Incremental Learning and Concept Drift: Overview

- Incremental learning
- The algorithm ID5R
- Taxonomy of incremental learning
- Concept Drift

Motivation

- “Batch”-Scenario: all examples are available at once
  - single run over all example
  - just one hypothesis
- “Stream”-Scenario: always new examples arrive
  - series of hypothesis
  - not a good idea to collect all data and build hypothesis from scratch
  - better: reuse already built hypothesis and modify it

Another point: In some situations, accuracy is not always the matter of concern

- sometimes, one wants to have a quick estimation / raw approximation very quickly
- one wants to keep track of the process of hypothesis creation

Desirable properties

- average cost of updating tree should be much lower than average cost of building new tree from scratch
- update cost independent of number of training examples
- tree should depend only on the set of examples, not their order

The algorithm ID5R

Developed by Utgoff

Basic ideas:

- keep counts on each node (that allow to compute entropies)
- when a new example arrives and the tree needs to be changed:
  - re-structure the tree
  - “pull up” instead of simply deleting subtrees
- leafs maintain lists of examples seen

Assumption: binary classification problem

Decision Trees created by ID5R

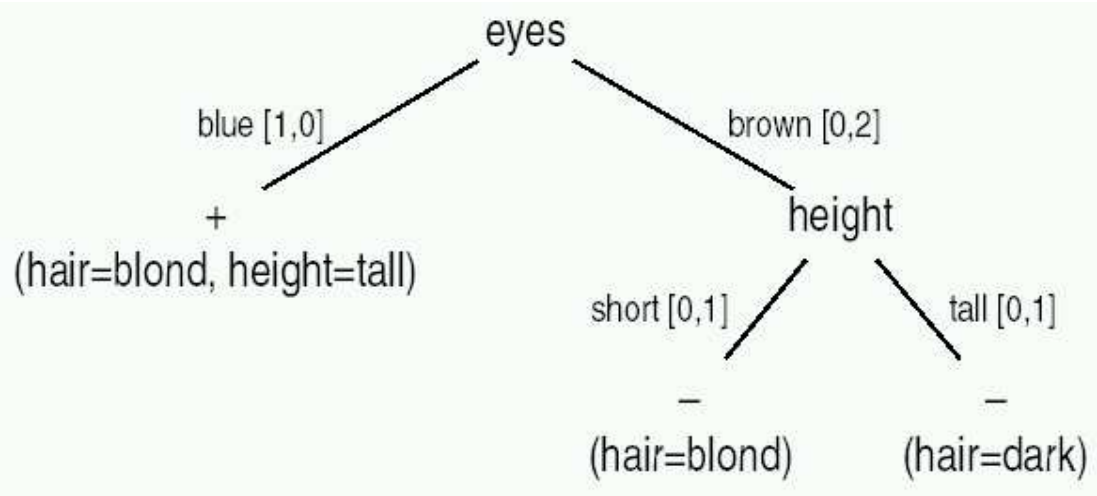
Extend definition of a tree:

Leaf nodes: also contain a set of instance descriptions

non-leaf nodes (decision nodes): also contain set of potential candidate tests, each with positive and negative counts for each possible outcome

branches: also contain positive and negative counts for this outcome

Example:



classification is done as usual

ID5R: The algorithm

- start with empty tree n

ID5R(n,*instance*):

1. If n is empty:
   - create leaf
   - set leaf class to class of *instance*
   - set of instances: singleton set consisting of *instance*
2. If n is a leaf and *instance* is of the same class:
   - add *instance* to the set of instances
3. Otherwise:
   (a) If n is a leaf: expand it one level (choose test arbitrary)
   (b) for all candidate tests at node n: update counts
   (c) if the current test is not the best one among all candidate tests:
      i. Restructure tree n so that the best test is now at root ("Pull-Up")
      ii. Recursively reestablish best test in each subtree (ignore path of instance)
   (d) Recursively update tree t along the path *instance* goes. Grow the branch if necessary.

