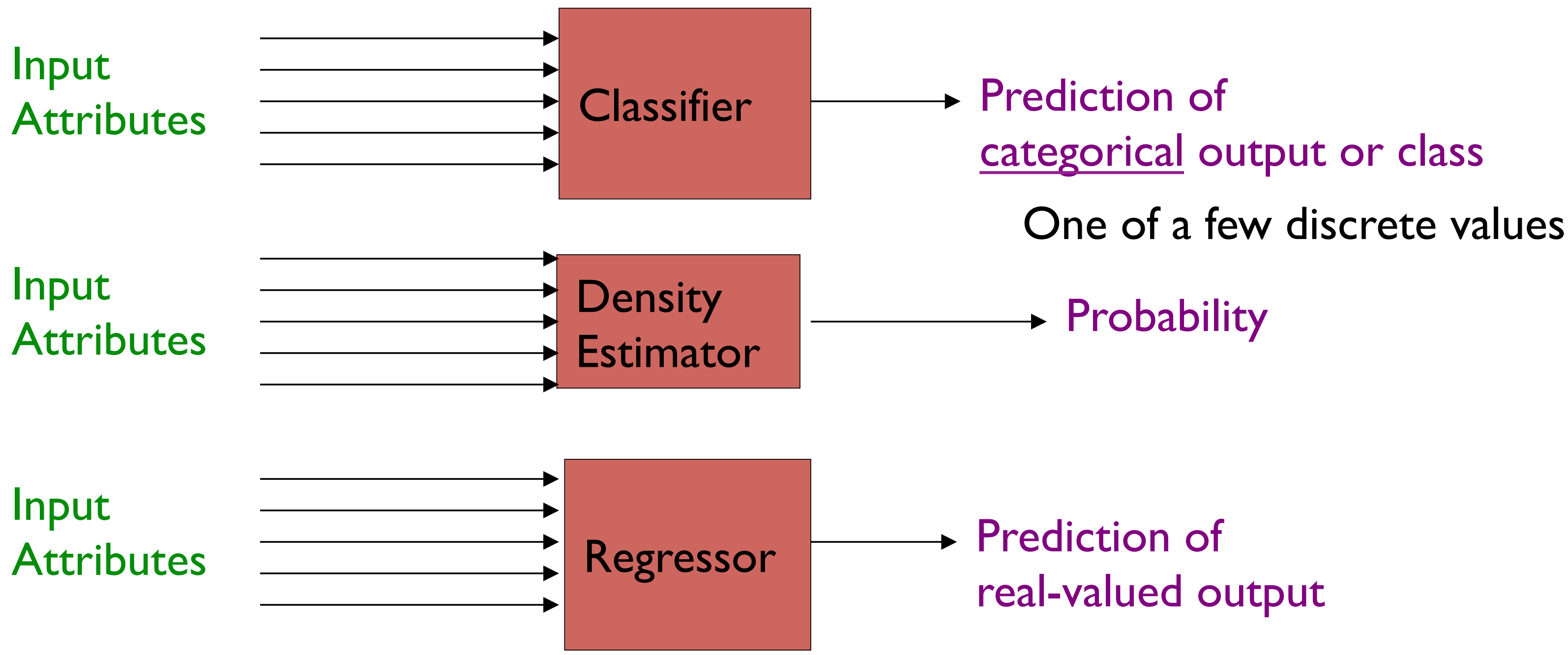
Classification with Decision Trees and Rules

Density Estimation – looking ahead

- Compare it against the two other major kinds of models:



DECISION TREE LEARNING: OVERVIEW

Decision tree learning

Google scholar decision tree quinlan Search

Scholar Articles and patents | anytime | include citations | Create email alert

Induction of **decision** trees

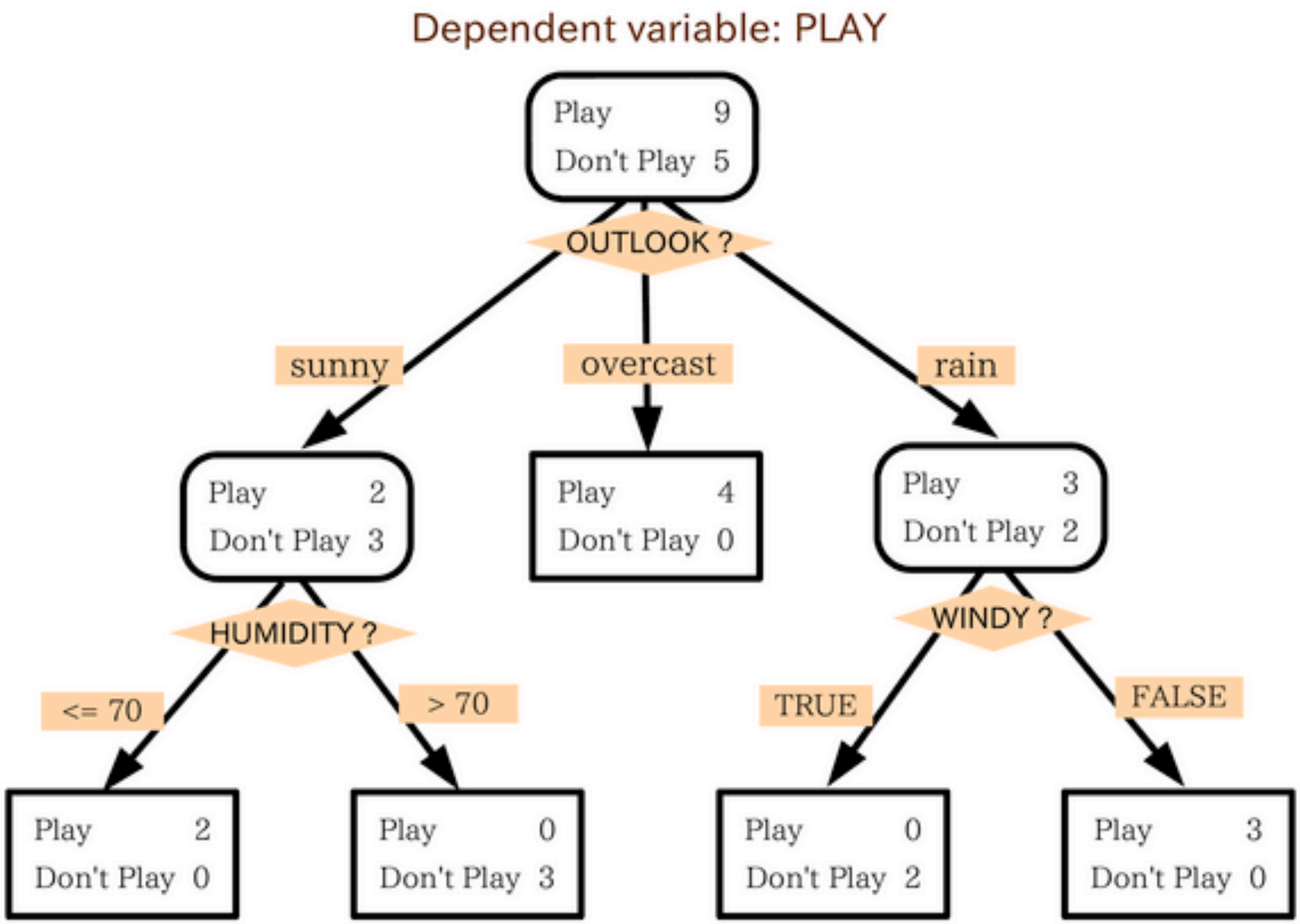
JR **Quinlan** - Machine learning, 1986 - Springer

... Several studies have been carried out to see how this modified procedure holds up under varying levels of noise (**Quinlan** 1983b, 1985a ... each experiment, the whole set of objects was artificially corrupted as described below and used as a training set to produce a **decision tree**. ...

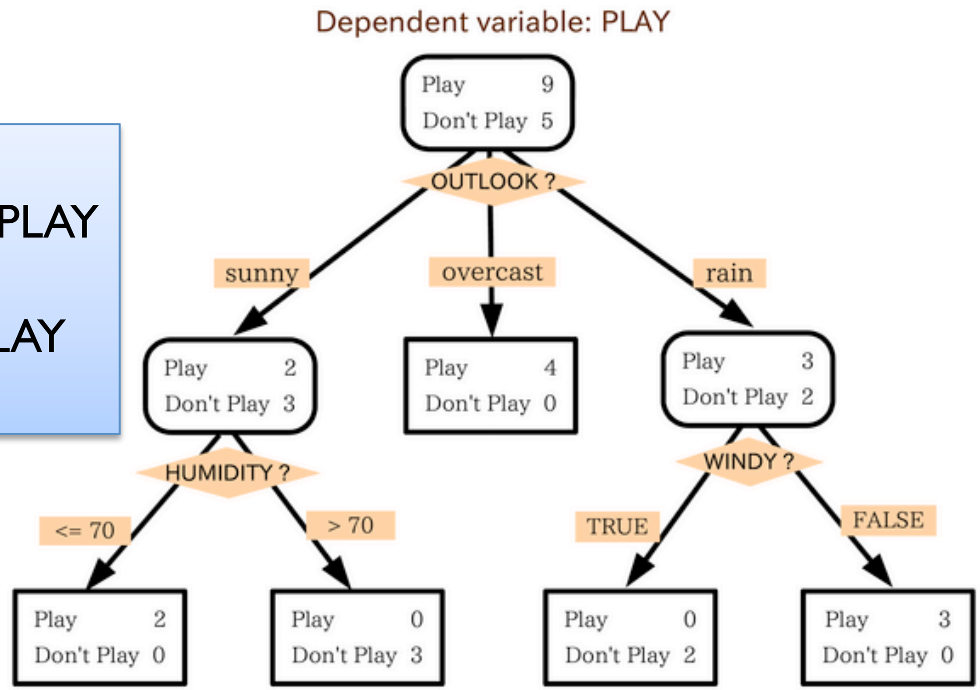
Cited by 10584 - Related articles - Library Search - All 75 versions

