COMP 652: Machine Learning

Lecture 11

Today

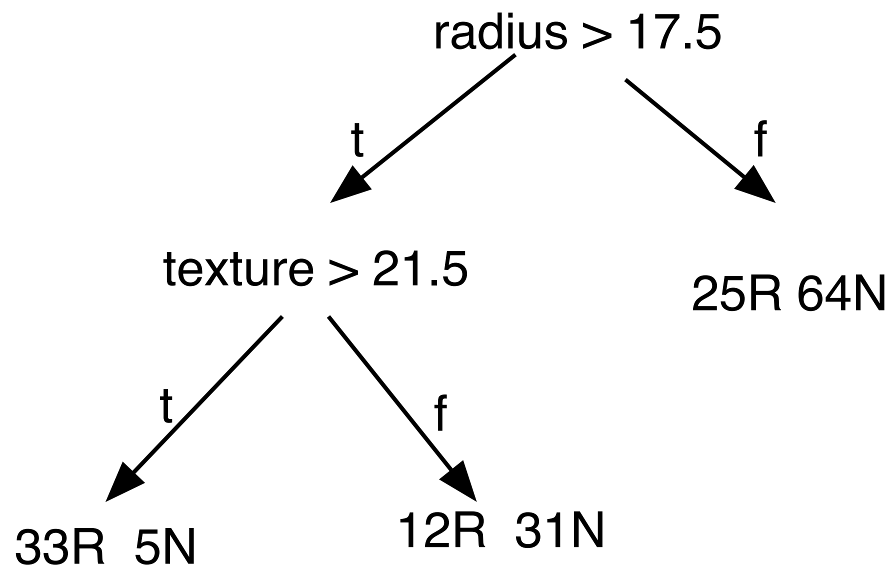
- ☐ Decision trees - what are they?
- ☐ Decision tree construction
- ☐ Brief interlude on information theory
- ☐ Overfitting avoidance
- ☐ Variations on the standard algorithm
- ☐ Regression trees

Parametric, nonparametric, and non-metric learning

- Parametric methods summarize the existing data set into a set of parameters
- Nonparametric methods "memorize" the data, then use a distance metric to measure the similarity of new query points to existing examples
- Today: *non-metric methods*
  We do not get a set of parameters, but at the same time there is no distance metric to assess similarity of different instances
- Typical examples:
  - Decision trees
  - Rule-based systems

- (We'll return to non-parametric, but metric, supervised learning next week, with support vector machines.)

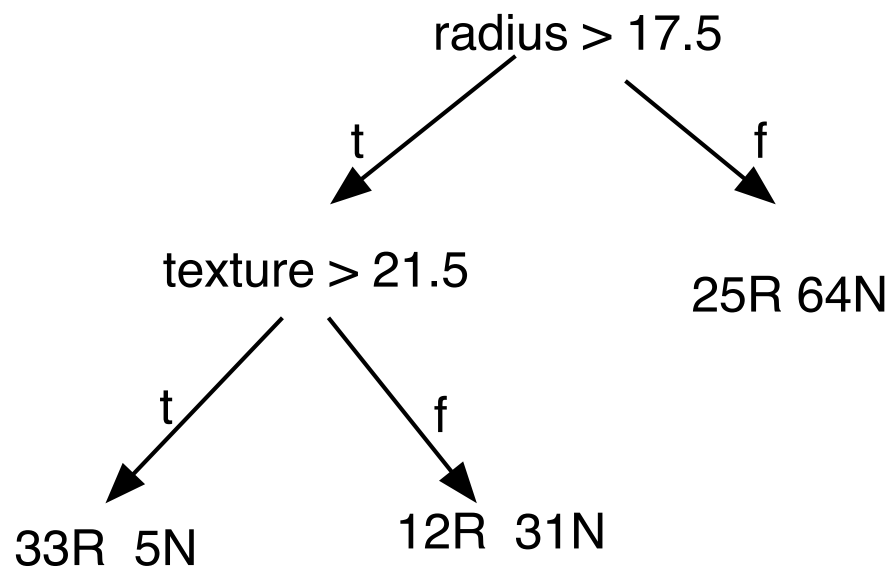
Example: decision tree for predicting cancer recurrence



- Internal nodes are tests on the values of different attributes (Need not be binary tests.)
- Each training example $(\mathbf{x}_i, y_i)$ falls in precisely one leaf.

Using decision trees for classification

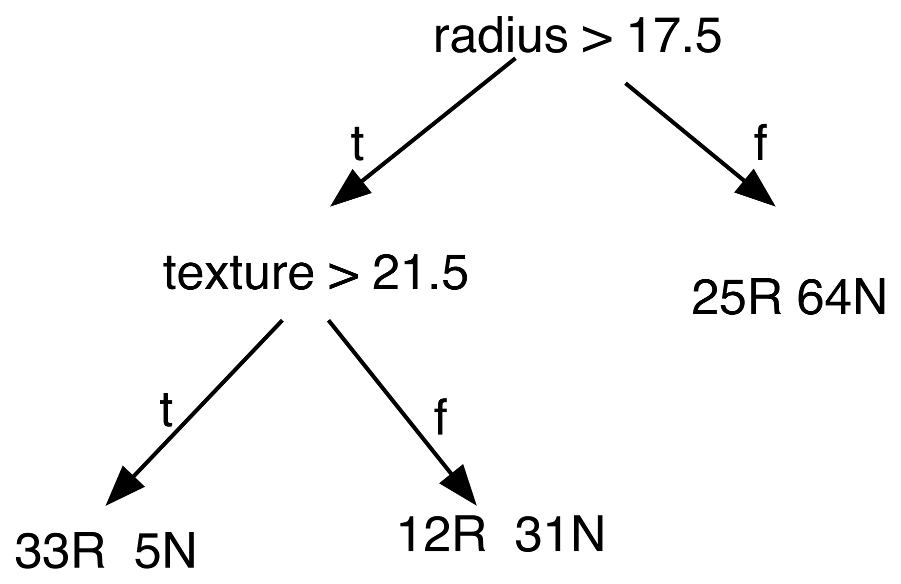
Suppose we get a new instance: *radius=18, texture=12, ...*
How do we classify it?