ID5R: An example

class	height	hair	eyes
–	short	blond	brown
–	tall	dark	brown
+	tall	blond	blue
–	tall	dark	blue
–	short	dark	blue
+	tall	red	blue
–	tall	blond	brown
+	short	blond	blue

First example:
(-, height=short, hair=blond, eyes=brown)
Hypothesis:

–
(eyes=brown, hair=blond, height=short)

Second example:
(-, height=tall, hair=dark, eyes=brown)
Hypothesis:

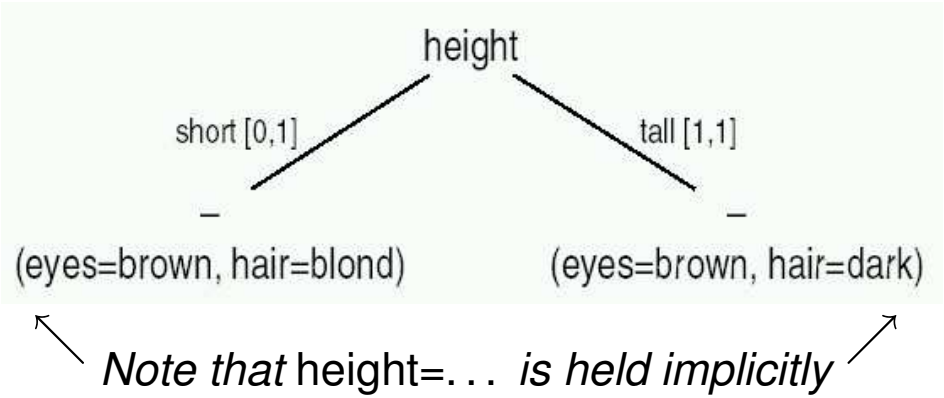
–
(eyes=brown, hair=dark, height=tall)
(eyes=brown, hair=blond, height=short)

ID5R: An example (2)

Third example: *(+, height=tall, hair=blond, eyes=blue)*

(a) Expand tree one level, choosing arbitrary test

(b) Update counts



(c) if the current test is not the best one among all candidate tests:

i. Restructure tree n so that the best test is now at root (“Pull-Up”)

ii. Recursively reestablish best test in each subtree

What means *best* here?

How to restructure trees?

ID5R's measure

ID3 uses measure information gain:

$$\mathit{Gain}(S, A) \equiv \mathit{Entropy}(S) - \sum_{v \in \mathit{values}(A)} \frac{|S_v|}{|S|} \mathit{Entropy}(S_v)$$

$$\mathit{Entropy}(S) \equiv -\frac{|S_\oplus|}{|S|} \log_2 \frac{|S_\oplus|}{|S|} - \frac{|S_\ominus|}{|S|} \log_2 \frac{|S_\ominus|}{|S|}$$

S: trainings set, A: test

$S_\oplus$ ($S_\ominus$): set of all positive (negative) examples in S

S_v subset of all examples in S for which test A has value v

ID5R uses the same measure (but omits to compute $\mathit{Entropy}(S)$ since it is the same for all tests in a node).

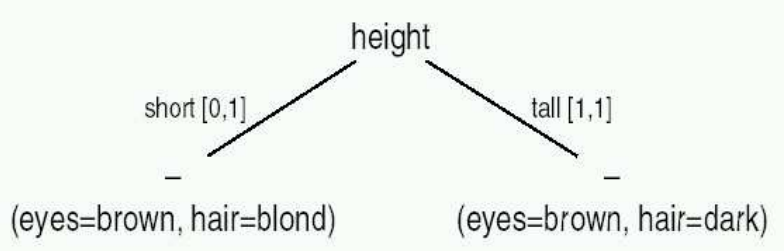
How to restructure trees?

Pull-Up a test A

1. If the test A is already at the root, do nothing
2. Otherwise:
   (a) recursively pull up A to the root of each immediate subtree
   (if necessary, expand leafes)
   (b) transpose the tree so that A is now in the root and the old root attribute is now in the roots of all immediate subtrees.