Cited by 13602

A decision tree



Another format: a set of rules



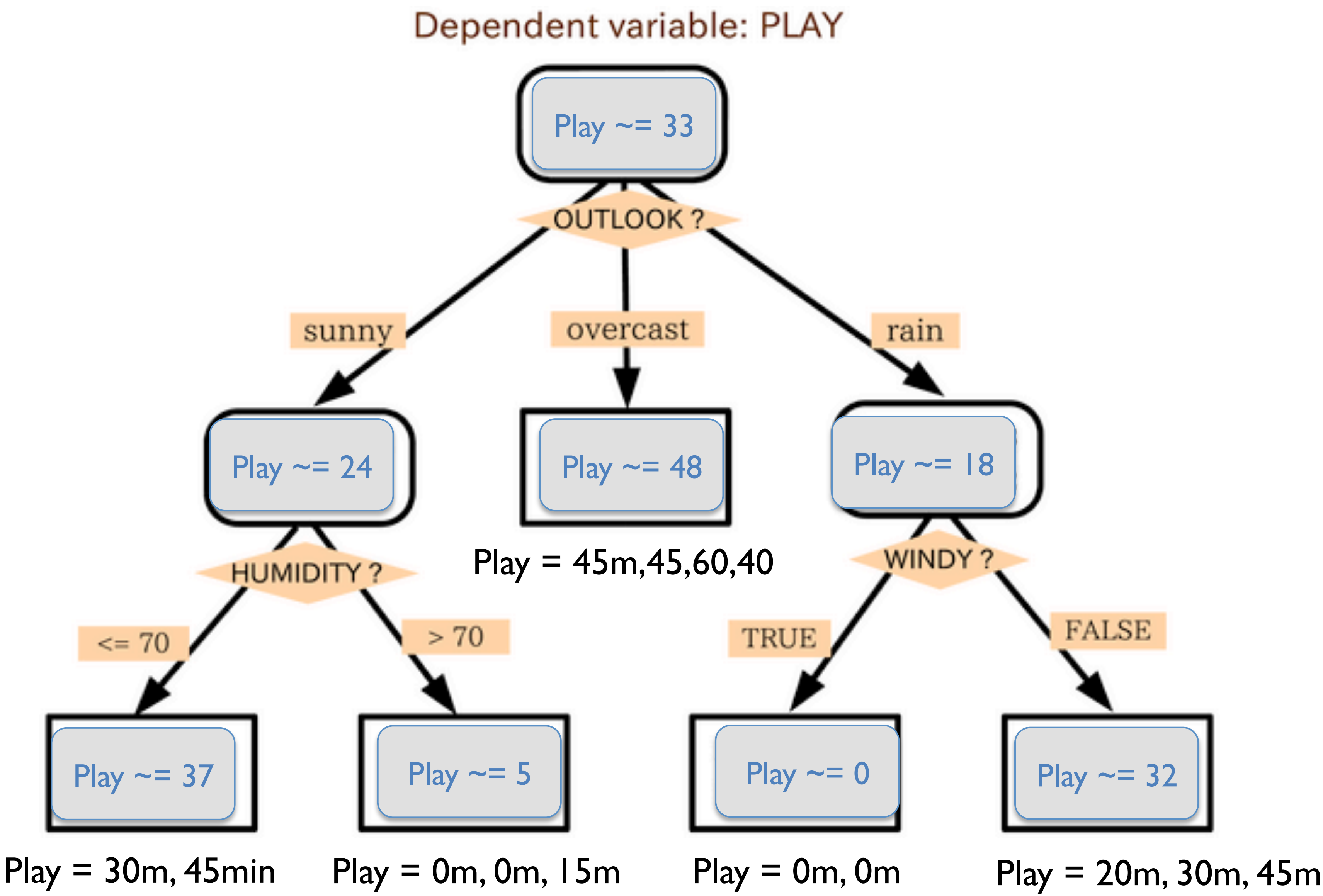
if O=sunny and H<= 70 then PLAY
else if O=sunny and H>70 then DON'T_PLAY
else if O=overcast then PLAY
else if O=rain and windy then DON'T_PLAY
else if O=rain and !windy then PLAY

One rule per leaf in the tree

Simpler rule set

if O=sunny and H> 70 then DON'T_PLAY
else if O=rain and windy then DON'T_PLAY
else PLAY

A regression tree



Motivations for trees and rules

- Often you can find a fairly accurate classifier which is **small and easy to understand**.
  - Sometimes this gives you useful **insight** into a problem, or helps you debug a feature set.
- Sometimes features **interact** in complicated ways
  - Trees can find interactions (e.g., "sunny and humid"). Again, sometimes this gives you some insight into the problem.
- Trees are very **inexpensive at test time**
  - You don't always **even need to compute all the features** of an example.
  - You can even build classifiers that take this into account….
  - Sometimes that's important (e.g., "bloodPressure<100" vs "MRIScan=normal" might have different costs to compute).

An example: “Is it the Onion”?

literallyunbelievable.org

Dataset: 200 Onion articles, ~500 Economist articles.

Accuracies: almost 100% with Naïve Bayes!

I used a rule-learning method called RIPPER

Fast Effective Rule Induction

Authors William W Cohen

Publication date 1995

Journal Proceedings of the Twelfth International Conference on Machine Learning, Lake Tahoe, California

Total citations Cited by 3040

Scholar articles Fast E ective Rule Induction *
WW Cohen - Proceedings of the Twelfth International Conference on ..., 1995
Cited by 3040 - Related articles - All 44 versions

```
bash-3.2$ ripper onion
Final hypothesis is:
fromOnion :- wordsInArticle ~ enlarge (173/0).
fromOnion :- wordsInArticle ~ monday (9/1).
fromOnion :- wordsInArticle ~ added, wordsInArticle ~ play (6/1).
fromOnion :- wordsInArticle ~ mocking (2/0).
fromOnion :- wordsInArticle ~ manhattan (2/1).
default fromEconomist (530/0).
============================== summary ==================================
Train error rate:   0.41% +/- 0.24% (725 datapoints)    <<
Hypothesis size:    5 rules, 11 conditions
Learning time:      1.09 sec
```

Translation:

if "enlarge" is in the set-valued attribute wordsArticle
then class = fromOnion. *this rule is correct 173 times, and never wrong*
...
if "added" is in the set-valued attribute wordsArticle
and "play" is in the set-valued attribute wordsArticle
then class = fromOnion. *this rule is correct 6 times, and wrong once*
...

```
bash-3.2$ grep -B2 -A2 -wi enlarge onion/*.txt | head -20
onion/onion.1.txt-   conference. "Many of them had been given nothing more than a pair of
onion/onion.1.txt-   tube socks or men's briefs to wear."
onion/onion.1.txt:   [4]Enlarge Image American Apparel
onion/onion.1.txt-
onion/onion.1.txt-   Law enforcement officials continued clearing models from the compound
--
onion/onion.10.txt-   reportedly grown too complacent to conduct its suicide mission, an
onion/onion.10.txt-   attack on the San Onofre Nuclear Generating Station.
onion/onion.10.txt:   [4]Enlarge Image After Five Years
onion/onion.10.txt-
onion/onion.10.txt-   Three of the six terrorists spend an afternoon together watching an
--
onion/onion.100.txt-   today."
onion/onion.100.txt-
onion/onion.100.txt:   [4]Enlarge Image Lieberman Pledges To Gloss Over The Boring Issues
onion/onion.100.txt-
onion/onion.100.txt-   Lieberman tells Hartford voters he'll be brief.
```

After cleaning ‘Enlarge Image’ lines

```
bash-3.2$ ripper clean-onion
Final hypothesis is:
fromOnion :- wordsInArticle ~ i, wordsInArticle ~ added, wordsInArticle ~ my (82/0).
fromOnion :- wordsInArticle ~ i, wordsInArticle ~ monday (42/0).
fromOnion :- wordsInArticle ~ said, wordsInArticle ~ tuesday (22/0).
fromOnion :- wordsInArticle ~ added, wordsInArticle ~ re (13/4).
fromOnion :- wordsInArticle ~ said, wordsInArticle ~ my, wordsInArticle ~ really (8/2).
fromOnion :- wordsInArticle ~ monday (5/1).
fromOnion :- wordsInArticle ~ ll, wordsInArticle ~ me (5/0).
fromOnion :- wordsInArticle ~ u, wordsInArticle ~ political (4/0).
fromOnion :- wordsInArticle ~ added, wordsInArticle ~ truly (3/0).
default fromEconomist (526/8).
=============================== summary ===============================
Train error rate:   2.07% +/- 0.53% (725 datapoints)    <<
Hypothesis size:    9 rules, 28 conditions
Learning time:      3.98 sec
```

Also, estimated test error rate increases from 1.4% to 6%

Different Subcategories of Economist Articles

```
Final hypothesis is:
aboutInternational :- wordsInArticle ~ countries, wordsInArticle ~ nations (9/4).
aboutInternational :- wordsInArticle ~ soil (7/1).
aboutInternational :- wordsInArticle ~ based, wordsInArticle ~ authorities (6/4).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ barack (19/0).
aboutNorthAmerica :- wordsInArticle ~ barack (9/3).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ americans (13/1).
aboutNorthAmerica :- wordsInArticle ~ texas (7/2).
aboutNorthAmerica :- wordsInArticle ~ pricing (4/2).
aboutNorthAmerica :- wordsInArticle ~ huckabee (2/0).
aboutNorthAmerica :- wordsInArticle ~ miller (2/0).
aboutLatinAmerica :- wordsInArticle ~ n, wordsInArticle ~ president (23/2).
aboutLatinAmerica :- wordsInArticle ~ brazil (15/6).
aboutLatinAmerica :- wordsInArticle ~ latin (6/3).
aboutLatinAmerica :- wordsInArticle ~ fidel (8/0).
aboutLatinAmerica :- wordsInArticle ~ lvaro (3/0).
aboutLatinAmerica :- wordsInArticle ~ bolivia (5/0).
aboutLatinAmerica :- wordsInArticle ~ canadians (2/0).
aboutAfrica :- wordsInArticle ~ africa, wordsInArticle ~ president (32/4).
aboutAfrica :- wordsInArticle ~ al (15/7).
aboutAfrica :- wordsInArticle ~ lebanon (5/0).
aboutAfrica :- wordsInArticle ~ nigeria (3/1).
aboutAsia :- wordsInArticle ~ china (35/13).
aboutAsia :- wordsInArticle ~ india (12/3).
aboutAsia :- wordsInArticle ~ e09as761 (6/0).
aboutAsia :- wordsInArticle ~ interim (6/2).
aboutAsia :- wordsInArticle ~ park (4/2).
aboutBritain :- wordsInArticle ~ britain, wordsInArticle ~ british (34/7).
aboutBritain :- wordsInArticle ~ technology (11/2).
aboutBritain :- wordsInArticle ~ brown (17/3).
aboutBritain :- wordsInArticle ~ england (5/1).
aboutBritain :- wordsInArticle ~ craft (2/0).
default aboutEurope (91/42).
============================== summary ==============================
Train error rate:  21.58% +/- 1.78% (533 datapoints)    <<
Hypothesis size:   31 rules, 69 conditions
Learning time:     6.47 sec
```

```
aboutAfrica :- wordsInArticle ~ africa, wordsInArticle ~ president (32/4).
aboutAfrica :- wordsInArticle ~ al (15/7).
aboutAfrica :- wordsInArticle ~ lebanon (5/0).
aboutAfrica :- wordsInArticle ~ nigeria (3/1).
aboutAsia :- wordsInArticle ~ china (35/13).
aboutAsia :- wordsInArticle ~ india (12/3).
aboutAsia :- wordsInArticle ~ e09as761 (6/0).
aboutAsia :- wordsInArticle ~ interim (6/2).
aboutAsia :- wordsInArticle ~ park (4/2).
```

```
bash-3.2$ grep -wi e09as761 economist/asia/*.txt
economist/asia/asia.14.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.18.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.64.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.78.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.8.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.83.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
```

