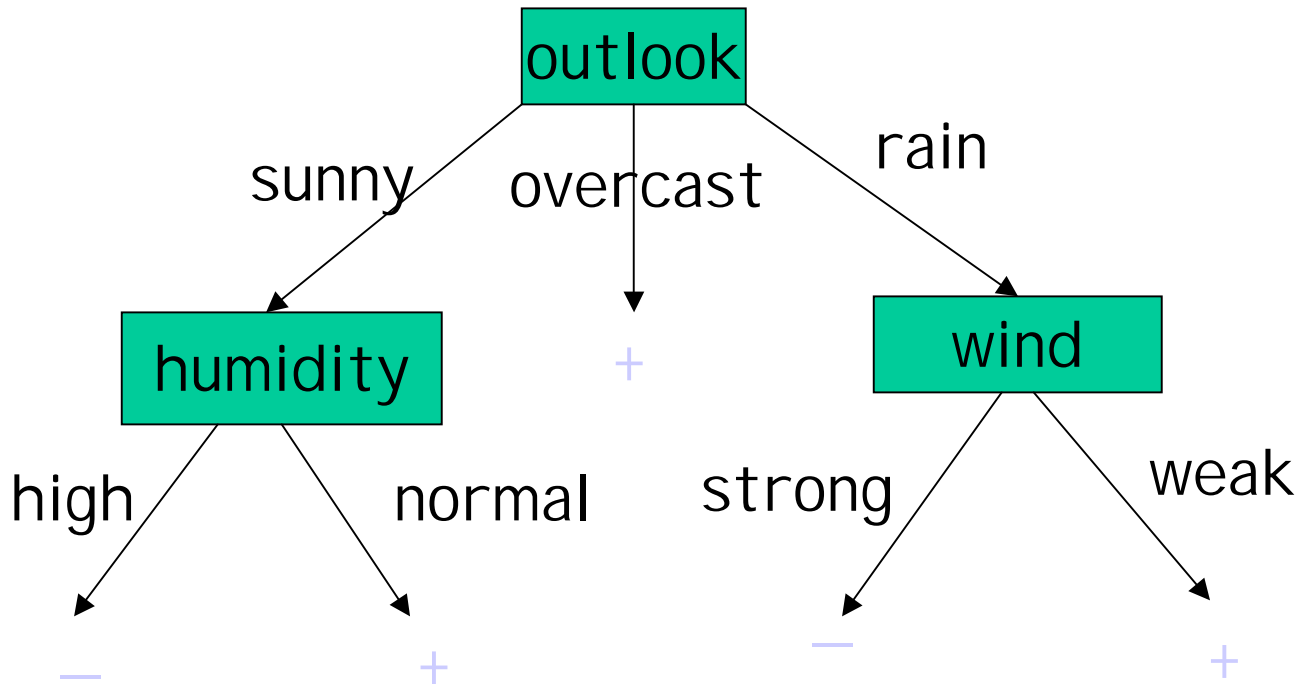
Dimensions

- Each example has a value for each dimension
- Values for each dimension are discrete

Decision tree learning

- Decision tree representation
- The ID3 decision tree learning algorithm
- Choosing a good order for attributes in the tree
 - Information gain
- Evaluating decision trees
- Overfitting: how to control it

A decision tree



Outlook, humidity and wind are *attributes*.

Decision tree representation

- Each internal node tests an attribute.
- Each branch corresponds to an attribute value.
- Each leaf node assigns a classification.
- **To classify a new example:**
 - **Start at the root of the decision tree. Find the value of the tested attribute in the given example, taking the branch appropriate to that value.**
 - **Continue in the same fashion until a leaf is reached.**
 - **The leaf gives you the class of your example.**
- Note that the cost (time complexity) of classifying examples depends on the length of the path from the root of the tree to the appropriate leaf.

Learning a decision tree hypothesis

Need to use training examples...

Training examples

Day	Outlook	Temp	Humidity	Wind	Play-Tennis
D1	sunny	hot	high	weak	-
D2	sunny	hot	high	strong	-
D3	overcast	hot	high	weak	+
D4	rain	mild	high	weak	+
...
...
D14	rain	mild	high	strong	-

Induction of decision tree hypotheses with ID3

- Function ID3 takes as inputs
 - A set of labeled training examples called Examples
 - A set Attributes of all attributes
- Returns
 - A decision tree that correctly classifies the training examples

The ID3 algorithm

- ID3(Examples, Attributes)
 - Create a root node for this (sub-)tree.
 - If all examples are labeled + (resp. –) return the single node tree root with label “+” (resp. label “–”).
 - If Attributes is empty, return the single node tree root with label equal to the most common label of examples in Examples.
 - This can happen if the examples are noisy or the attributes are inadequate to learn the target concept.