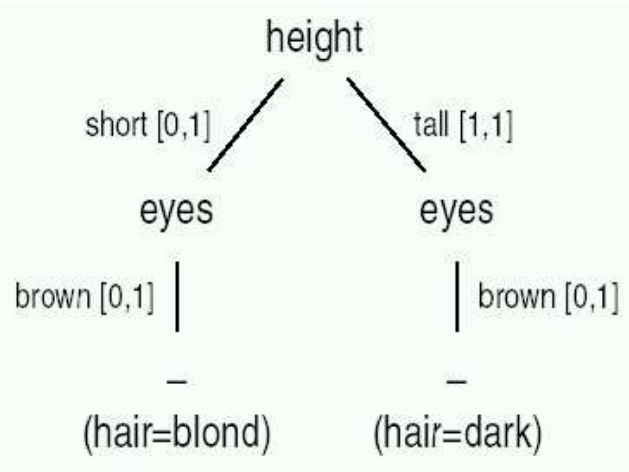
ID5R: An example (3)

Back to our example, the situation is:

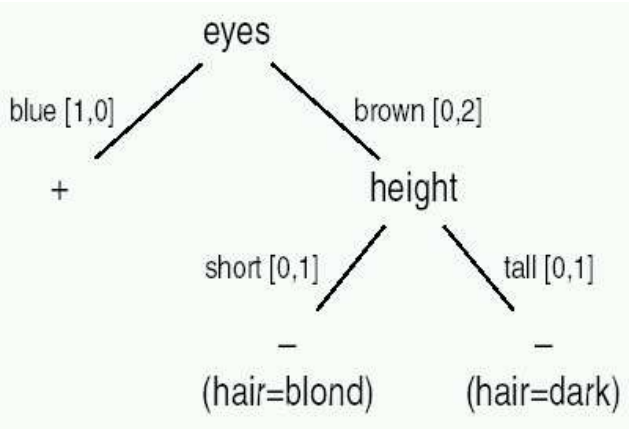


The best candidate test (due to Gain) is *eyes*. $\longrightarrow$ Pull up *eyes*.

First: recursively pull up A to the root of each immediate subtree (if necessary, expand leafes)



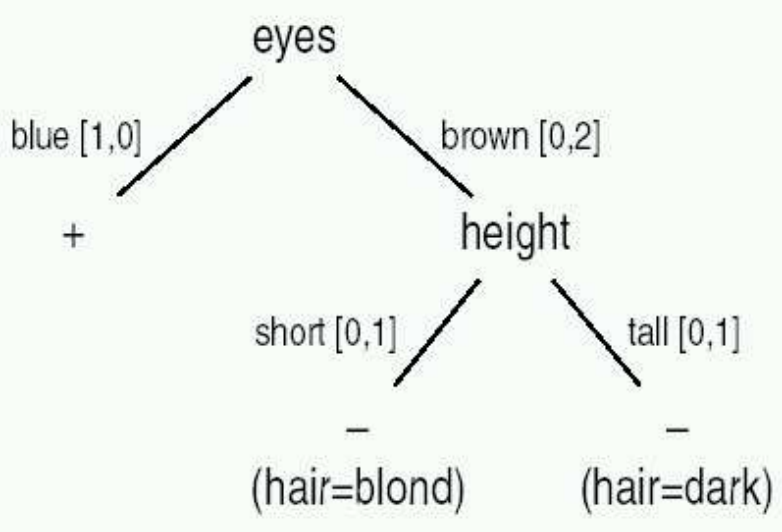
Then, transpose the tree:



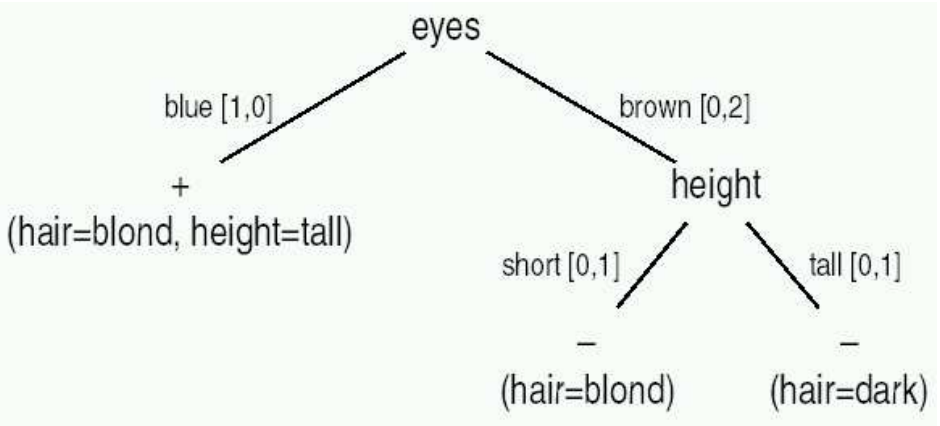
ID5R: An example (4)

Next, recursively reestablish best test in each subtree.