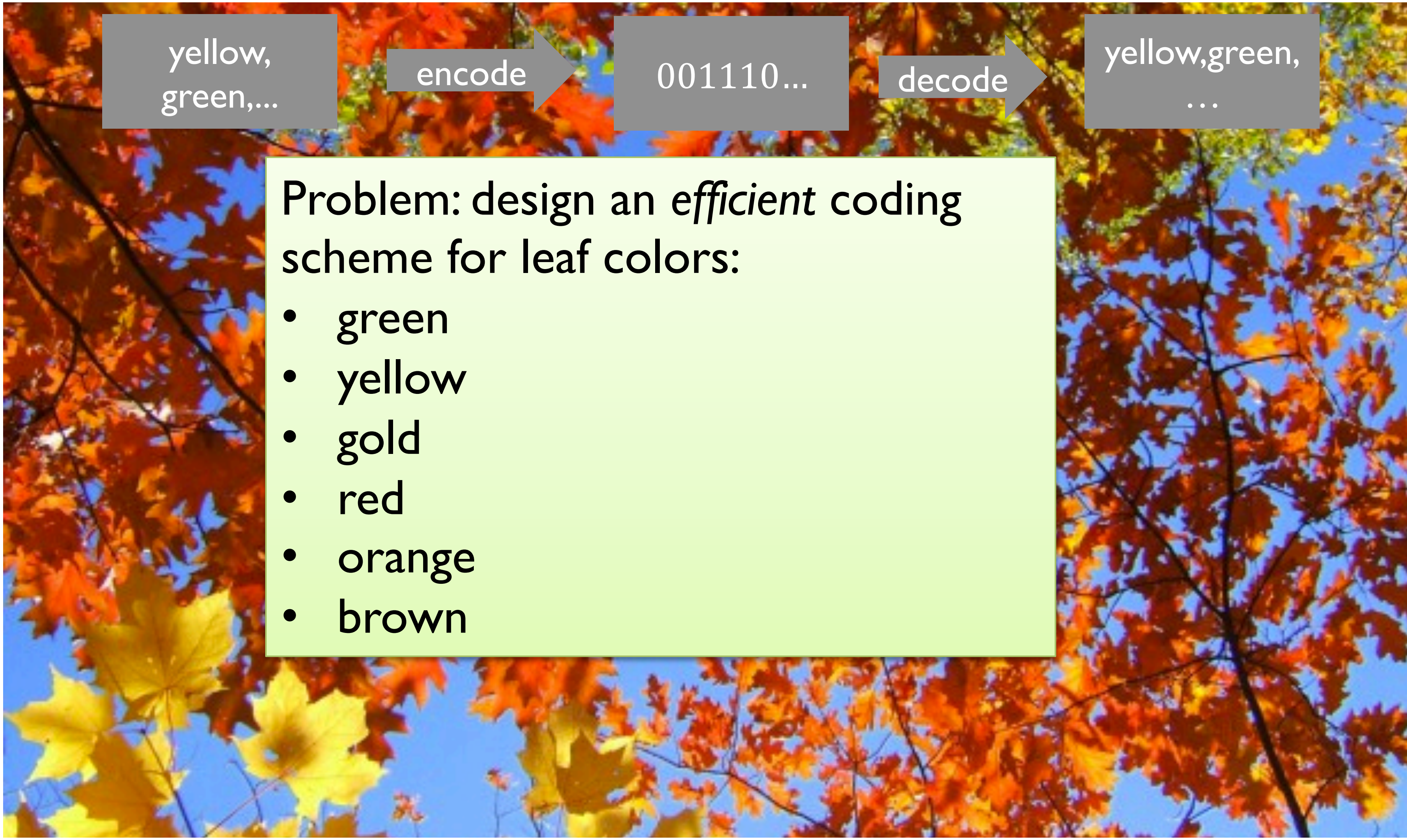
Motivations for trees and rules

- Often you can find a fairly accurate classifier which is **small and easy to understand.**
  - Sometimes this gives you useful **insight** into a problem, or helps you debug a feature set.
- Sometimes features interact in complicated ways
  - Trees can find interactions (e.g., "sunny and humid") that linear classifiers can't
  - Again, sometimes this gives you some insight into the problem.
- Trees are very inexpensive at test time
  - You don't always even need to compute all the features of an example.

Rest of the class: the algorithms. But first… decision tree learning algorithms are based on information gain heuristics.