The ID3 algorithm (contd.)

- Otherwise:
 - Let A be the attribute from $Attributes$ that best classifies $Examples$.
 - Make A the root of this (sub-)tree.
 - For each possible value v of A do:
 - Let $Examples(v)$ be the subset of $Examples$ for which $A = v$
 - Add a new tree branch below root corresponding to the test $A = v$
 - If $Examples(v)$ is empty, then add a new leaf with label equal to the most common value of A in $Examples$;
Else add the sub-tree $ID3(Examples(v), (Attributes - \{A\}))$

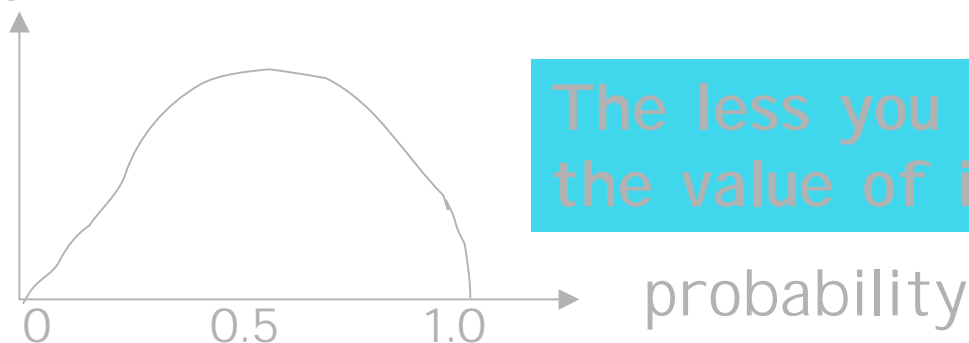
Tree Size

- Smaller (i.e., simpler) trees generalize more and therefore tend to have better predictive capabilities on many real-world problems (Principle of *Occam's Razor*).
- The size of the tree depends on the order in which attributes are tested.
- Therefore, we want to choose an attribute order that generates a small tree.

Picking the best attribute to achieve a small tree

- We use the *information-theoretic* measure of *entropy* to determine which attribute to split the tree on.
- Information theory was introduced by Shannon and Weaver (1949) in a “Mathematical Theory of Communication.”
- Information theory says that there are 2 ways to increase information/entropy:
 - Increase the number of choices
 - Make the probabilities of the choices more equal

Entropy(S)



The less you know, the greater the value of information

Information theory

- Information theory measures the value of information in **bits**.
- Suppose a message tells you the correct answer from a number of choices. Then, in the simplest case, the amount of information/entropy of this message is equal to the \log_2 of the number of choices. This is the *optimal length code* for such a message.
- Why logarithm?
 - Want the information, when there is one on/off relay (2 choices), to be 1 bit of information. We get this with $\log_2 2$.
 - One relay: 0 or 1 are the choices (on or off)
 - Want the information, when there are 3 relays ($2^3 = 8$ choices) to be 3 times as much (or 3 bits of) information. We get this with $\log_2 8 = 3$.
 - Three relays: 000, 001, 011, 010, 100, 110, or 111 give possible values for all three relays.
- Why base 2?
 - We are expressing message length in terms of the number of bits.

Information theory

- In general, the optimal length code for a message v with probability $p(v)$ is: $-\log_2 p(v)$. If there are multiple possible values, to get the optimal length code we weight the possible values by their probabilities.
- If we have v_i possible values, each with probability $p(v_i)$, then the information content (entropy, or optimal length code) of the actual value (the message) is given by:

$$\text{Entropy/Info} = \sum_{i=1}^n -p(v_i) \log_2 p(v_i)$$

This entropy formula comes from statistical mechanics and thermodynamics

- In the previous example, there were 8 possible values for the on/off status of 3 relays. We can use this formula to determine the information content of a message that tells you which of these relays will be chosen.
- As another example, the information from tossing a fair coin is:
 - Entropy/Info = $-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$ bit.
 - As the probability of heads goes to 1, the Entropy/Info goes to 0 bits.