Situation before:



Currently, *height* is test in the subtree just by chance
$\longrightarrow$ we have to descend into that subtree.
Just by chance, *height* is the best test in the right subtree, so we get



Note: The right subtree could be collapsed, but ID5R does not!

Why? it is possible that the tree may be expanded again.
Shown by experiments that the efford would not pay off.

ID5R: An example (5)

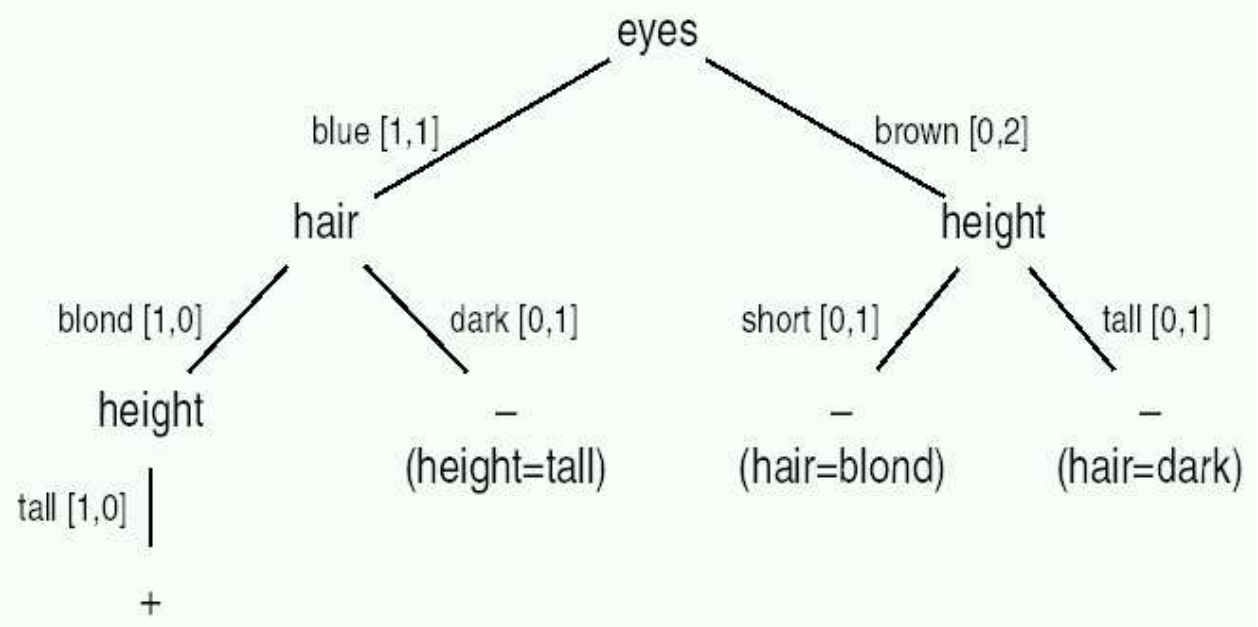
4th example: *(-, height=tall, hair=dark, eyes=blue)*

eyes is still best attribute in root.

$\longrightarrow$ expand left node, e.g. by test *height*.

$\longrightarrow$ best test here is *hair*.