BACKGROUND: ENTROPY AND OPTIMAL CODES

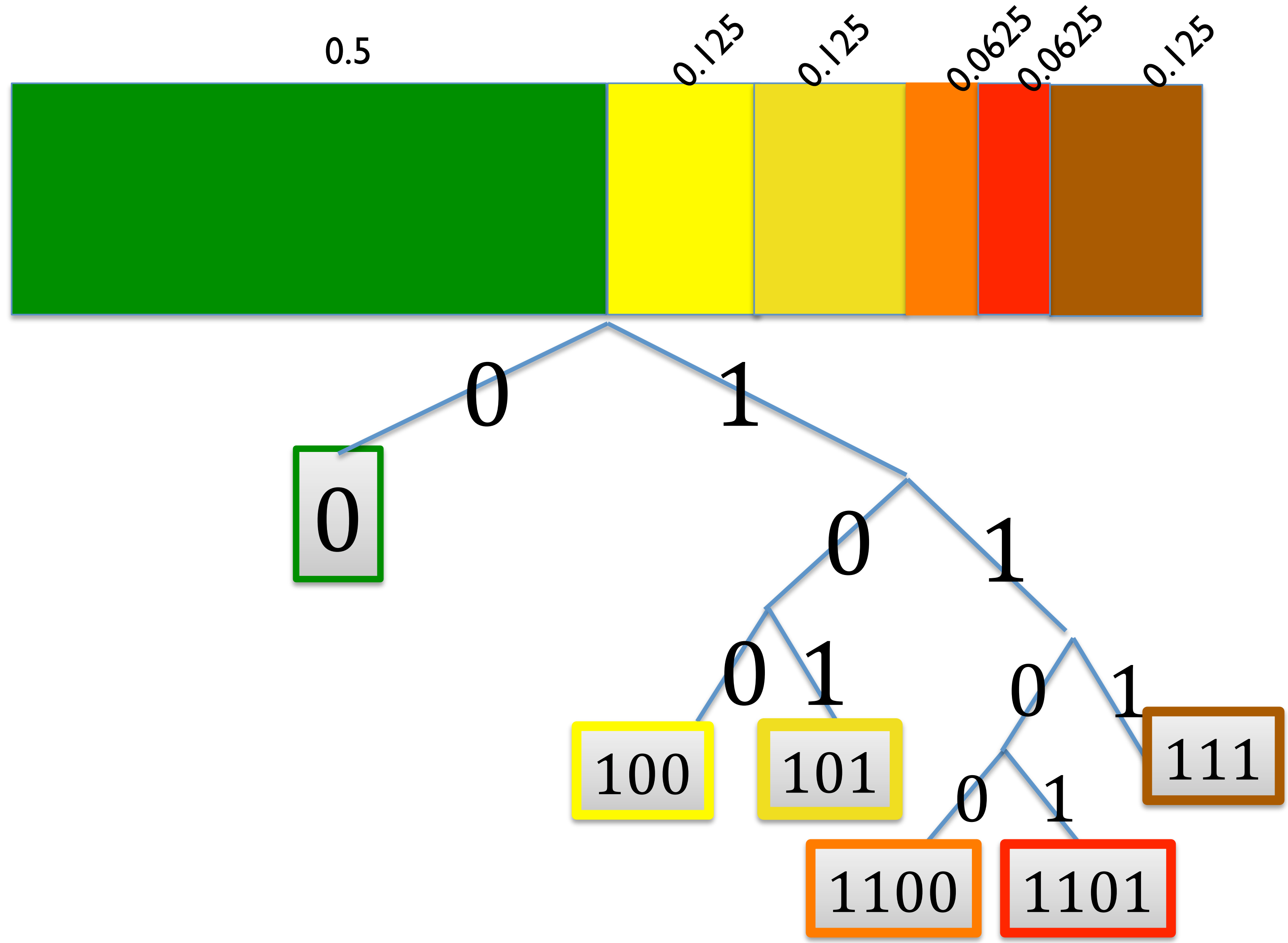
Information theory

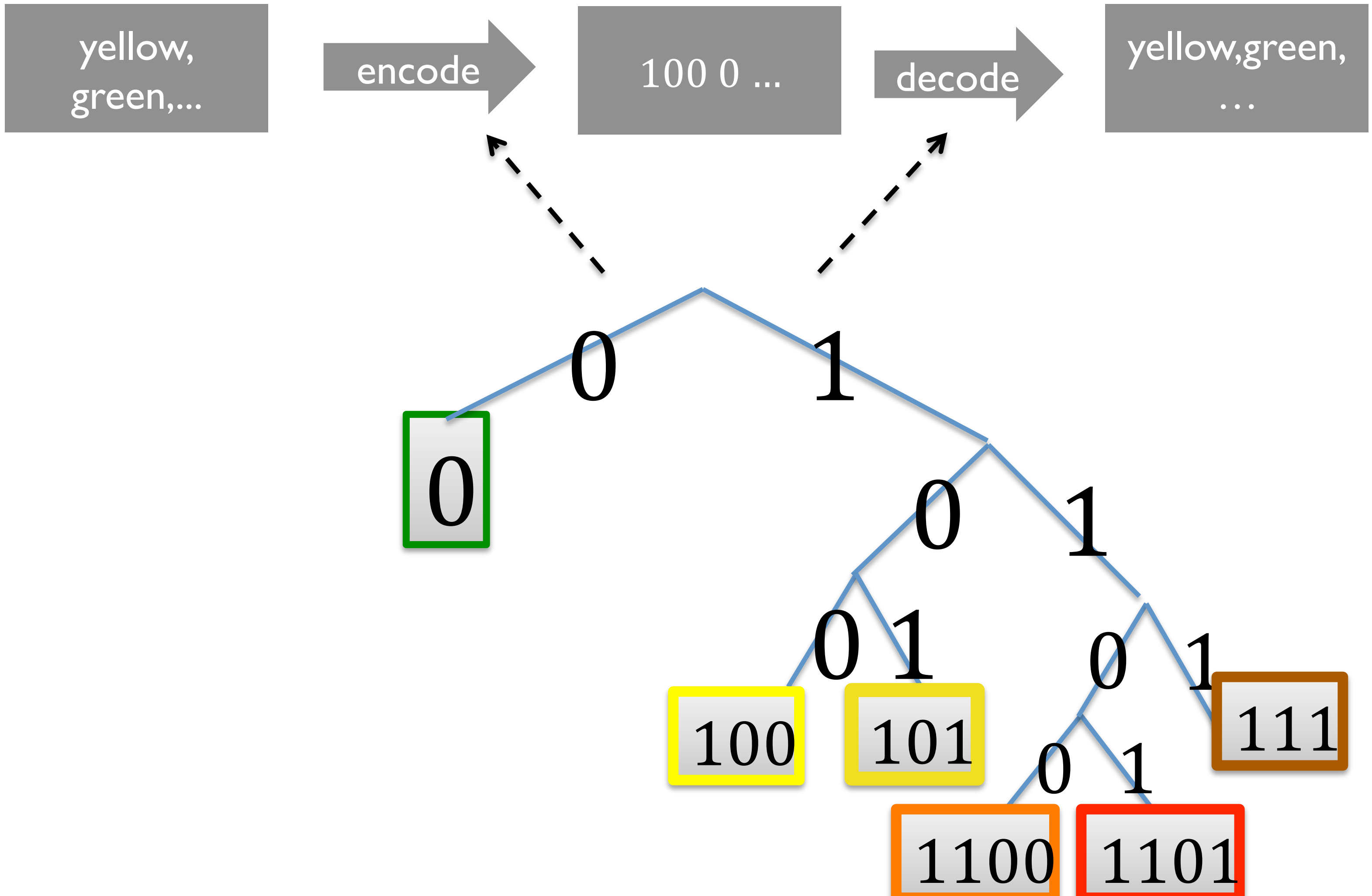


Problem: design an *efficient* coding scheme for leaf colors:

- green
- yellow
- gold
- red
- orange
- brown

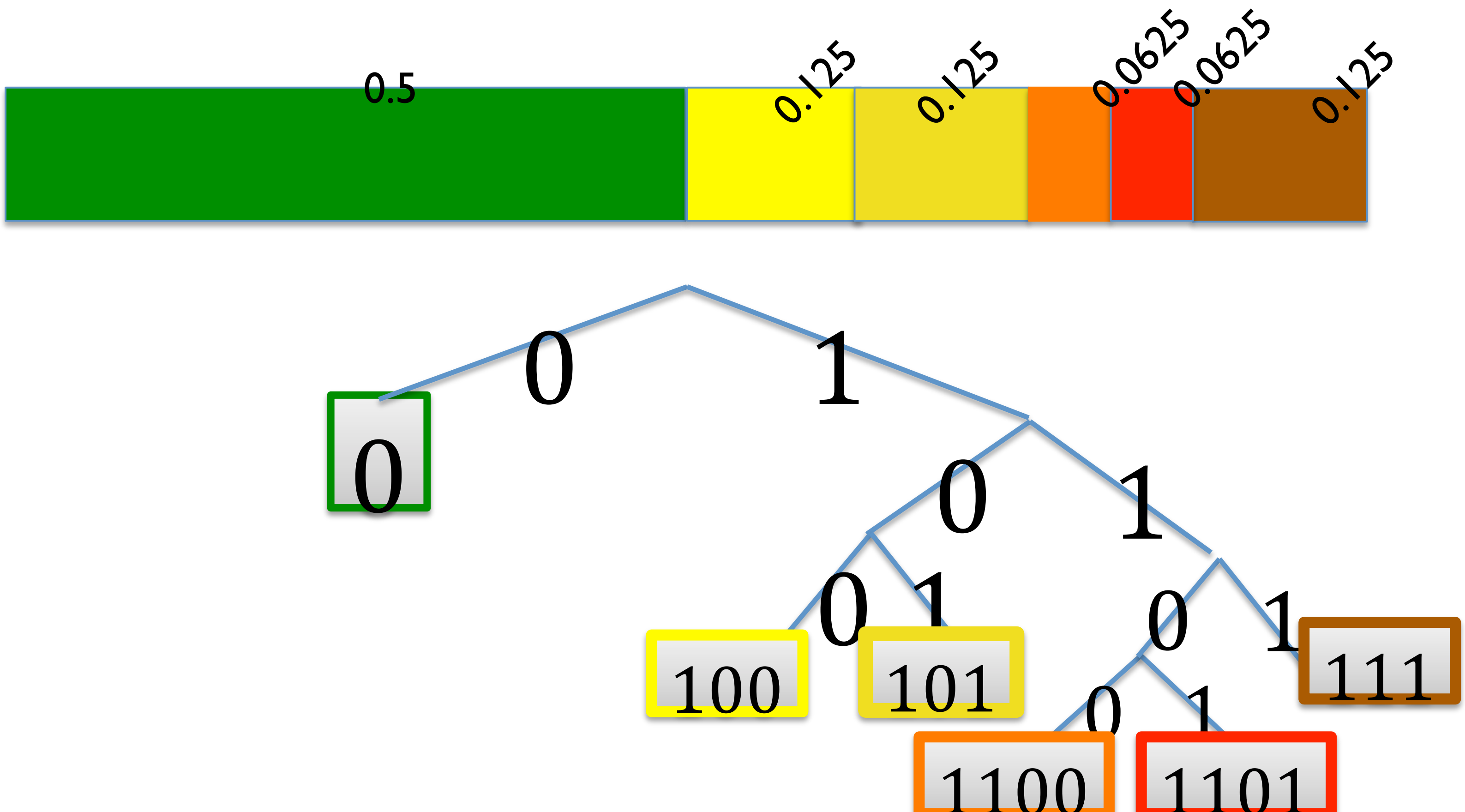
$$E[nbits] = \frac{1}{2}*1 + \frac{1}{8}*3 + \frac{1}{8}*3 + \frac{1}{16}*4 + \frac{1}{16}*4 + \frac{1}{8}*3 = 2.125$$



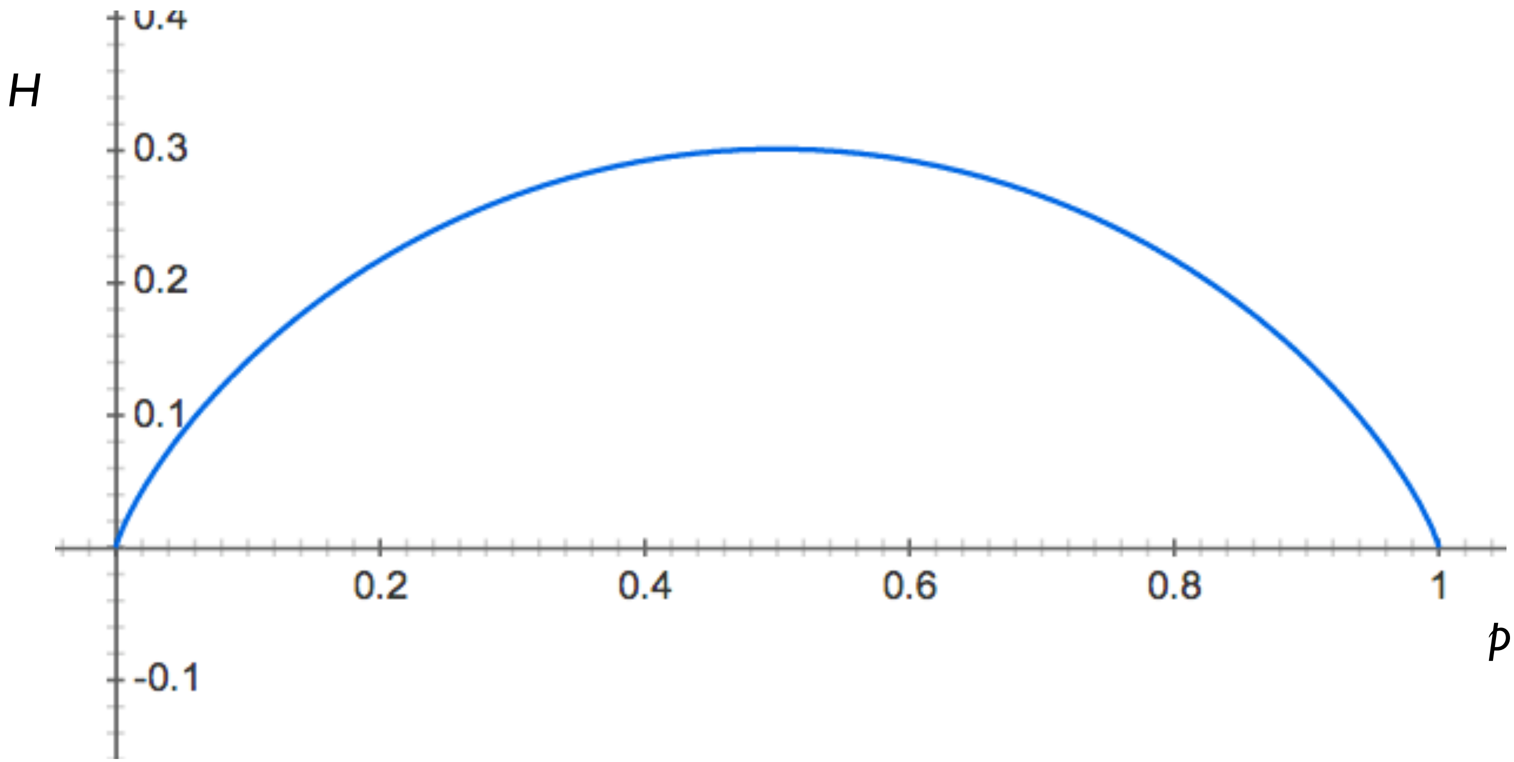


yellow,
green,...
encode
100 0 ...
decode
yellow,green,
...
0
1
0
0
1
0
1
0
1
100
101
111
0
1
1100
1101