Picking the best attribute to achieve a small tree

- Let S = the set of training examples.
- Entropy(S) = the expected number of bits needed to encode the class (+ or -) of a randomly drawn member of S assuming the optimal, shortest length code. *Greater entropy -> less desirable (because it implies a bigger tree).*
- We estimate entropy/info by counting examples. This applies the frequency notion of probability.
- **Entropy(S) = $-(p/p+n) \log_2 (p/p+n) - (n/p+n) \log_2 (n/p+n)$**
 - $p/p+n$ = proportion of positive examples in S
 - $n/p+n$ = proportion of negative examples in $S = (1 - p/p+n)$

Call this
 $E(A)$

Information Gain

- **Gain(S,A)** is the expected *reduction* in entropy due to a split of the examples in S on attribute A.
- Let S_v be the subset of the training examples with value v of attribute A. Then

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- Gain(S,A) is the total information in the tree minus the information needed to complete the tree after testing on A.
 - The information needed to complete the tree, $E(A)$, is the weighted average of the information in all its subtrees.
- ***We want to keep choosing the attribute with the biggest Gain.***

Example

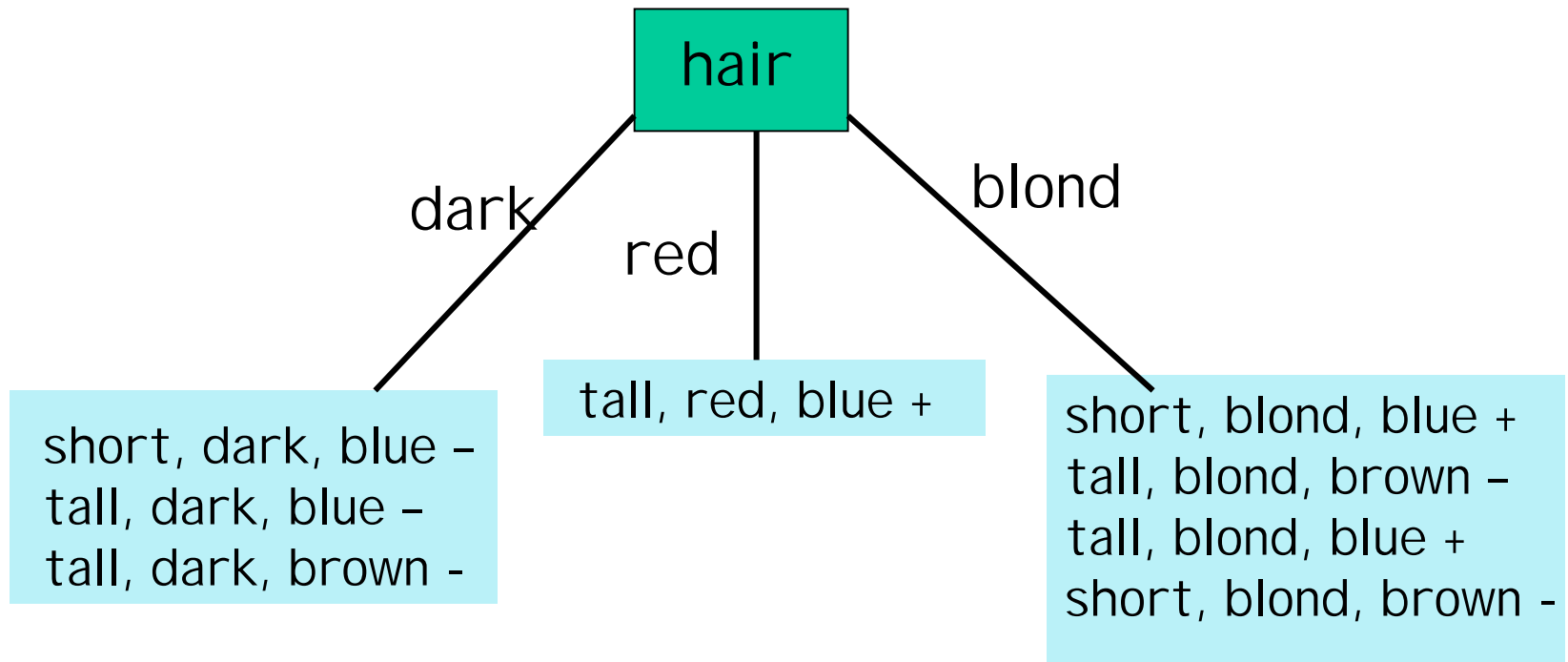
Let Attributes be {height, hair, eyes}. Let S be:

short, blond, blue: +
tall, blond, brown: -
tall, red, blue: +
short, dark, blue: -
tall, dark, blue: -
tall, blond, blue: +
tall, dark, brown: -
short, blond, brown: -

* Taken from “Learning efficient classification procedures and their application to chess end games” by R. Quinlan in Machine Learning: An Artificial Intelligence Approach

