Decision Trees

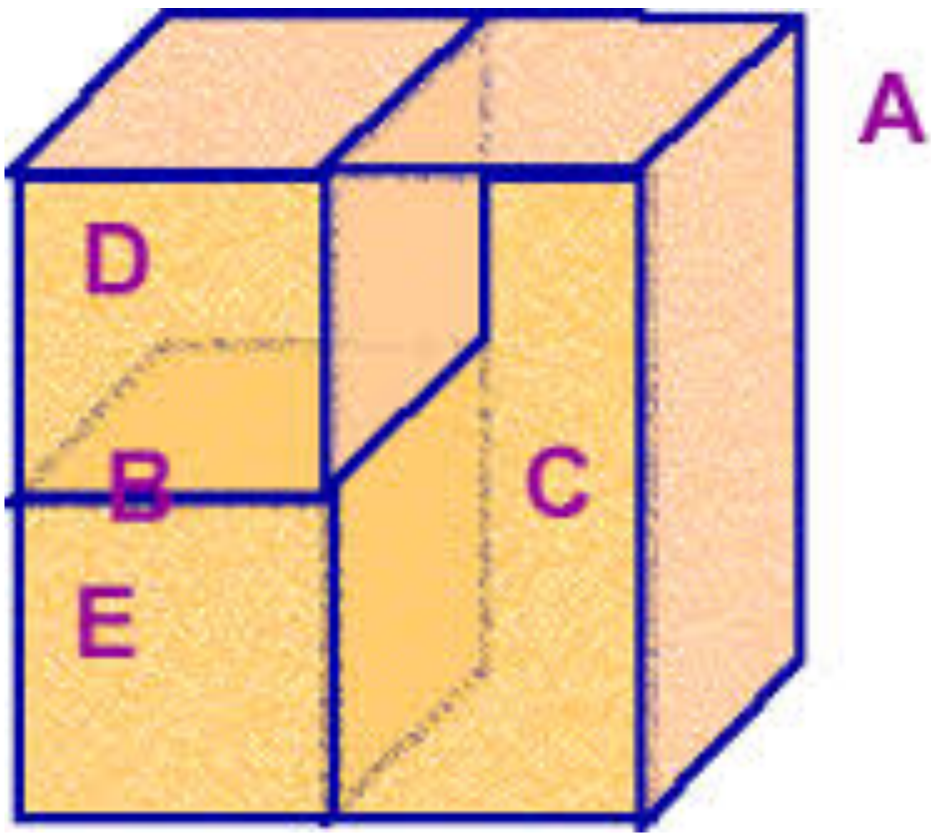
(slides from McGill COMP-551 course)

Slides: Joelle Pineau (*jpineau@cs.mcgill.ca*)

Back to machine learning: A quick recap

- Linear regression: Fit a linear function from input data to output.
- Linear classification: Find a **linear** hyper-plane separating classes.
- Many problems require more sophisticated models!

Richer partitioning of the input space



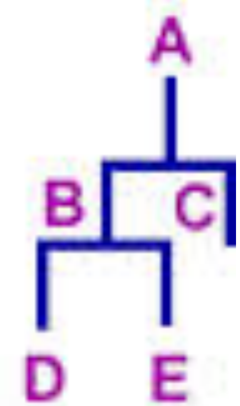


Image from *http://www.euclideanspace.com*

Example: Cancer Outcome Prediction

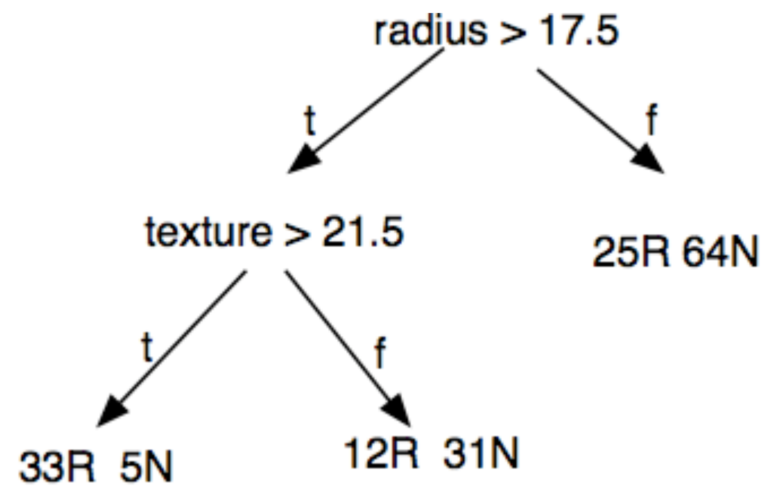
- Researchers computed 30 different features (or attributes) of the cancer cells' nuclei
  - Features relate to radius, texture, perimeter, smoothness, concavity, etc. of the nuclei
  - For each image, mean, standard error, and max of these properties across nuclei
- Vectors in the form ($(x_1, x_2, x_3, \ldots\ldots)$, y) = (**X** **,** y)
- Data set **D** in the form of a data table:

tumor size	texture	perimeter	...	outcome
18.02	27.6	117.5		N
17.99	10.38	122.8		N
20.29	14.34	135.1		R
...				

Decision tree example

- What does a node represent?
  - A partitioning of the input space.

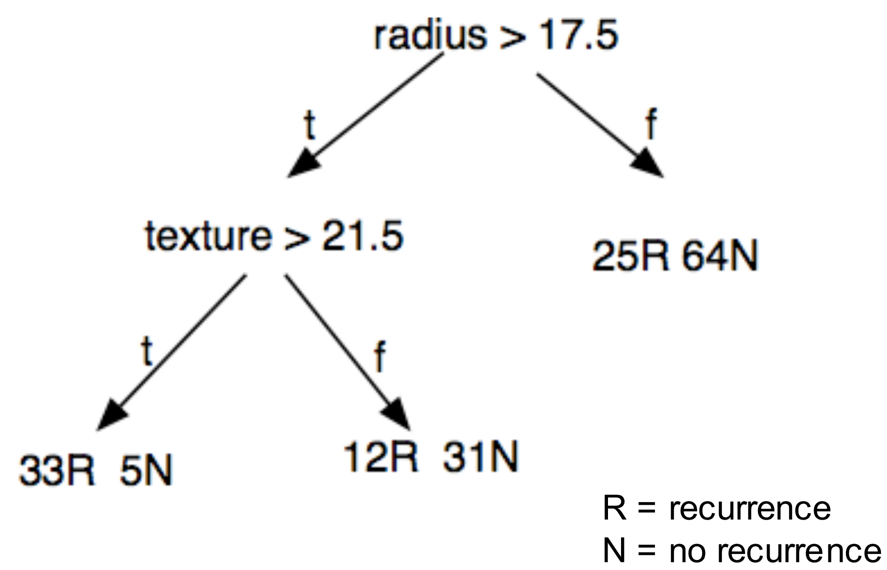
- Internal nodes are tests on the values of different features.
  - Don't need to be binary test.



R = recurrence
N = no recurrence

Decision tree example

- What does a node represent?
  - A partitioning of the input space.



- Internal nodes are tests on the values of different features.
  - Don't need to be binary test.

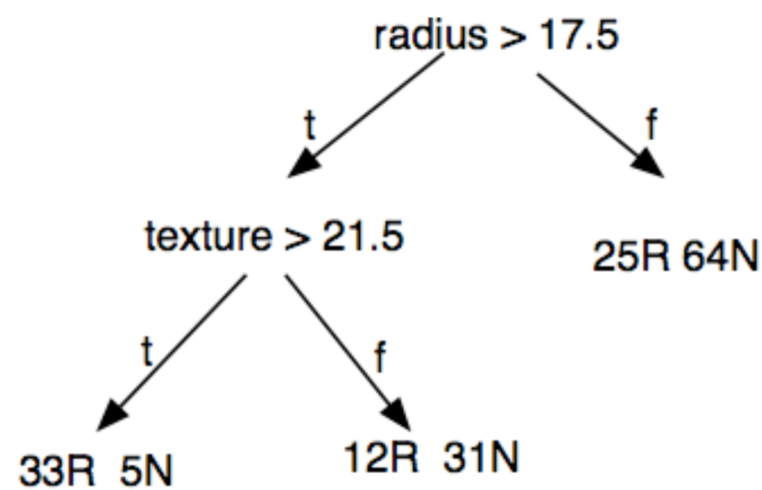
- Leaf nodes include the set of training examples that satisfy the tests along the branch.
  - Each training example falls in precisely one leaf.
  - Each leaf typical contains more than one example.