- At every node, test the corresponding attribute
- Follow the appropriate branch of the tree
- At a leaf, one can predict the class of the majority of the examples for the corresponding leaf, or the probabilities of the two classes.

Decision trees as logical representations

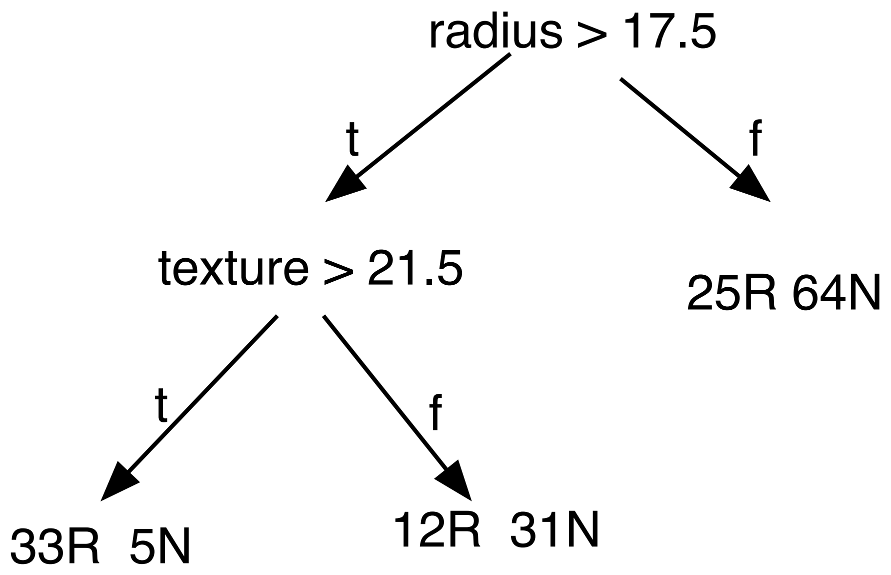
We can convert a decision tree into an equivalent set of if-then rules.



IF	THEN most likely class is
radius$>$17.5 AND texture$>$21.5	R
radius$>$17.5 AND texture$\leq$21.5	N
radius$\leq$17.5	N

Decision trees as logical representations

We can convert a decision tree into an equivalent set of if-then rules.



IF	THEN P(R) is
radius$>$17.5 AND texture$>$21.5	$\frac{33}{33+5}$
radius$>$17.5 AND texture$\leq$21.5	$\frac{12}{12+31}$
radius$\leq$17.5	$\frac{25}{25+64}$

Decision trees, more formally: tests

- Each internal node contains a ***test***
- Depends on the value of one (typically) or more feature values
- Test produces discrete outcome. E.g.,
  - radius > 17.5
  - radius $\in [12, 18]$
  - grade is $\{A, B, C, D, F\}$
  - grade is $\geq B$
  - color is RED
  - $2*$radius$-3*$texture > 16
- For discrete features, we typically branch on all possibilities
- For real features, we typically branch on a threshold value

$\Rightarrow$ A ***finite*** set of possible tests is usually decided before learning the tree; learning thus comprises choosing the shape of the tree and the tests at every node.

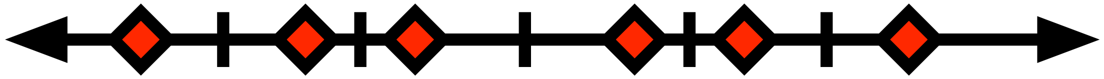
More on tests for real-valued features

- Suppose feature j is real-valued
- How do we choose a finite set of possible thresholds, for tests of the form $\mathbf{x}_j > \tau$?

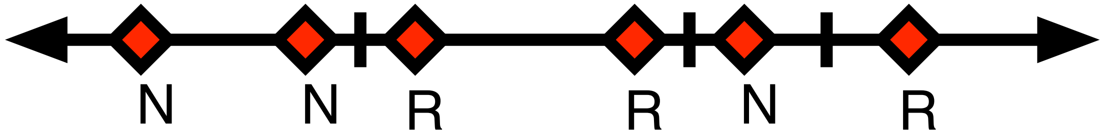
More on tests for real-valued features

- Suppose feature j is real-valued,
- How do we choose a finite set of possible thresholds, for tests of the form $\mathbf{x}_j > \tau$?