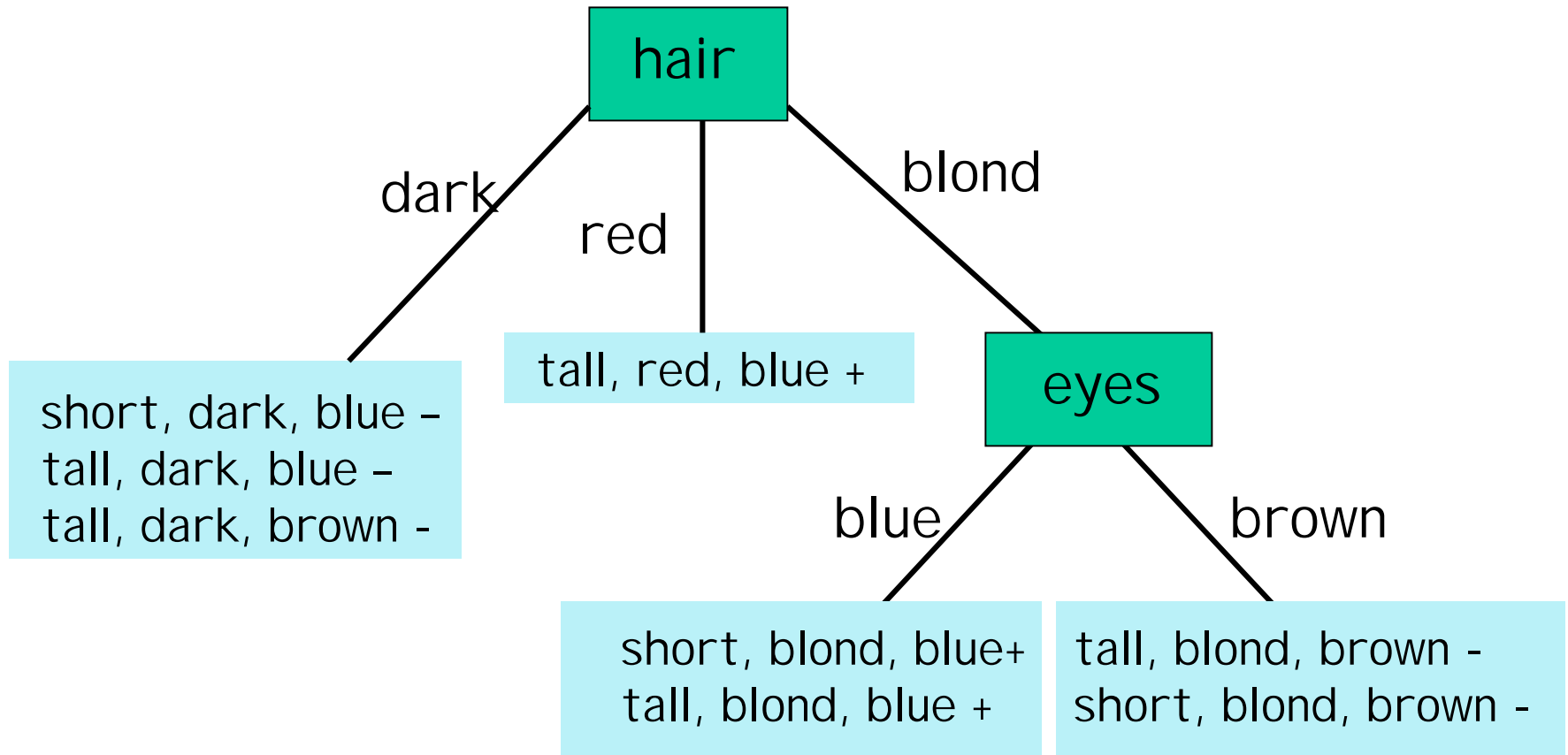
Example (Cont'd)

Suppose we select hair as the first attribute A. Then we get:



Since the rightmost branch is not all one class we need to continue...