Using decision trees for classification

- Suppose we get a new instance:

  radius=18, texture=12, …

  How do we classify it?



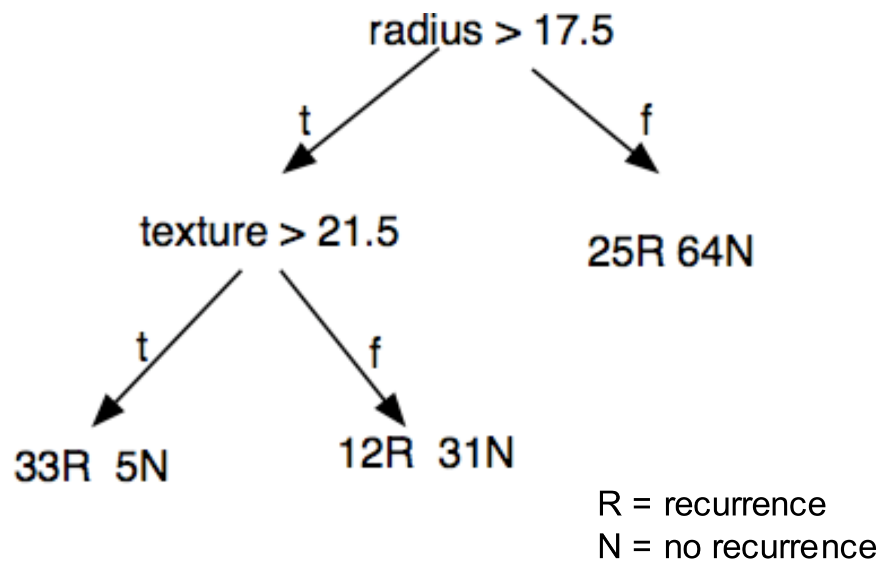
R = recurrence
N = no recurrence

Using decision trees for classification

- Suppose we get a new instance:

  radius=18, texture=12, …

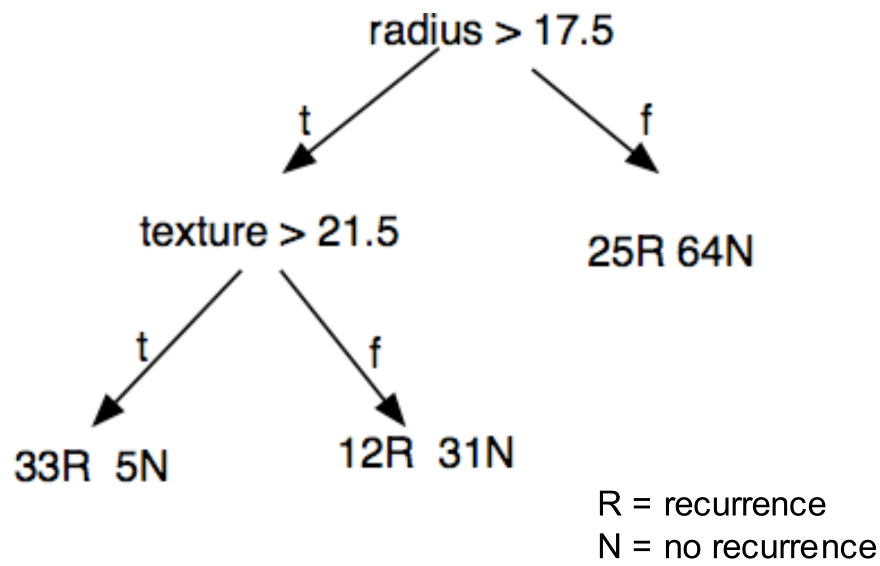
  How do we classify it?



- Simple procedure:
  - At every node, test the corresponding attribute
  - Follow the appropriate branch of the tree
  - At a leaf, either predict the class of the majority of the examples for that leaf, or sample from the probabilities of the two classes.

Interpreting decision trees

Can always convert a decision tree into equivalent set of if-then rules.