$$Entropy(P) = H(P) = -\sum_x p(x) \lg_2 p(x)$$



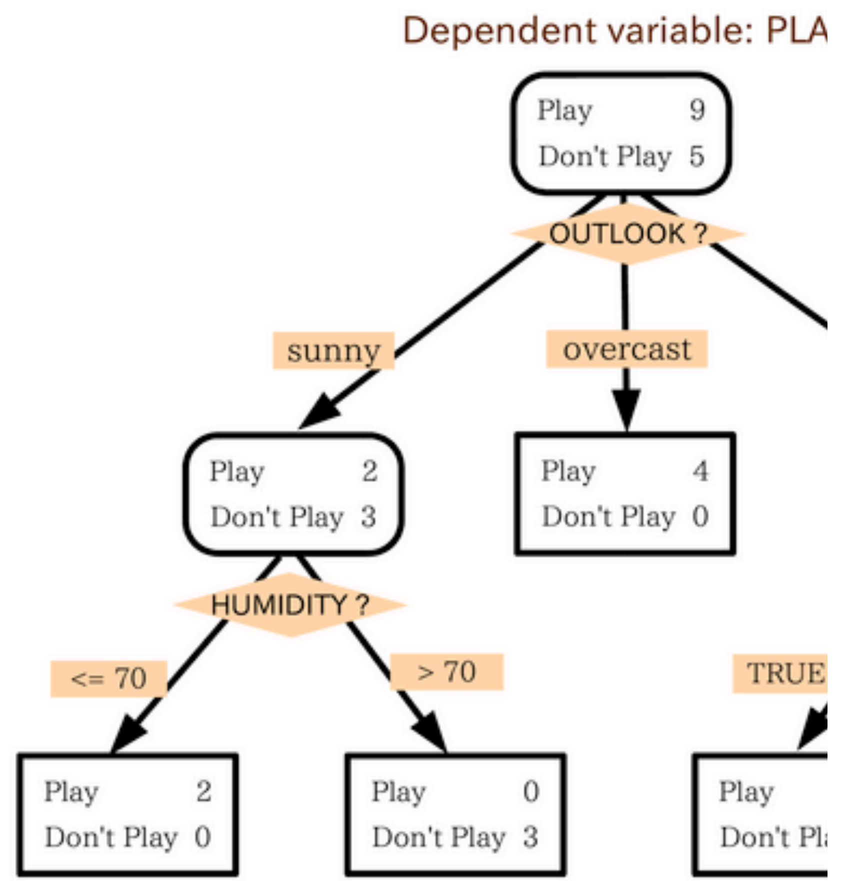
$$H(P) = -\sum_{x} p(x)\lg_2 p(x) = -p\lg p - (1-p)\lg(1-p)$$



DECISION TREE LEARNING: THE ALGORITHM(S)

Most decision tree learning algorithms

1. Given dataset D:
   - return *leaf(y)* if all examples are in the same class *y* … or nearly so
   - pick the best split, on the best attribute *a*
     - $a=c_1$ or $a=c_2$ or …
     - $a<\theta$ or $a \geq \theta$
     - a or $not(a)$
     - a in $\{c_1,\ldots,c_k\}$ or not
   - split the data into D_1, D_2, … D_k and recursively build trees for each subset
2. "Prune" the tree



Most decision tree learning algorithms

1. Given dataset D:
   - return *leaf(y)* if all examples are in the same class *y* … or nearly so...
   - pick the best split, on the best attribute *a*
     - $a=c_1$ or $a=c_2$ or …
     - $a<\theta$ or $a \geq \theta$
     - *a* or *not(a)*
     - *a in* $\{c_1,\ldots,c_k\}$ or not
   - split the data into D_1, D_2, … D_k and recursively build trees for each subset
2. "Prune" the tree

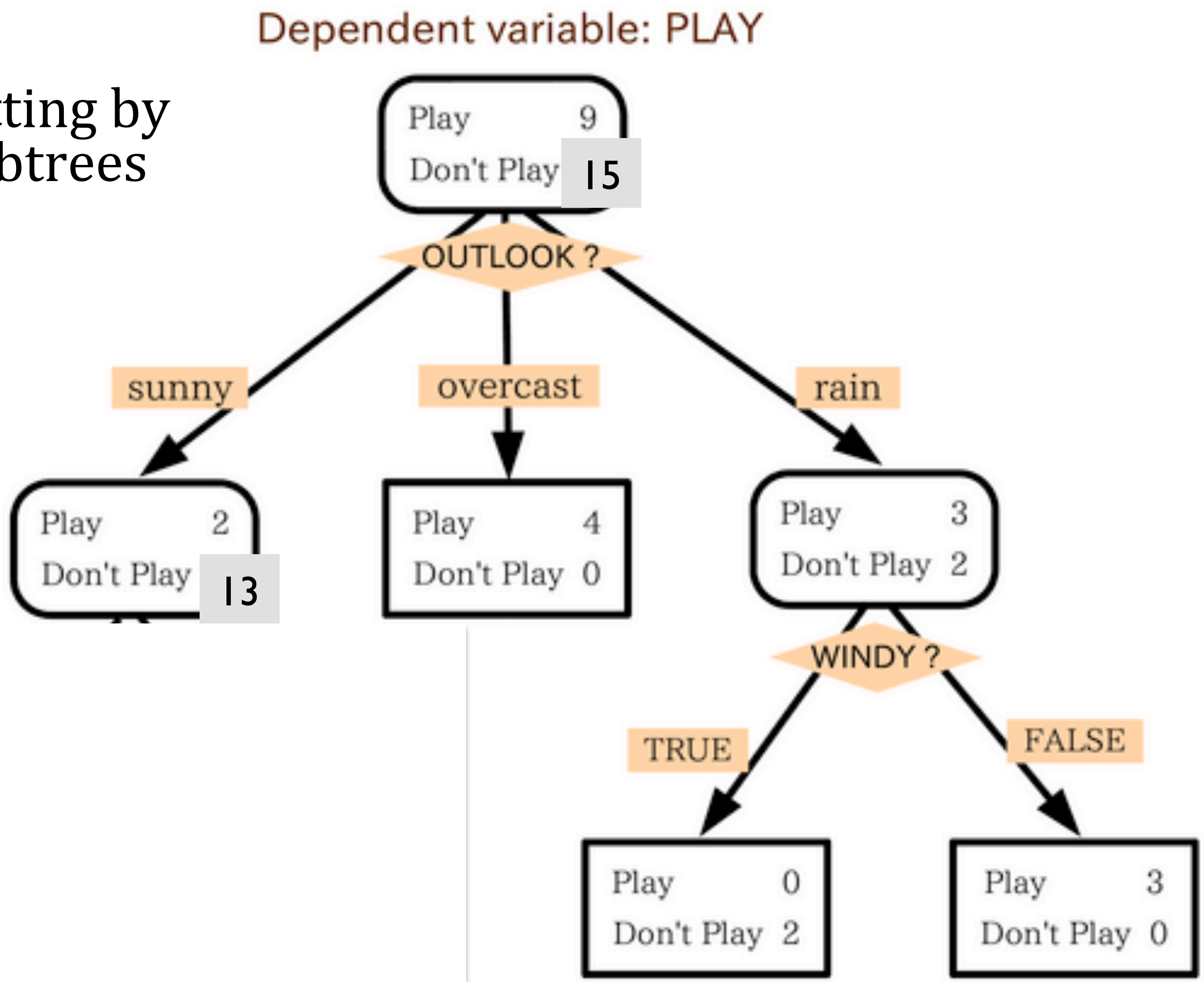
Popular splitting criterion: try to lower *entropy* of the *y* labels on the resulting partition

- i.e., prefer splits that have skewed distributions of labels

$$H(D) = \sum_k \Pr_D(Y = y_k)\log\left[\Pr_D(Y = y_k)\right]$$

Most decision tree learning algorithms

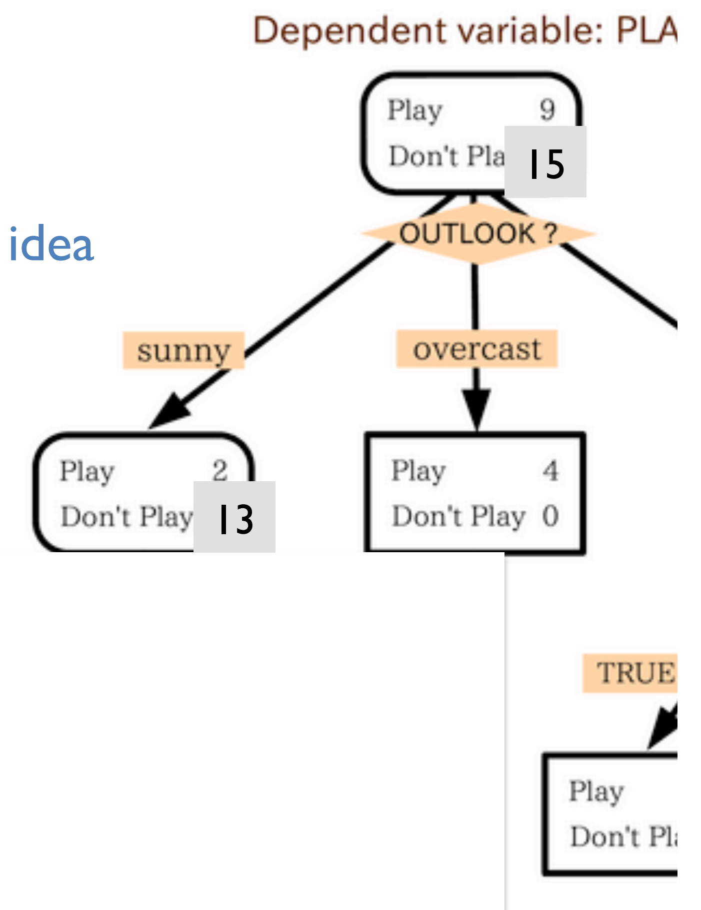
- "Pruning" a tree
  - avoid overfitting by removing subtrees somehow