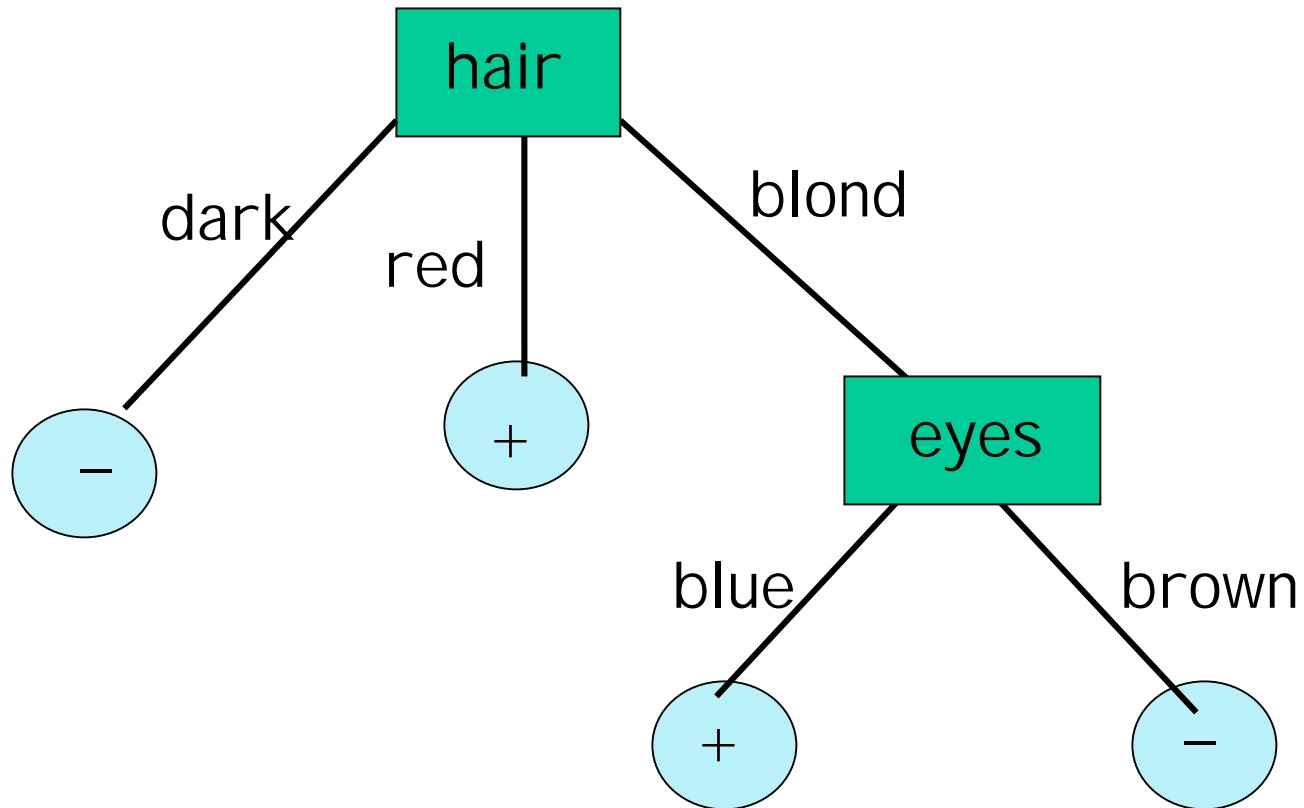
Example (Cont'd)



We are done because all leaves have examples of one class.

Example (Cont'd)

We can even replace the training examples by class names.