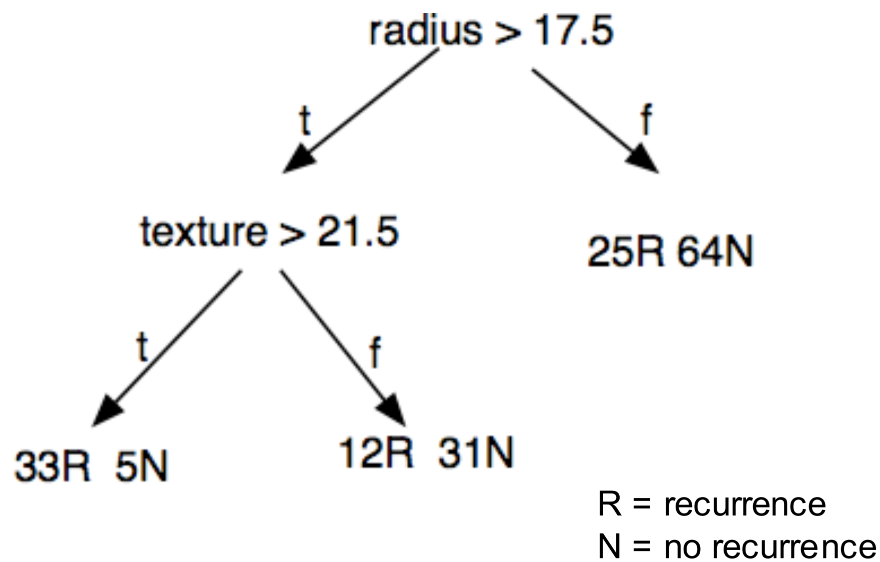


IF	THEN most likely class is
radius>17.5 AND texture>21.5	R
radius>17.5 AND texture≤21.5	N
radius≤17.5	N

Interpreting decision trees

Can always convert a decision tree into equivalent set of if-then rules.

Can also calculate an estimated probability of recurrence.



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

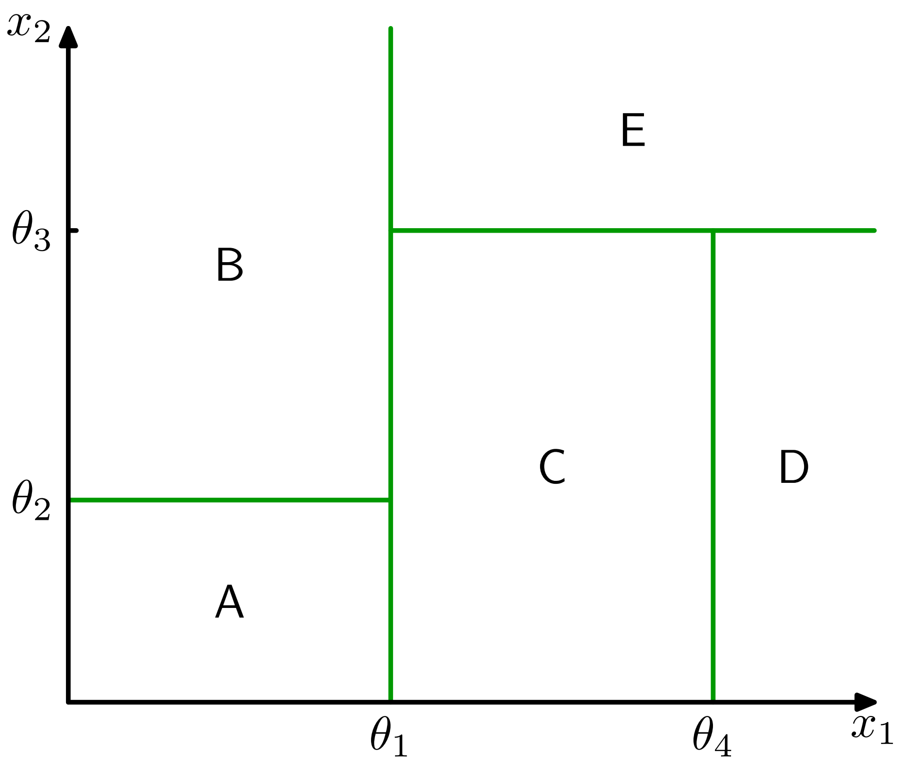
More Formally : Tests

- Each internal node contains a test.
  - Test typically depends on only on feature.
  - For discrete features, we typically branch on all possibilities
  - For real features, we typically branch on a threshold value