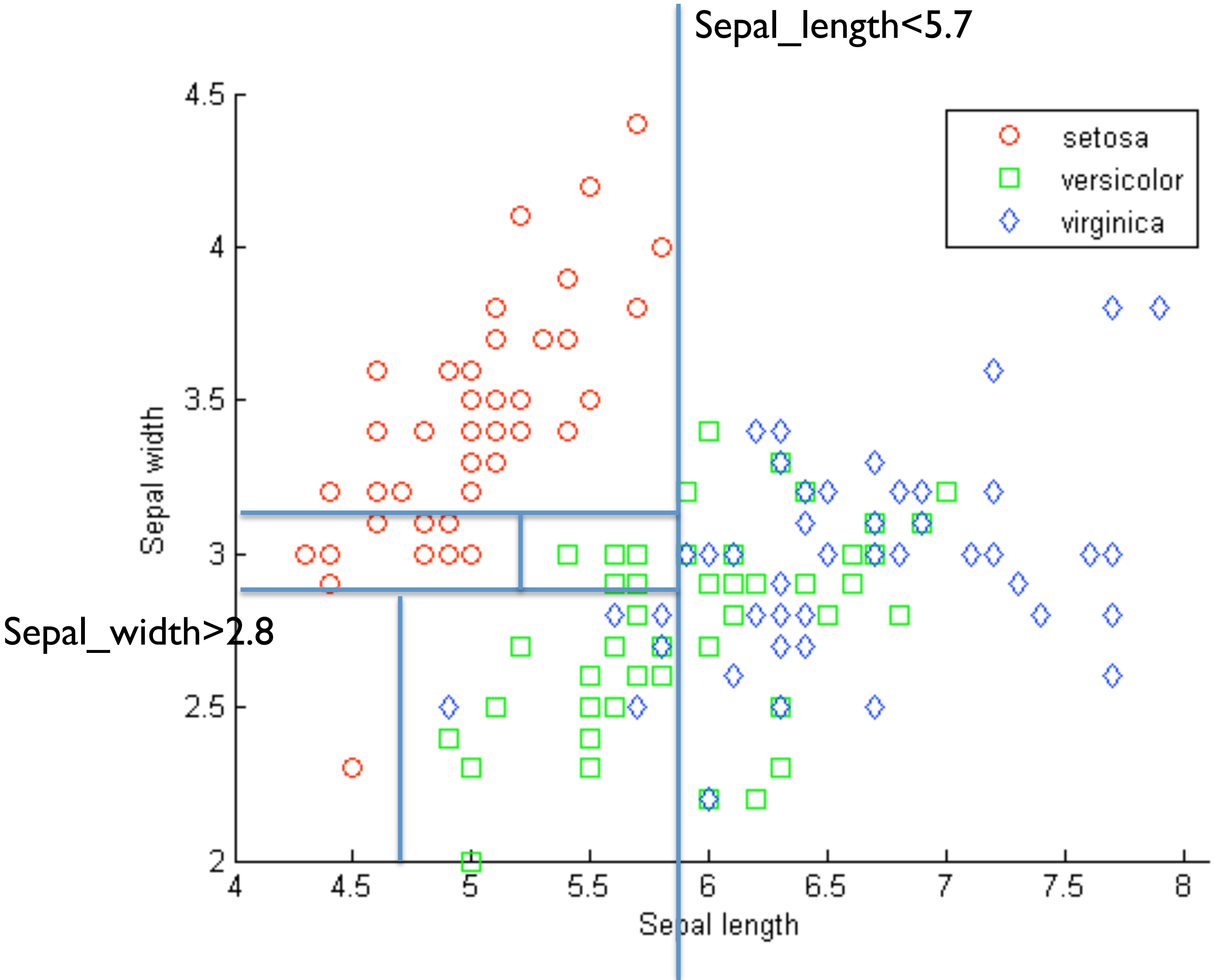
Most decision tree learning algorithms

1. Given dataset D:
   – return *leaf(y)* if all examples are in the same class *y* … or nearly so..
   – pick the best split, on the best attribute *a*
     - $a < \theta$ or $a \geq \theta$
     - a or $not(a)$
     - $a = c_1$ or $a = c_2$ or …
     - a in $\{c_1, \ldots, c_k\}$ or not
   – split the data into D_1, D_2, … D_k and recursively build trees for each subset
2. "Prune" the tree