- Choose some discrete (e.g., evenly space thresholds) to cover the range of the variable
- Choose midpoints of the observed data values, $x_{1,j}, x_{2,j}, \ldots, x_{m,j}$



- Choose midpoints of data values with different y values



Representational power and efficiency of decision trees

- Suppose $\mathbf{x}$ comprises N binary features
- Can / how can a decision tree represent:
  - $y = x_1$ AND x_2 AND ... AND x_N
  - $y = x_1$ OR x_2 OR ... OR x_N
  - $y = x_1$ XOR x_2 XOR ... XOR x_N

Representational power and efficiency of decision trees

- □ Suppose $\mathbf{x}$ comprises N binary features
- □ Can / how can a decision tree represent:
  - – $y = x_1$ AND x_2 AND ... AND x_N
  - – $y = x_1$ OR x_2 OR ... OR x_N
  - – $y = x_1$ XOR x_2 XOR ... XOR x_N

⇒ With typical univariate tests, AND and OR are easy, taking O(M) tests, whereas XOR takes $O(2^M)$ tests

Representational power and efficiency of decision trees (II)

- Suppose $\mathbf{x}$ comprises 2 real-valued features
- Can / how can a decision tree represent:
  - $y = (x_1 > 10 \text{ OR } x_2 < 3)$
  - $y = (x_1 + x_2 > 1)$

Representational power and efficiency of decision trees (II)

- □ Suppose $\mathbf{x}$ comprises 2 real-valued features
- □ Can / how can a decision tree represent:
  - $y = (x_1 > 10$ OR $x_2 < 3)$
  - $y = (x_1 + x_2 > 1)$

⇒ Again, with typical tests, decision trees are good a classifications comprising large, connected, axis-orthogonal regions of inputs space.

How do we learn decision trees?

- We *could* enumerate all possible trees (assuming number of possible tests is finite),
  - Each tree could be evaluated using the training set or, better yet, a validation set
  - But there are many possible trees!
  - We'd probably overfit the data anyway
- Usually, decision trees are constructed in two phases:
  1. An recursive, top-down procedure "grows" a tree (possibly until the training data is completely fit)
  2. The tree is "pruned" back to avoid overfitting

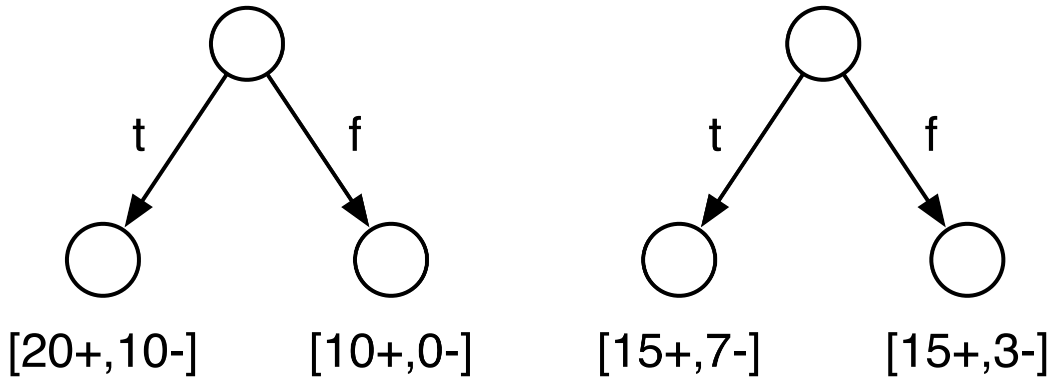
Top-down (recursive) induction of decision trees

Given a set of labeled training instances:

1. If all the training instances have the same class, create a leaf with that class label and exit.
2. Pick the ***best*** test to split the data on
3. Split the training set according to the value of the outcome of the test
4. Recurse on each subset of the training data

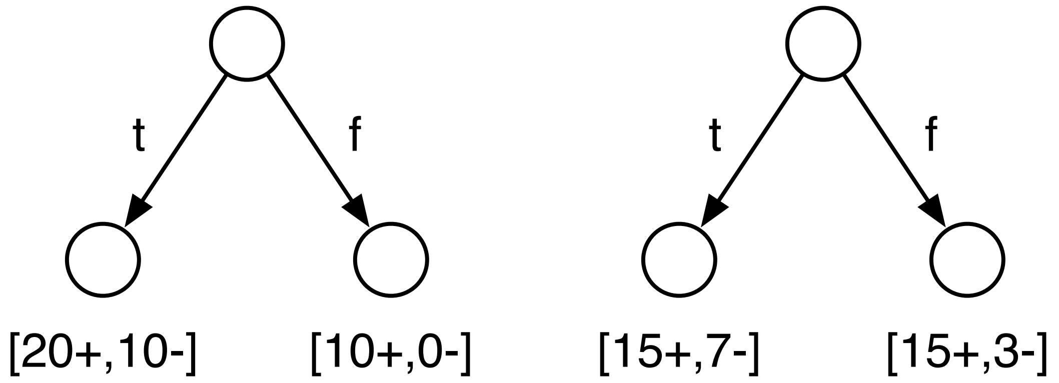
Which test is best?

The test should provide **information** about the class label.
Suppose we have 30 positive examples, 10 negative ones, and we are considering two tests that would give the following splits of instances:



Which test is best?

The test should provide **information** about the class label.
Suppose we have 30 positive examples, 10 negative ones, and we are considering two tests that would give the following splits of instances:



Intuitively, we would like an attribute that *separates* the training instances as well as possible
We need a mathematical measure for the **purity** of a set of instances

What is information?