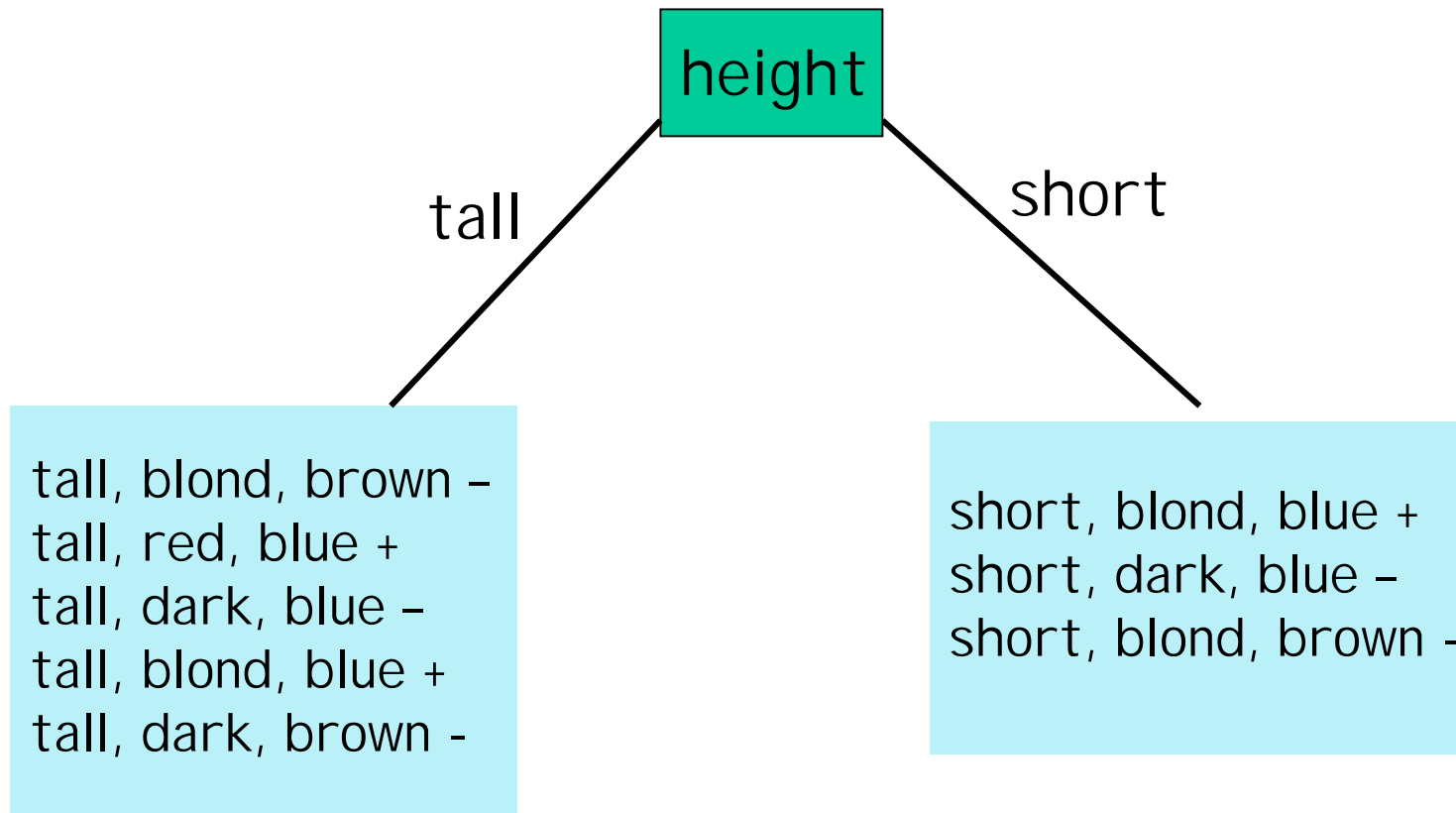


Example (Cont'd)

- This will work provided two examples with the identical attributes don't belong to different classes, which could happen if:
 - The data is noisy, or
 - The attributes are inadequate

Example (Cont'd)

Here's the tree if height is the root attribute A. Note that even more splitting would have to be done than if hair is the root.



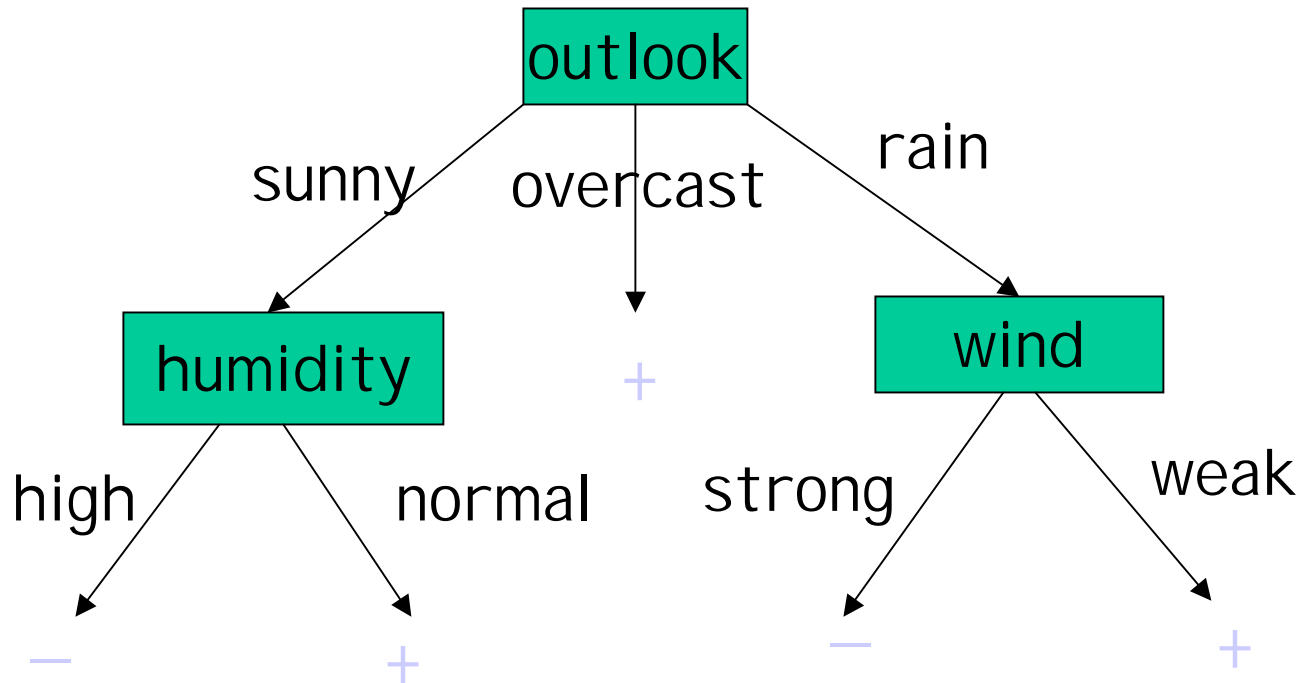
Information theory for choice of An (Example Cont'd)