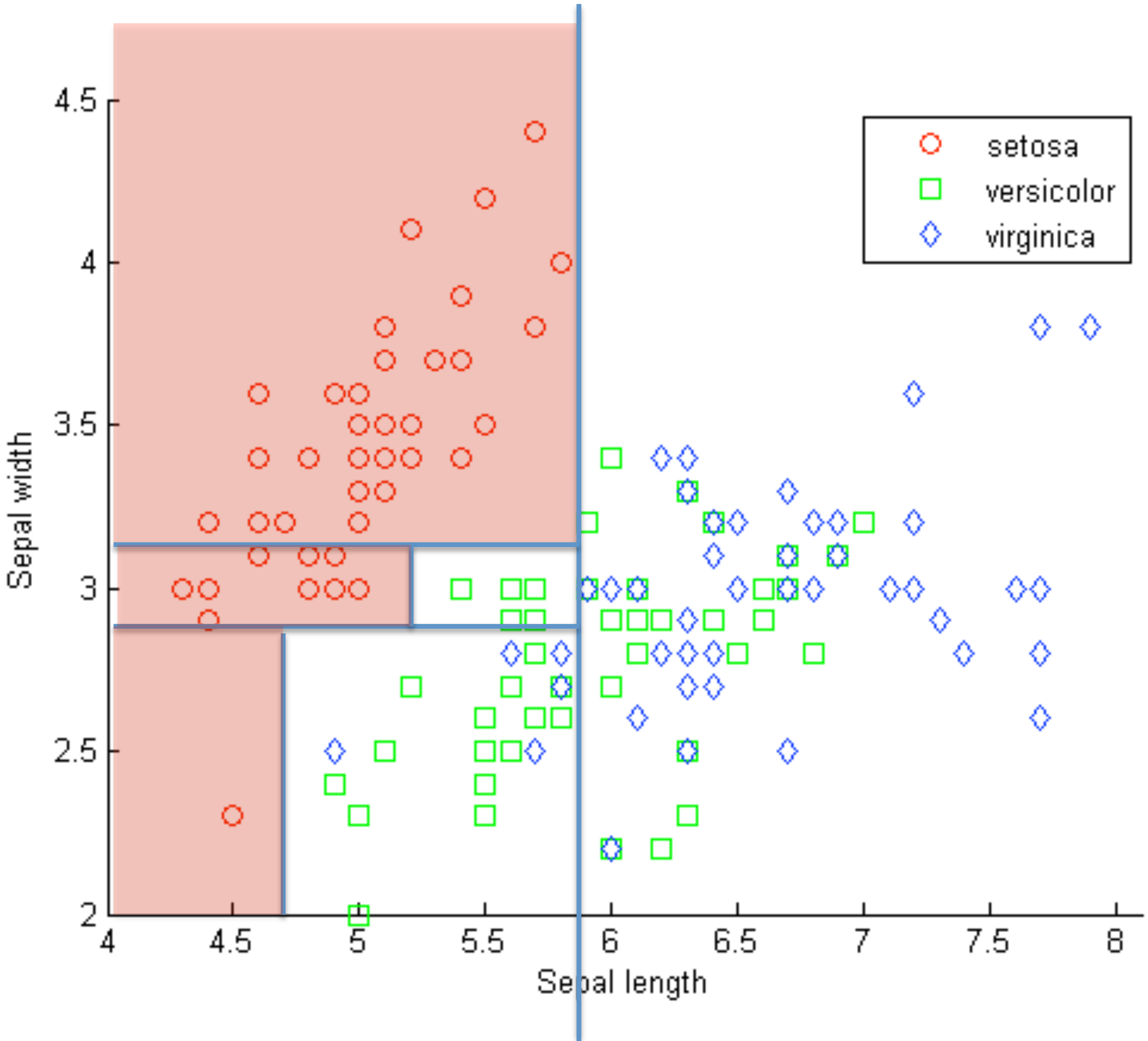
Same idea



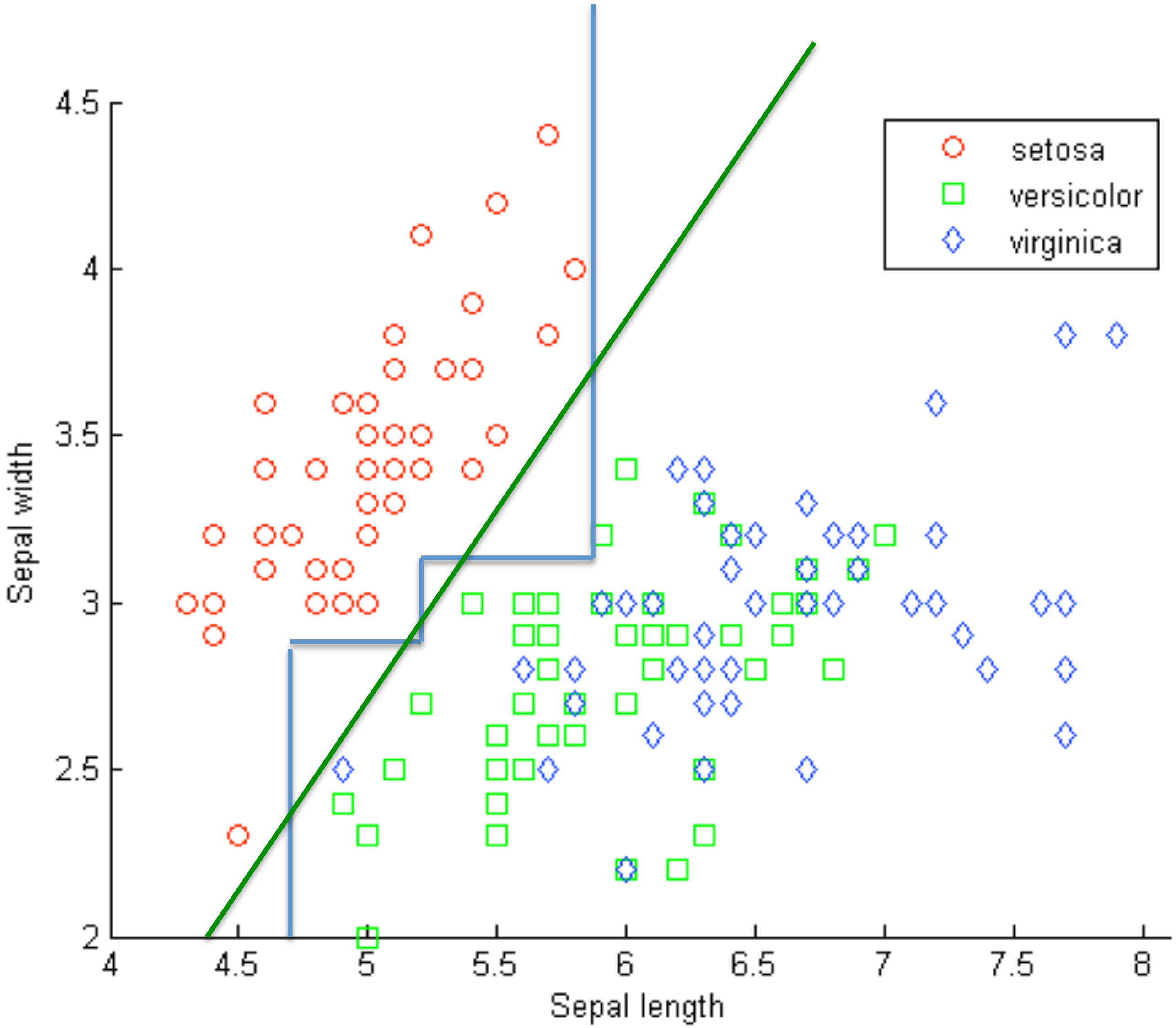
Another view of a decision tree



Another view of a decision tree

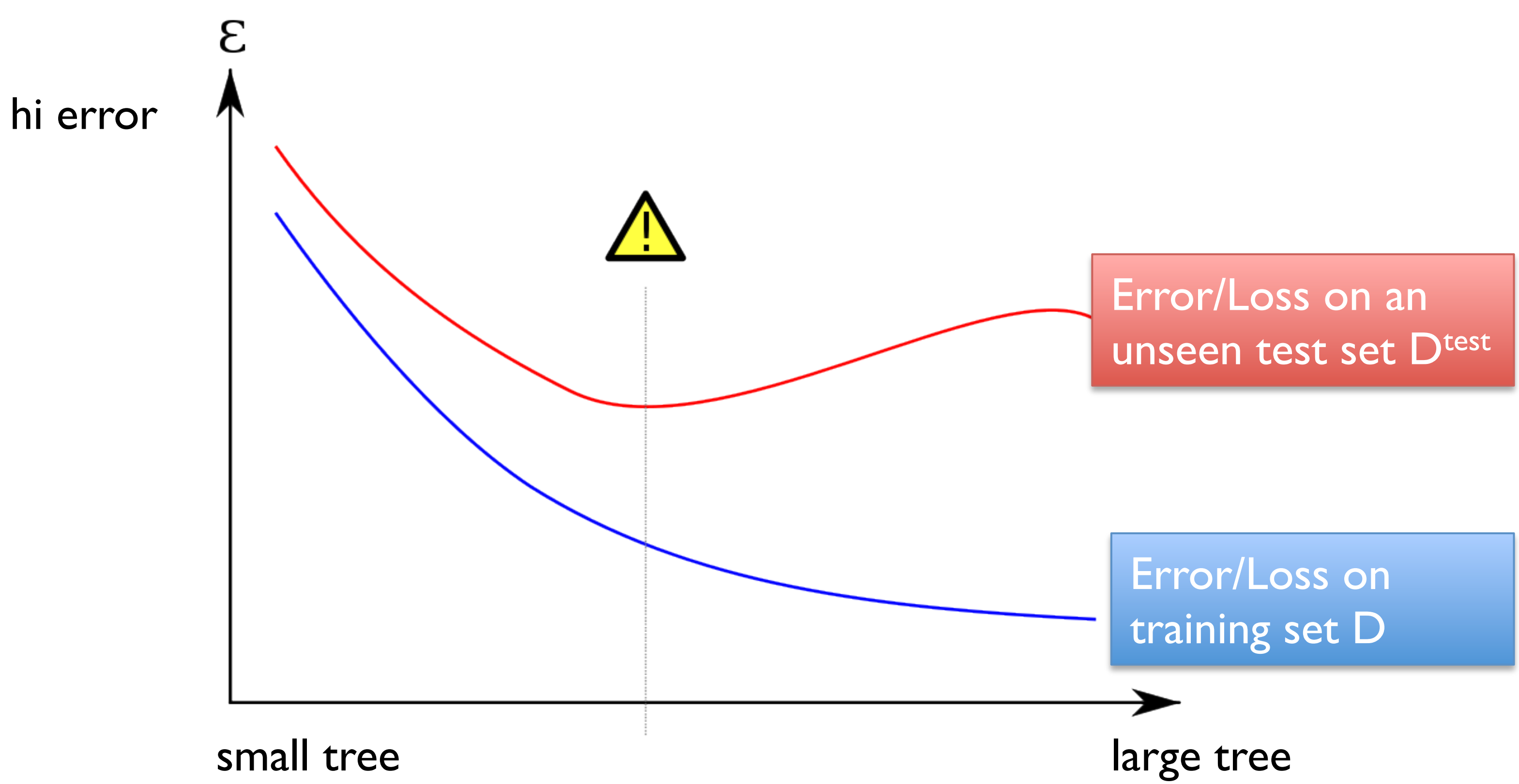


Another view of a decision tree



Overfitting and k-NN

- Small tree ➔ a smooth decision boundary
- Large tree ➔ a complicated shape
- What's the best size decision tree?



DECISION TREE LEARNING: BREAKING IT DOWN

Breaking down decision tree learning

- First: how to classify - assume everything is binary

```
function prPos = classifyTree(T, x)
    if T is a leaf node with counts n,p
        prPos = (p + 1)/(p + n +2)          -- Laplace smoothing
    else
        j = T.splitAttribute
        if x(j)==0 then prPos = classifyTree(T.leftSon, x)
        else prPos = classifyTree(T.rightSon, x)
```

Breaking down decision tree learning

- Reduced error pruning with information gain
  - Split the data D (2/3, 1/3) into D_{grow} and D_{prune}
  - Build the tree recursively with D_{grow}

    $T = growTree(D_{grow})$
  - Prune the tree with D_{prune}

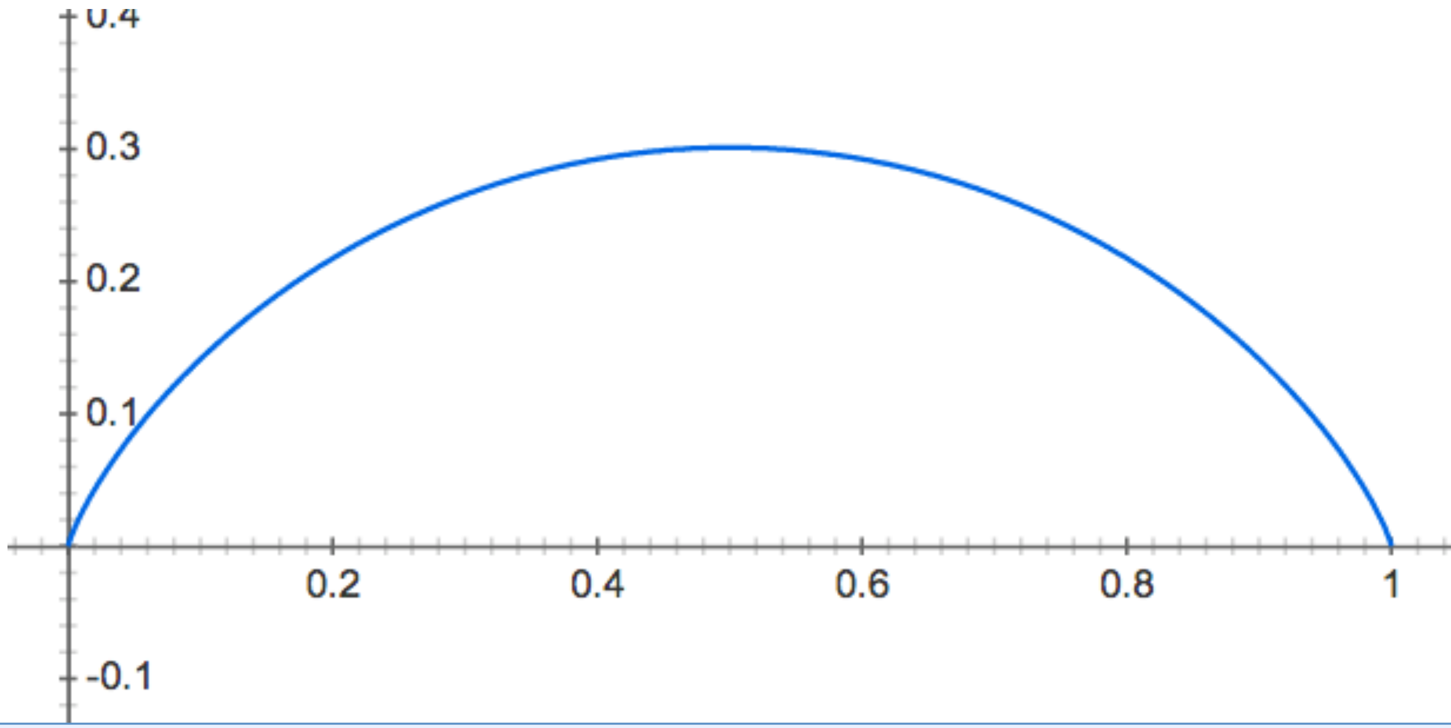
    $T' = pruneTree(D_{prune}, T)$
  - Return T'

Breaking down decision tree learning

- First: divide & conquer to build the tree with D_{grow}

```
function T = growTree(X,Y)
    if |X|<10 or allOneLabel(Y) then
        T = leafNode(|Y==0|,|Y==1|)          -- counts for n,p
    else
        for i = 1,…n                          -- for each attribute i
            ai = X(: , i)                     -- column i of X
            gain(i) = infoGain( Y, Y(ai==0), Y(ai==1) )
        j = argmax(gain);  -- the best attribute
        aj = X(:, j)
        T = splitNode( growTree(X(aj==0),Y(aj==0)),   -- left son
                       growTree(X(aj==1),Y(aj==1)),  --right son
                       j)
```

Breaking down decision tree learning




```
function e = entropy(Y)
    n = |Y|;    p0 = |Y==0|/n;   p1 = |Y==1| /n;
    e = - p0*log(p0) - p1*log(p1)
```

Breaking down decision tree learning

- First: how to build the tree with D_{grow}

```
function g = infoGain(Y,leftY,rightY)
     n = |Y|;  nLeft = |leftY|; nRight = |rightY|;
     g = entropy(Y)
           - (nLeft/n)*entropy(leftY) - (nRight/n)*entropy(rightY)

function e = entropy(Y)
     n = |Y|;    p0 = |Y==0|/n;   p1 = |Y==1| /n;
     e = - p1*log(p1) - p2*log(p2)
```

Breaking down decision tree learning

- Reduced error pruning with information gain
  - Split the data D (2/3, 1/3) into D_{grow} and D_{prune}
  - Build the tree recursively with D_{grow}