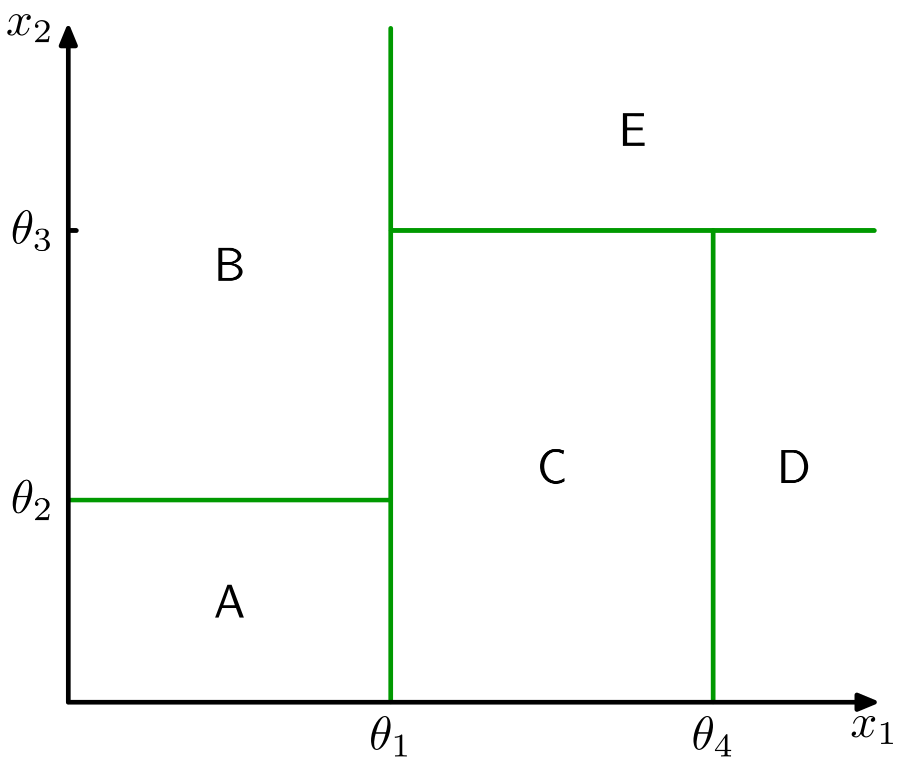
More Formally : Tests

- Each internal node contains a test.
  - Test typically depends on only on feature.
  - For discrete features, we typically branch on all possibilities
  - For real features, we typically branch on a threshold value
- Test returns a 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
- Learning = choosing tests at every node and shape of the tree.
  - A finite set of candidate tests is chosen before learning the tree.