$\longrightarrow$ restructure left subtree, this gives



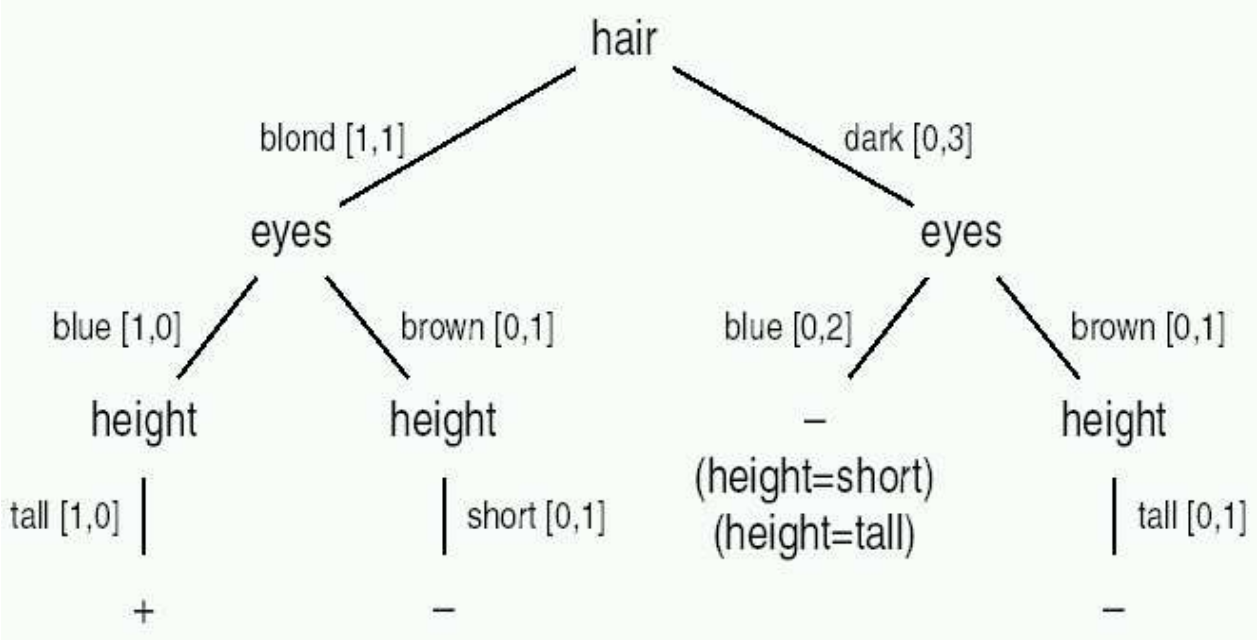
ID5R: An example (6)

5th example: *(-, height=short, hair=dark, eyes=blue)*

best test in root is now *hair*

$\longrightarrow$ need to pull up *hair* and reestablish best tests in the subtrees

result:



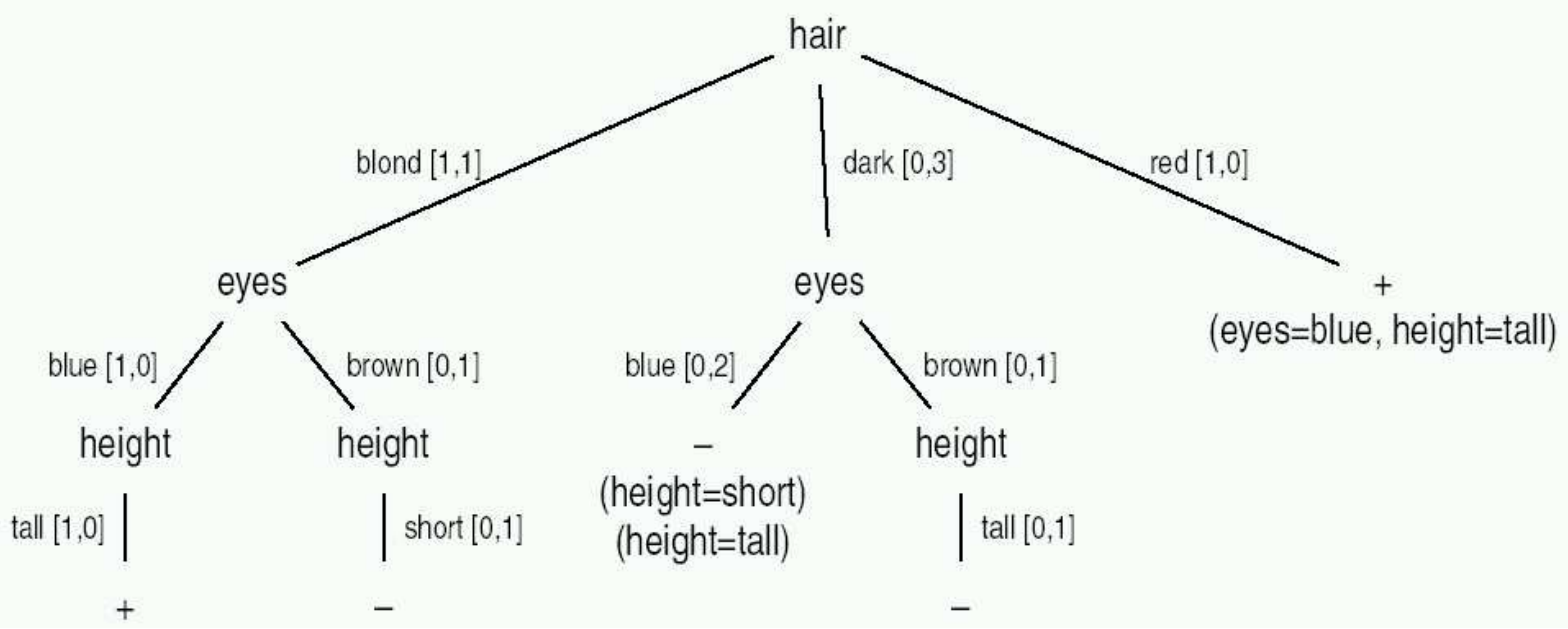
ID5R: An example (7)