- **How do we make a good choice of the root attribute, i.e., the first one to be tested in the tree?** Let's see...
- Since there are 3 positive and 5 negative examples in S, $\text{Entropy}(S) = -3/8 \log_2 3/8 - 5/8 \log_2 5/8 = 0.954$
- If we test height first, then the information still needed for a rule for the “tall” branch is:
 - $\text{Entropy}(S_{\text{tall}}) = -2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.971$
- And the information still needed for a rule for the “short” branch is:
 - $\text{Entropy}(S_{\text{short}}) = -1/3 \log_2 1/3 - 2/3 \log_2 2/3 = 0.918$
- Therefore, $E(\text{height}) = (5/8)\text{Entropy}(S_{\text{tall}}) + (3/8)\text{Entropy}(S_{\text{short}}) = 0.951$
- The information gained by testing “height” first is:
 $\text{Gain}(S, \text{height}) = \text{Entropy}(S) - E(\text{height}) = 0.954 - 0.951 = 0.003$, which is negligible.

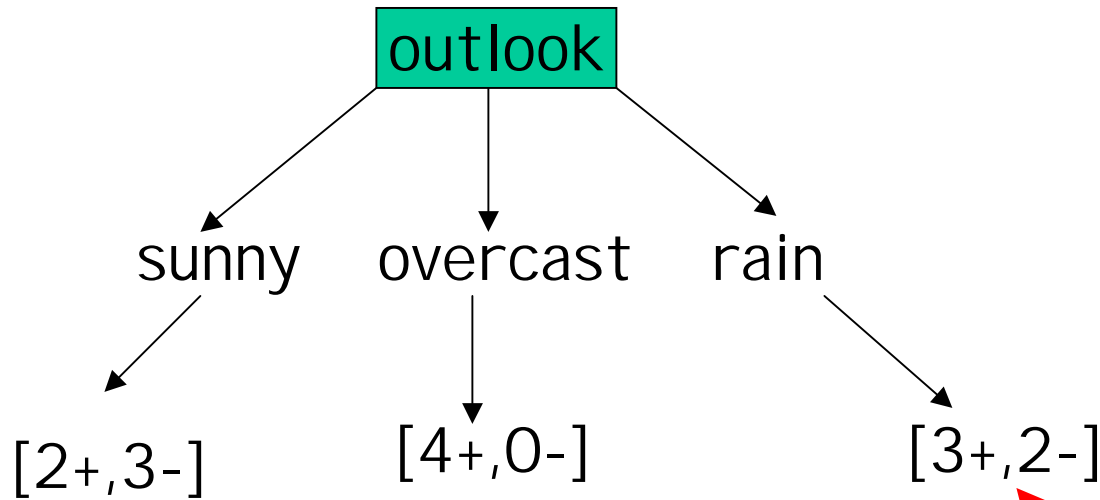
Example (Cont'd)

- We can perform a similar computation for testing “hair” as the first (root) attribute:
 - $E(\text{hair}) = 3/8 * 0 + 1/8 * 0 + 4/8 * 1 = 0.500$
 - $\text{Gain}(S, \text{hair}) = \text{Entropy}(S) - E(\text{hair}) = 0.954 - 0.5 = 0.454$
- We can do the same for testing “eyes” as the root:
 - $\text{Gain}(S, \text{eyes}) = \text{Entropy}(S) - E(\text{eyes}) = 0.347$
- The principle of maximizing expected information gain would lead ID3 to select “hair” as the attribute to form the root of the decision tree.

Let's continue our earlier example...



Example (Cont'd)



Done with this leaf

The gain of attribute humidity is the highest among all attributes, so a split on humidity will occur here.

The gain of attribute wind is the highest among all attributes, so a split on wind will occur here.

Entropy calculation when there are more than two classes

- **Entropy(S) = $-\sum_{i=1}^n (c_i/e) \log_2 (c_i/e)$**
 - n = the total number of classes
 - c_i = the number of examples in S that are class c_i
 - e = the total number of examples in S
 - c_i/e = the fraction of examples in S that are class c_i

When is learning decision trees appropriate?