    $T = \text{growTree}(D_{grow})$
  - Prune the tree with D_{prune}

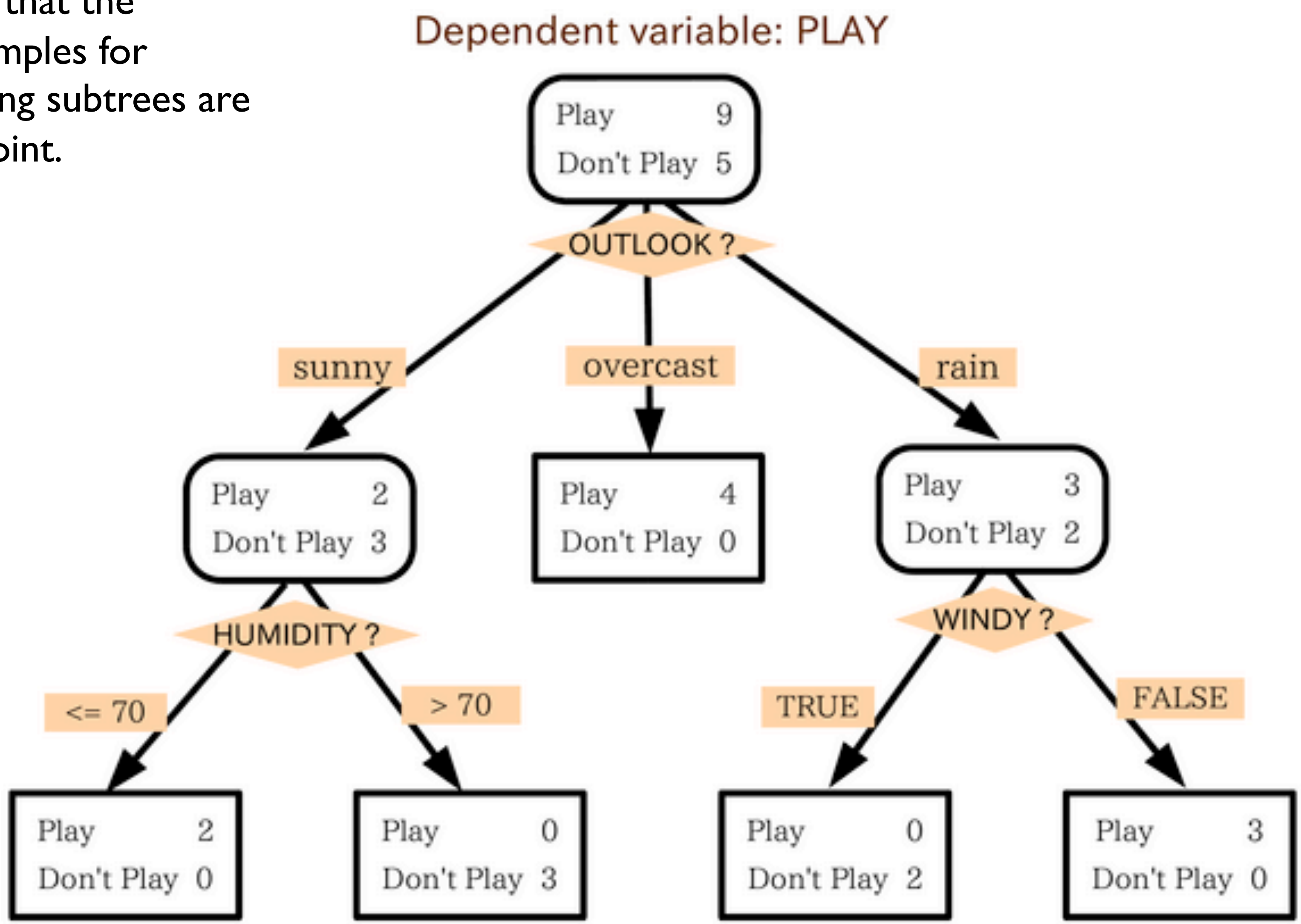
    $T' = \text{pruneTree}(D_{prune})$
  - Return T’

Breaking down decision tree learning

- Next: how to prune the tree with D_{prune}
  - Estimate the error rate of every subtree on D_{prune}
  - Recursively traverse the tree:
    - Reduce error on the left, right subtrees of T
    - If T would have lower error if it were converted to a leaf, convert T to a leaf.

A decision tree

We're using the fact that the examples for sibling subtrees are disjoint.



Breaking down decision tree learning

- To estimate error rates, classify the whole pruning set, and keep some counts

```
function classifyPruneSet(T, X, Y)
    T.pruneN = |Y==0|; T.pruneP = |Y==1|
    if T is not a leaf then
        j = T.splitAttribute
        aj = X(:, j)
        classifyPruneSet( T.leftSon,    X(aj==0), Y(aj==0) )
        classifyPruneSet( T.rightSon, X(aj==1),  Y(aj==1) )
```

```
function e = errorsOnPruneSetAsLeaf(T):
    min(T.pruneN, T.pruneP)
```

Breaking down decision tree learning

- Next: how to prune the tree with D_{prune}
  - Estimate the error rate of every subtree on D_{prune}
  - Recursively traverse the tree:

```
function T1 = pruned(T)
    if T is a leaf then
        -- copy T, adding an error estimate T.minErrors
        T1= leaf(T, errorsOnPruneSetAsLeaf(T))
    else
        e1 = errorsOnPruneSetAsLeaf(T)
        TLeft = pruned(T.leftSon); TRight = pruned(T.rightSon);
        e2 = TLeft.minErrors + TRight.minErrors;
        if e1 <= e2 then T1 = leaf(T,e1) -- cp + add error estimate
        else T1 = splitNode(T, e2)       -- cp + add error estimate
```

Decision trees: plus and minus

- Simple and fast to learn
- Arguably easy to understand (if compact)
- Very fast to *use*:
  - often you don't even need to compute all attribute values
- Can find interactions between variables (play if it's cool and sunny or ....) and hence non-linear decision boundaries
- Don't need to worry about how numeric values are scaled

Decision trees: plus and minus

- Hard to prove things about
- Not well-suited to probabilistic extensions
- Sometimes fail badly on problems that seem easy
  - the IRIS dataset is an example

Fixing decision trees….

- Hard to prove things about
- Don’t (typically) improve over linear classifiers when you have lots of features
- Sometimes fail badly on problems that linear classifiers perform well on
  - One solution is to build *ensembles* of decision trees
  - more on this later

RULE LEARNING: OVERVIEW

Rules for Subcategories of Economist Articles

```
Final hypothesis is:
aboutInternational :- wordsInArticle ~ countries, wordsInArticle ~ nations (9/4).
aboutInternational :- wordsInArticle ~ soil (7/1).
aboutInternational :- wordsInArticle ~ based, wordsInArticle ~ authorities (6/4).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ barack (19/0).
aboutNorthAmerica :- wordsInArticle ~ barack (9/3).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ americans (13/1).
aboutNorthAmerica :- wordsInArticle ~ texas (7/2).
aboutNorthAmerica :- wordsInArticle ~ pricing (4/2).
aboutNorthAmerica :- wordsInArticle ~ huckabee (2/0).
aboutNorthAmerica :- wordsInArticle ~ miller (2/0).
aboutLatinAmerica :- wordsInArticle ~ n, wordsInArticle ~ president (23/2).
aboutLatinAmerica :- wordsInArticle ~ brazil (15/6).
aboutLatinAmerica :- wordsInArticle ~ latin (6/3).
aboutLatinAmerica :- wordsInArticle ~ fidel (8/0).
aboutLatinAmerica :- wordsInArticle ~ lvaro (3/0).
aboutLatinAmerica :- wordsInArticle ~ bolivia (5/0).
aboutLatinAmerica :- wordsInArticle ~ canadians (2/0).
aboutAfrica :- wordsInArticle ~ africa, wordsInArticle ~ president (32/4).
aboutAfrica :- wordsInArticle ~ al (15/7).
aboutAfrica :- wordsInArticle ~ lebanon (5/0).
aboutAfrica :- wordsInArticle ~ nigeria (3/1).
aboutAsia :- wordsInArticle ~ china (35/13).
aboutAsia :- wordsInArticle ~ india (12/3).
aboutAsia :- wordsInArticle ~ e09as761 (6/0).
aboutAsia :- wordsInArticle ~ interim (6/2).
aboutAsia :- wordsInArticle ~ park (4/2).
aboutBritain :- wordsInArticle ~ britain, wordsInArticle ~ british (34/7).
aboutBritain :- wordsInArticle ~ technology (11/2).
aboutBritain :- wordsInArticle ~ brown (17/3).
aboutBritain :- wordsInArticle ~ england (5/1).
aboutBritain :- wordsInArticle ~ craft (2/0).
default aboutEurope (91/42).
============================== summary ==============================
Train error rate:  21.58% +/- 1.78% (533 datapoints)    <<
Hypothesis size:   31 rules, 69 conditions
Learning time:     6.47 sec
```

Trees vs Rules

- For every tree with L leaves, there is a corresponding rule set with L rules
  - So one way to learn rules is to extract them from trees.
- But:
  - Sometimes the extracted rules can be drastically simplified
  - For some rule sets, there is *no* tree that is nearly the same size
  - So rules are more expressive given a size constraint
- This motivated learning rules *directly*

Separate and conquer rule-learning

- Start with an empty rule set
- Iteratively do this
  - Find a rule that works well on the data

    On later iterations, the data is different
  - Remove the examples "covered by" the rule (they satisfy the "if" part) from the data
- Stop when all data is covered by some rule
- Possibly *prune* the rule set

Separate and conquer rule-learning

- Start with an empty rule set
- Iteratively do this
  - Find a rule that works well on the data
    - Start with an empty rule
    - Iteratively
      - Add a condition that is true for many positive and few negative examples
      - Stop when the rule covers no negative examples (or almost no negative examples)
  - Remove the examples "covered by" the rule
- Stop when all data is covered by some rule

Separate and conquer rule-learning

```
functon Rules = separateAndConquer(X,Y)
    Rules = empty rule set
    while there are positive examples in X,Y not covered by rule R do
        R = empty list of conditions
        CoveredX = X; CoveredY = Y;
        -- specialize R until it covers only positive examples
        while CoveredY contains some negative examples
            -- compute the "gain" for each condition x(j)==1
            ….
            j = argmax(gain); aj=CoveredX(:,j);
            R = R conjoined with condition "x(j)==1"  -- add best condition
            -- remove examples not covered by R from CoveredX,CoveredY
            CoveredX = X(aj); CoveredY = Y(aj);
        Rules = Rules plus new rule R
        -- remove examples covered by R from X,Y
        …
```