Imagine:

1. You are about to observe the outcome of a dice roll
2. You are about to observe the outcome of a coin flip
3. You are bout to observe the outcome of a biased coin flip
4. Someone is about to tell you your own name

Intuitively, in each situation you have a different amount of uncertainty as to what outcome / message you will observe.

Information=Reduction in uncertainty

Let E be an event that occurs with probability $P(E)$. If we are told that E has occurred with certainty, then we received

$$I(E) = \log_2 \frac{1}{P(E)}$$

bits of **information**.

- □ You can also think of information as the amount of "surprise" in the outcome (e.g., consider $P(E) = 1$, $P(E) \approx 0$)
- □ Example: result of a fair coin flip provides $\log_2 2 = 1$ bit of information
- □ Example: result of a fair dice roll provides $\log_2 6 \approx 2.58$ bits of information.

Information is additive

Suppose you have k independent fair coin tosses. How much information do they give?

$$I(k \text{ fair coin tosses }) = \log_2 \frac{1}{1/2^k} = k \text{ bits}$$

A cute example:

- Consider a random word drawn from a vocabulary of 100,000 words: $I(\text{word}) = \log_2 100,000 \approx 16.61$ bits
- Now consider a 1000 word document drawn from the same source: $I(\text{document}) \approx 16610$ bits
- Now consider a 480×640 gray-scale image with 16 grey levels: $I(\text{picture}) = 307,200 \cdot \log_2 16 = 1,228,800$ bits!

$\Longrightarrow$ A picture is worth (more than) a thousand words!

Average information

Suppose we have an information source S which emits symbols from an alphabet $\{s_1, \dots s_k\}$ with probabilities $\{p_1, \dots p_k\}$. Each emission is independent of the others.
What is the **average amount of information** when observing the output of S?

Entropy

Suppose we have an information source S which emits symbols from an alphabet $\{s_1, \ldots s_k\}$ with probabilities $\{p_1, \ldots p_k\}$. Each emission is independent of the others.

What is the **average amount of information** when observing the output of S?

$$H(S) = \sum_i p_i I(s_i) = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i$$

Call this **entropy** of S.

Note that this depends *only on the probability distribution* and not on the actual alphabet (so we can really write $H(P)$).

Interpretations of entropy

$$H(P) = \sum_i p_i \log \frac{1}{p_i}$$

- Average amount of information per symbol
- Average amount of surprise when observing the symbol
- Uncertainty the observer has before seeing the symbol
- Average number of bits needed to communicate the symbol

Entropy and coding theory

- Suppose I will get data from a 4-value alphabet $\mathbf{z}_j$ and I want to send it over a channel. I know that the probability of item $\mathbf{z}_j$ is p_j.
- Suppose all values are equally likely. Then I can encode them in two bits each, so on every transmission I need 2 bits
- Suppose now $p_0 = 0.5$, $p_1 = 0.25$, $p_2 = p_3 = 0.125$. What is the best encoding?

Entropy and coding theory

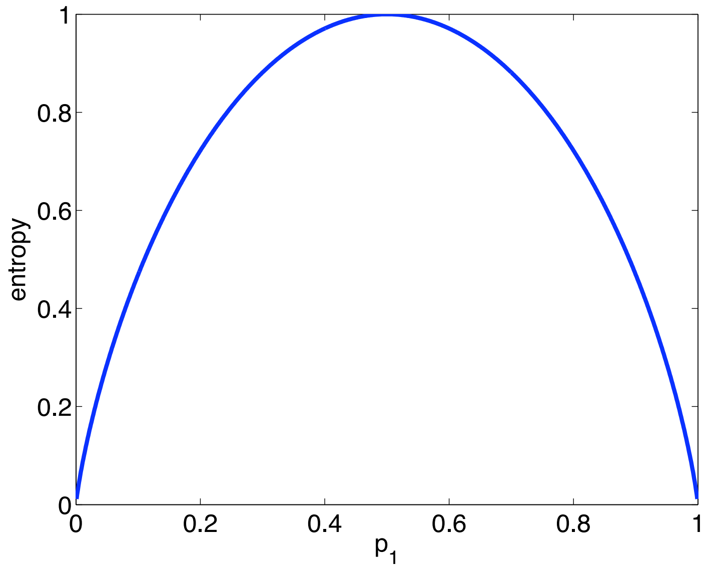
- Suppose I will get data from a 4-value alphabet $\mathbf{z}_j$ and I want to send it over a channel. I know that the probability of item $\mathbf{z}_j$ is p_j.
- Suppose all values are equally likely. Then I can encode them in two bits each, so on every transmission I need 2 bits
- Suppose now $p_0 = 0.5$, $p_1 = 0.25$, $p_2 = p_3 = 0.125$. What is the best encoding?
  $\mathbf{z}_0 = 0$, $\mathbf{z}_1 = 10$, $\mathbf{z}_2 = 110$, $\mathbf{z}_3 = 111$.
  What is the expected length of the message over time?