6th example: *(+, height=tall, hair=red, eyes=blue)*

best test in root is now still *hair*, but:

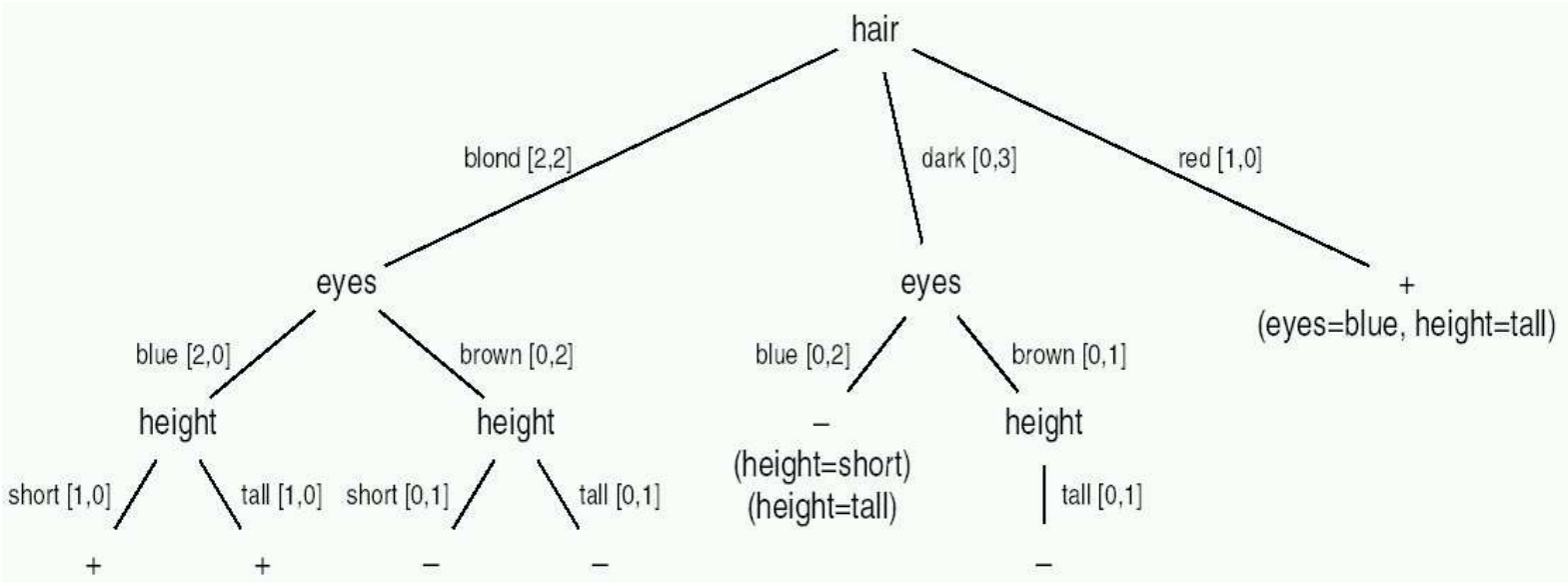
new value: *red*

$\longrightarrow$ grow new branch, result:

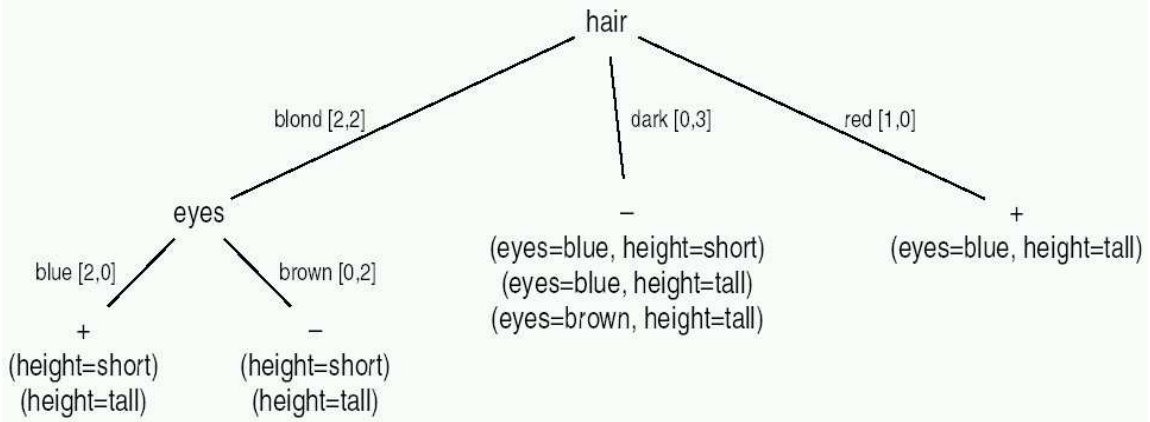


ID5R: An example (finished)

the remaining examples finally lead to



This is equivalent to



Properties of ID5R

- computes the same hypothesis as ID3!

Complexities:

d... number of attributes
b... maximum number of possible values for tests
n... number of training instances

	ID3	ID3*	ID5R
instance-count additions:	$\mathcal{O}(n \cdot d^2)$	$\mathcal{O}(n^2 \cdot d^2)$	$\mathcal{O}(n \cdot d \cdot b^d)$
how often compute entropy?	$\mathcal{O}(b^d)$	$\mathcal{O}(n \cdot b^d)$	$\mathcal{O}(n \cdot b^d)$

ID3: only *one* hypothesis is constructed!

ID3*: for each new example, a new tree is computed.

Taxonomy of incremental algorithms

full example memory Store *all* examples

- allows for efficient restructuring
- good accuracy
- huge storage needed

Examples: ID5, ID5R, IDL, ITI, BOAT

no example memory Only store statistical information in the nodes

- loss of accuracy (depending on the information stored or again huge storage needed)
- relatively low storage space

Examples: ID4, MSC, G, TG

partial example memory Only store *selected* examples

- trade of between storage space and accuracy

Examples: HILLARY, FLORA, MetaL, AQ-PM, INC-DT

Partial Example Memory Learning

Problem: How to select examples?

Obvious approach: only keep the last n examples $\longrightarrow$ Windowing

- How large shall n be?
  - First Variant: Let n be fixed.
  - Second Variant: Adapt n depending on data situation.

Other ideas, which examples to be stored:

- store only examples that have been classified incorrectly
- store only examples which are believed to be important:
  - AQ-PM: store examples that are close to the seperating hyperplanes
  - INC-DT: store examples where the classifier believes it has a weak competence

Concept Drift

- Usual assumption in ML: examples are drawn i.i.d. (identically and independently distributed) w.r.t. to a fixed distribution
- In real world situations, target concept as well as data distribution change over time
  - example: Spam Filtering

- Concept Drift slow changes
- Concept Shift rapid changes

Approaches to cope with concept drift/shift

- Aging: weight data according to their age (and/or utility for the classification task)
- Windowing
  - fixed size
  - adaptive size
    - in phases without or with little concept drift, enlarge window
    - in phases with concept drift, decrease window size (drop oldest examples)

How can we detect concept drift/shift?

- Performance measures
  - e.g. accuracy
- properties of the classification model
  - e.g. complexity of hypothesis
- properties of the data
  - e.g. class distribution, attribute value distribution, current top attributes