- Instances described by attribute-value pairs.
- Target function is discrete-valued (although continuous-valued can also be handled).
- Disjunctive hypotheses may be required.
- Possibly noisy/incomplete training data.
- Used in a number of real-world applications: credit risk analysis, medical diagnosis, etc.

Properties of ID3

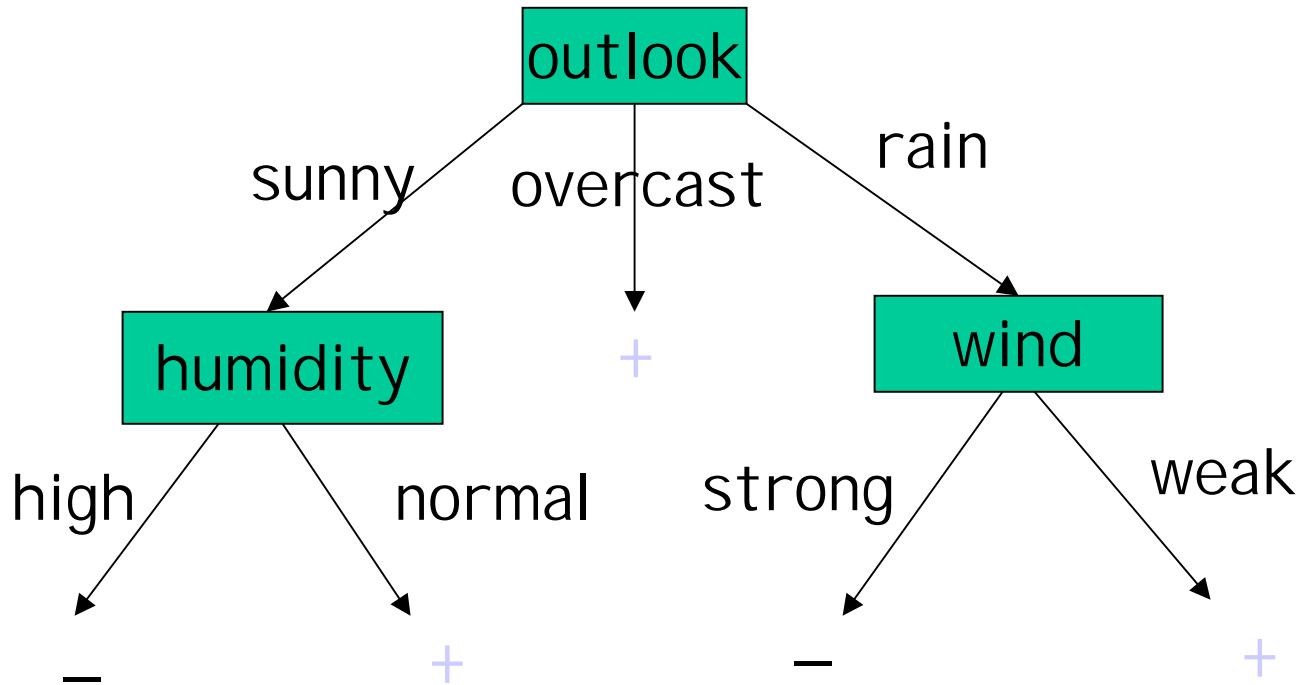
- The hypothesis space is complete: all possible discrete-valued functions on the given attributes are in the space.
- It outputs a single hypothesis.
- It is a greedy algorithm and can get stuck in local minima.
- It tends to be robust to noisy data.
- ID3's inductive bias is a shortest-tree preference, which is hard to formalize.

Evaluating a decision tree

- **Evaluation:** We test the learned tree on a test set of examples. We determine its prediction accuracy.
- A tree that *overfits* the training data typically predicts poorly on the test set. Furthermore, overfitting is bad when the examples are noisy.

ID3 and overfitting the training data

- Consider adding the following noisy example:
 - Sunny, hot, normal, strong, Play-Tennis = –



A couple of approaches to avoiding overfitting

- Early stopping: stop growing tree when data splits not statistically significant.
 - Apply a statistical test to determine whether splitting a node improves test set accuracy.
- Post pruning: Grow the full tree, then systematically prune nodes using a test set.

Version Space Learning

- Learns a precise concept from examples
- The *boundary* determines the space of acceptable concepts
- Each example can modify the boundary
- Algorithm succeeds when boundary converges

Observations About Learning

- Selecting the right representation crucial
 - Inductive bias heavily influences final result
- Learning is essentially the problem of searching the hypothesis space for an accurate and generalizable function
- Many learning algorithms are expressible as search algorithms
 - Just like seemingly everything else in AI...