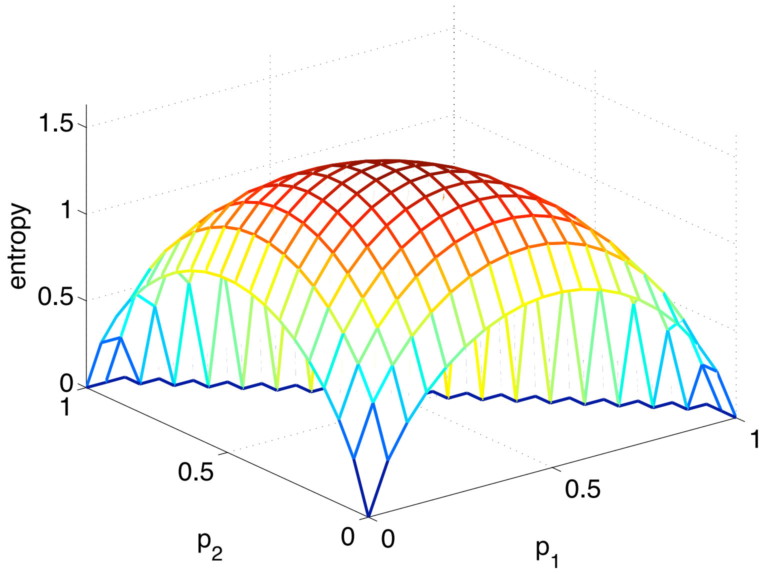
Shannon: there are codes that will communicate the symbols with efficiency arbitrarily close to $H(P)$ bits/symbol. There are no codes that will do it with efficiency greater than $H(P)$ bits/symbol.

Properties of entropy

$$H(P) = \sum_{i=1}^{k} p_i \log \frac{1}{p_i}$$

- Non-negative: $H(P) \geq 0$, with equality if and only if any $p_i = 1$.
- $H(P) \leq \log k$ with equality if and only if $p_i = \frac{1}{k}, \forall i$
- The further P is from uniform, the lower the entropy





Entropy applied to binary classfication

Consider our data set D and let

- ☐ $p_\oplus$ is the proportion of positive examples in D
- ☐ $p_\ominus$ is the proportion of negative examples in D

Entropy measures the impurity of D, based on empirical probabilities of the two classes:

$$H(D) \equiv p_\oplus \log_2 \frac{1}{p_\oplus} + p_\ominus \log_2 \frac{1}{p_\ominus}$$

So we can use it to measure purity!

Conditional entropy

Suppose I am trying to predict output y and I have input x, e.g.:

x =*HasKids*	y =*OwnsDumboVideo*
Yes	*Yes*
Yes	*Yes*
Yes	*Yes*
Yes	*Yes*
No	*No*
No	*No*
Yes	*No*
Yes	*No*

- From the table, we can estimate $P(y = YES) = 0.5 = P(y = NO)$.
- Thus, we estimate $H(y) = 0.5 \log \frac{1}{0.5} + 0.5 \log \frac{1}{0.5} = 1$.
- *Specific conditional entropy* is our uncertainty in y given a particular x value. E.g.,
  - $P(y = YES|x = YES) = \frac{2}{3}$, $P(y = NO|x = YES) = \frac{1}{3}$
  - $H(y|x = YES) = \frac{2}{3} \log \frac{1}{(\frac{2}{3})} + \frac{1}{3} \log \frac{1}{(\frac{1}{3})} \approx 0.9183$.

Conditional entropy

- *Conditional entropy*, $H(y|x)$, is the average conditional entropy of y given specific values for x:

$$H(y|x) = \sum_v P(x = v) H(y|x = v)$$

- E.g. (from previous slide),
  - $H(y|x = YES) = \frac{2}{3} \log \frac{1}{(\frac{2}{3})} + \frac{1}{3} \log \frac{1}{(\frac{1}{3})} \approx 0.6365$
  - $H(y|x = NO) = 0 \log \frac{1}{0} + 1 \log \frac{1}{1} = 0.$
  - $H(y|x) = H(y|x = YES)P(x = YES) + H(y|x = NO)P(x = NO) = 0.6365 * \frac{3}{4} + 0 * \frac{1}{4} \approx 0.4774$
- Interpretation: the expected number of bits needed to transmit y if both the emitter and the receiver know the value of x (but before they are told x's value).

Information gain

Suppose I have to transmit y. How many bits on the average would it save me if both me and the receiver knew x?

Information gain

Suppose I have to transmit y. How many bits on the average would it save me if both me and the receiver knew x?

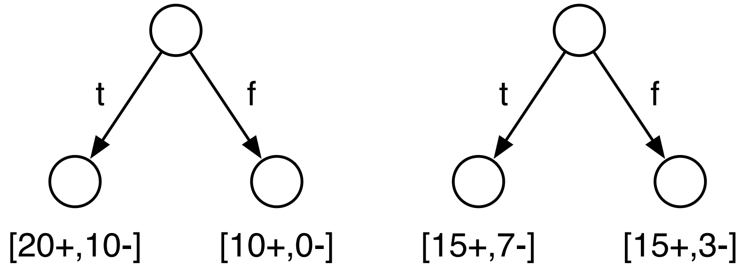
$$IG(y|x) = H(y) - H(y|x)$$

This is called **information gain**
Alternative interpretation: how much reduction in entropy do I get if I know x.

Information gain to determine best test

- Returning to decision tree construction, how do we use information theory?
- We choose, recursively at each interior node, the test that has highest information gain. (Equivalently, results in lowest conditional entropy.)
- If tests are binary:

$$\begin{aligned} IG(D, \text{Test}) &= H(D) - H(D|\text{Test}) \\ &= H(D) - \frac{|D_{\text{Test}}|}{|D|} H(D_{\text{Test}}) - \frac{|D_{\neg\text{Test}}|}{|D|} H(D_{\neg\text{Test}}) \end{aligned}$$



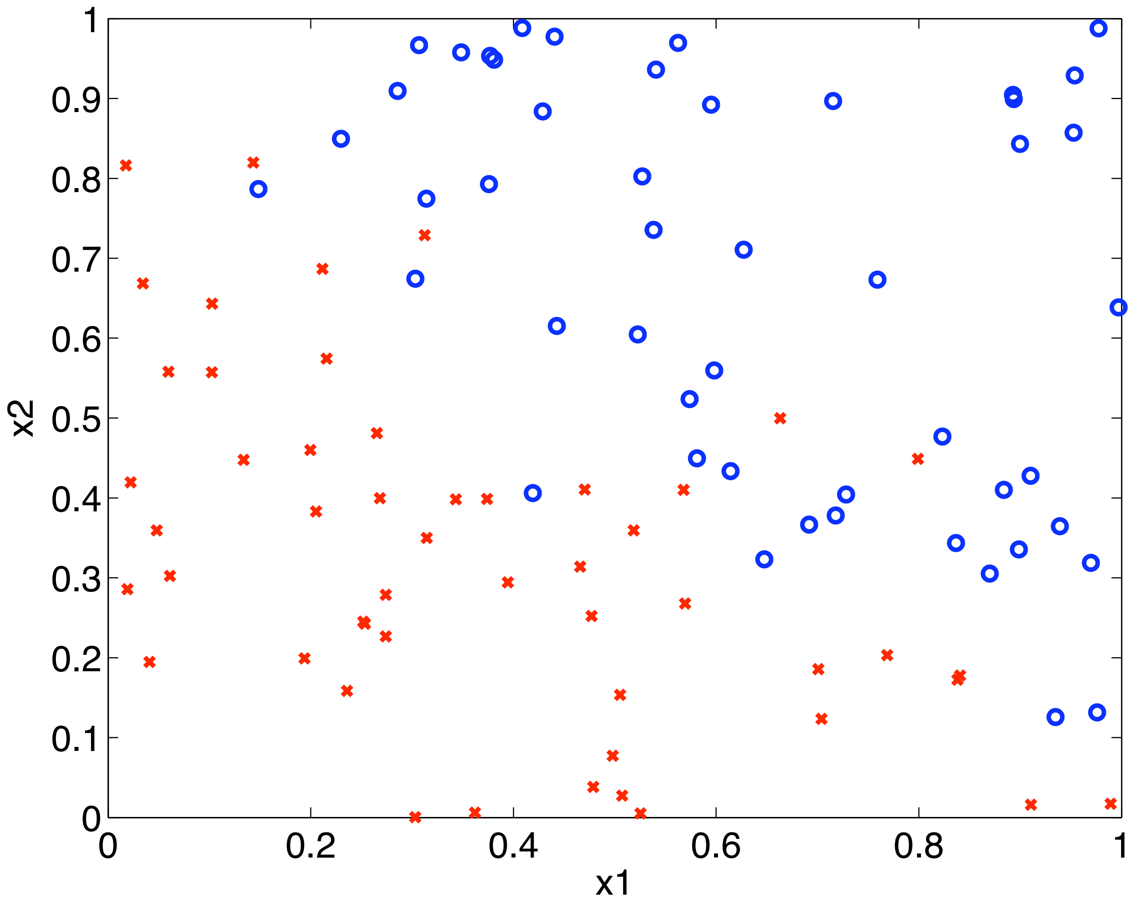
Check that in this case, Test1 wins.

Caveats on tests with multiple values

- If the outcome of a test is *not binary*, the number of possible values influences the information gain
- The more possible values, the higher the gain! (the more likely it is to form small, but pure partitions)
- C4.5 (the most popular decision tree construction algorithm in ML community) uses only binary tests:
  - Attribute=Value for discrete attributes
  - Attribute $<$ or $>$ Value for continuous attributes
- Other approaches consider smarter metrics (e.g. gain ratio), which account for the number of possible outcomes

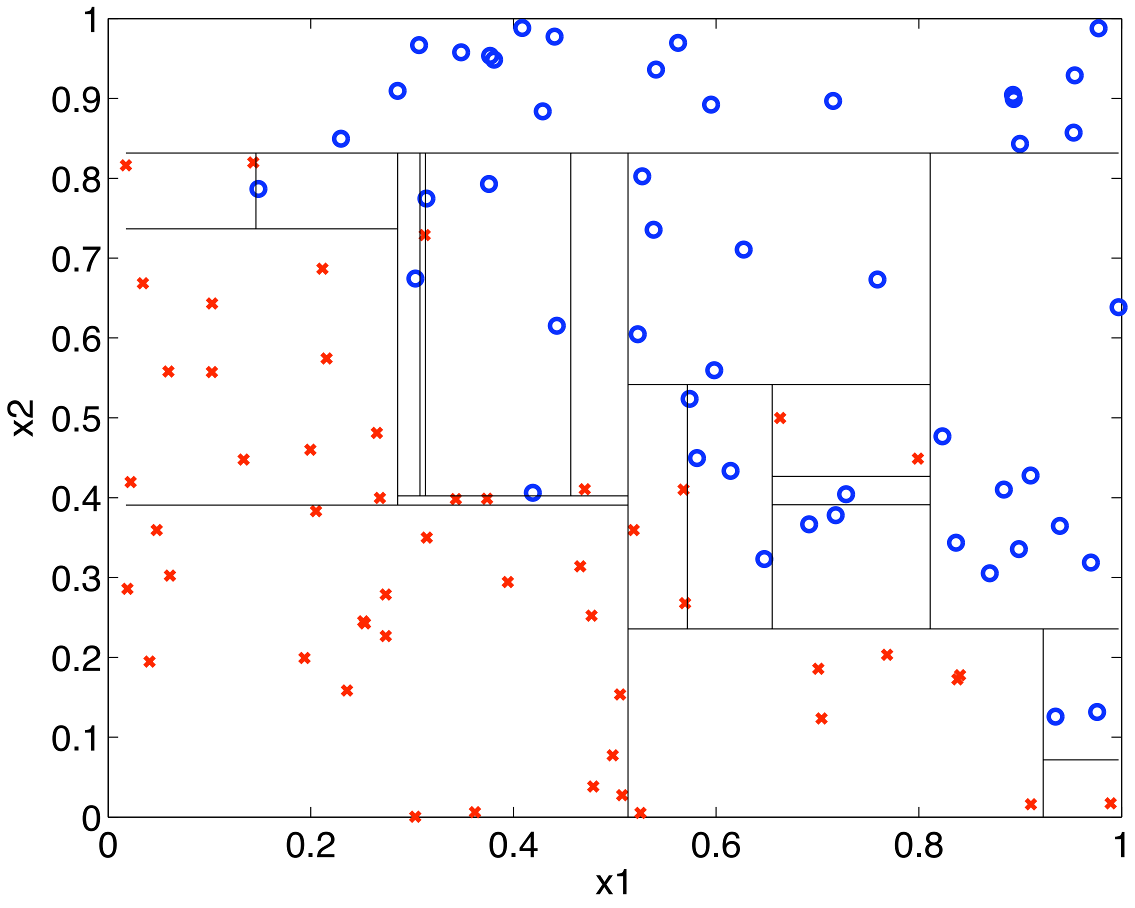
A complete (artificial) example

An artificial binary classification problem with two real-valued input features:



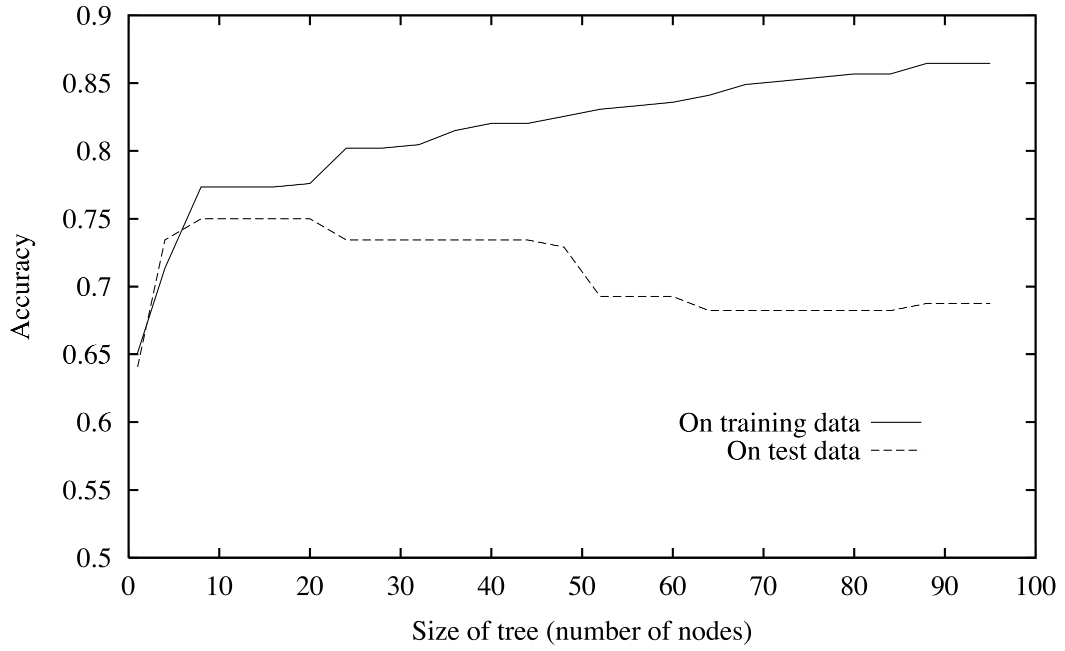
A complete (artificial) example (II)

The decision tree, graphically:



Overfitting in decision trees

- Remember, decision tree construction proceeds until all leaves are pure – all examples having the same y value.
- As the tree grows, the generalization perform can start to degrade, because the algorithm is finding **irrelevant** attributes / tests / outliers.



(Example from . . . ?)

Avoiding overfitting

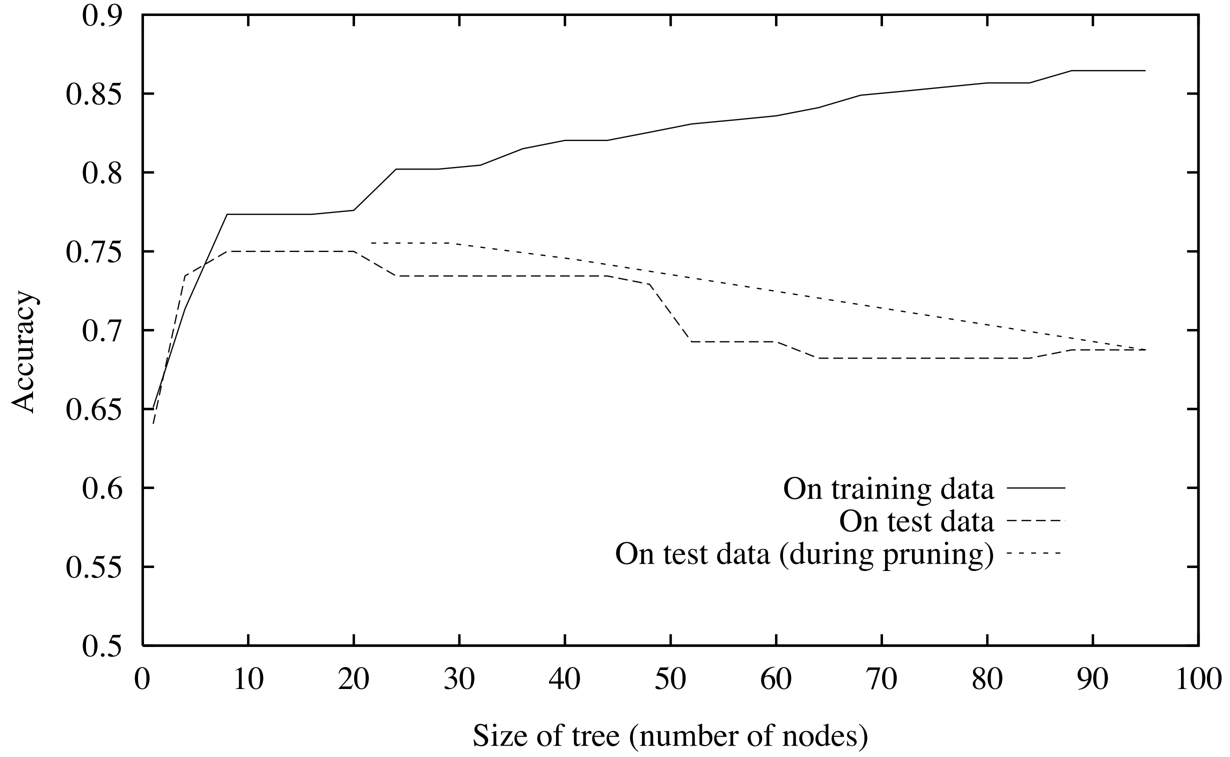
1. Stop growing the tree when further splitting the data does not yield a statistically significant improvement
2. Grow a full tree, then **prune** the tree, by eliminating nodes

The second approach has been more successful in practice
We will select the best tree, for now, by measuring performance on a separate validation data set.

Example: Reduced-error pruning

1. Split the “training data” into a training set and a validation set
2. Grow a large tree (e.g. until each leaf is pure)
3. For each node:
   (a) Evaluate the validation set accuracy of pruning the subtree rooted at the node
   (b) Greedily remove the node that most improves validation set accuracy, with its corresponding subtree
   (c) Replace the removed node by a leaf with the majority class of the corresponding examples.
4. Stop when pruning starts hurting the accuracy on the validation set.

Example: Effect of reduced-error pruning



Example: Rule post-pruning in C4.5

1. Convert the decision tree to rules
2. Prune each rule independently of the others, by removing preconditions such that the accuracy is improved
3. Sort final rules in order of estimated accuracy

Advantages:

- Can prune attributes higher up in the tree *differently on different paths*
- There is no need to reorganize the tree if pruning an attribute that is higher up
- Most of the time people want rules anyway, for readability

Variations: Gini impurity index

- An alternative to entropy for measuring impurity is the Gini index.
- For a distribution $P(y)$,

$$Gini(P) = 1 - \sum_{y} P(y)^2$$

- This criterion is used by the CART program (for Classification And Regression Trees) – the other major decision tree algorithm, besides C4.5

Variations: Constructing tests on the fly

- Suppose we are interested in more sophisticated tests at nodes, such a linear combinations of features:

$$3.1 \times \text{radius} - 1.9 \times \text{texture} \geq 1.3$$

- We can use a fast, simple classifier (such as LDA, Logistic classification) to determine a decision boundary, and us it as a test

Regression trees

- Like classification trees but for regression problems
- Tests can be the same as before
- At the leaves, instead of predicting the majority of the examples, we can predict the mean. . . or do something more complicated like a linear regression fit of the examples that get to the leaf. (It must be fast though!)
- Question: What is the equivalent of the class entropy for choosing the best test?

Decision (and regression) tree summary

- Fast learning algorithms (e.g. C4.5, CART)
- Standard learning algorithm is to:
  1. Construct the tree top down by greedily choosing the test which minimizes the conditional entropy of y
  2. Prune the tree, or corresponding rule set, based on a validation set to avoid overfitting
- Attributes may be discrete or continuous
- Scaling / normalization not needed, as we use no notion of "distance" between examples
- Provide a general representation of classification rules
- Easy to understand! Though. . .
  - Exact tree output may be sensitive to small changes in data
  - With many features, tests may not be meaningful
- In standard form, good for (nonlinear) piecewise axis-orthogonal decision boundaries – not good with smooth, curvilinear boundaries
- Good accuracy in practice – many applications!