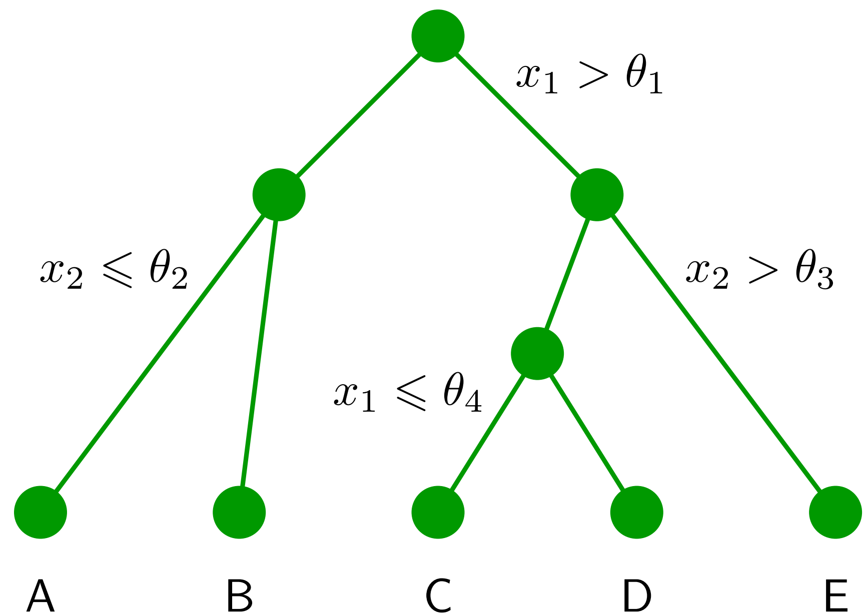
Example



Example



x_2
θ_3
θ_2
θ_1
θ_4
x_1
E
B
C
D
A



$x_1 > \theta_1$
$x_2 \leqslant \theta_2$
$x_2 > \theta_3$
$x_1 \leqslant \theta_4$
A
B
C
D
E

Expressivity of decision trees

What kind of function can be encoded via a decision tree?

Expressivity of decision trees

What kind of function can be encoded via a decision tree?

- Every Boolean Function can be **fully** expressed
  - Each entry in truth table could be one path (very inefficient!)
  - Most boolean functions can be encoded more compactly.

Expressivity of decision trees

What kind of function can be encoded via a decision tree?

- Every Boolean Function can be **fully** expressed
  - Each entry in truth table could be one path (very inefficient!)
  - Most boolean functions can be encoded more compactly.
- Some functions are harder to encode
  - Parity Function = Returns 1 iff an even number of inputs are 1
    - An exponentially big decision tree $O(2^M)$ would be needed
  - Majority Function = Returns 1 if more than half the inputs are 1

Expressivity of decision trees

What kind of function can be encoded via a decision tree?

- Every Boolean Function can be **fully** expressed
- Many other functions can be approximated by a Boolean function.
- With real-valued features, decision trees are good at problems in which the class label is constant in large connected axis-orthogonal regions of the input space.

Learning Decision trees

- We could enumerate all possible trees (assuming the number of possible tests is finite)
  - Each tree could be evaluated using the training set or, better yet, a test set
  - There are many possible trees!   We'd probably overfit the data.

Learning Decision trees

- We could enumerate all possible trees (assuming the number of possible tests is finite)
  - Each tree could be evaluated using the training set or, better yet, a test set
  - There are many possible trees! We'd probably overfit the data.

- Usually, decision trees are constructed in two phases:
  1. A 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.

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.

   Else

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. Recursively repeat steps 1 - 3 on each subset of the training data.

How do we pick the **best test**?

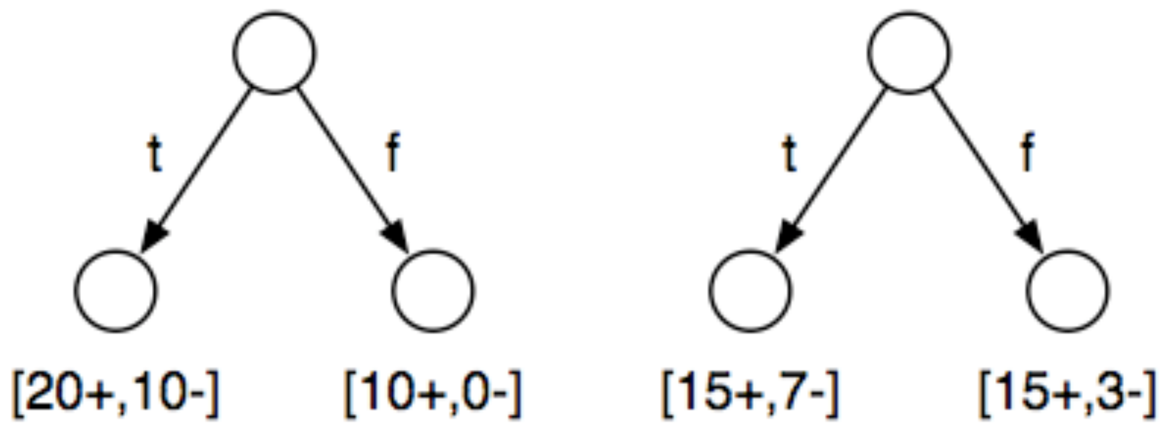
What is a good Test?

- The test should provide **information** about the class label.

  E.g. You are given 40 examples: 30 positive, 10 negative

  Consider two tests that would split the examples as follows:

  Which is best?



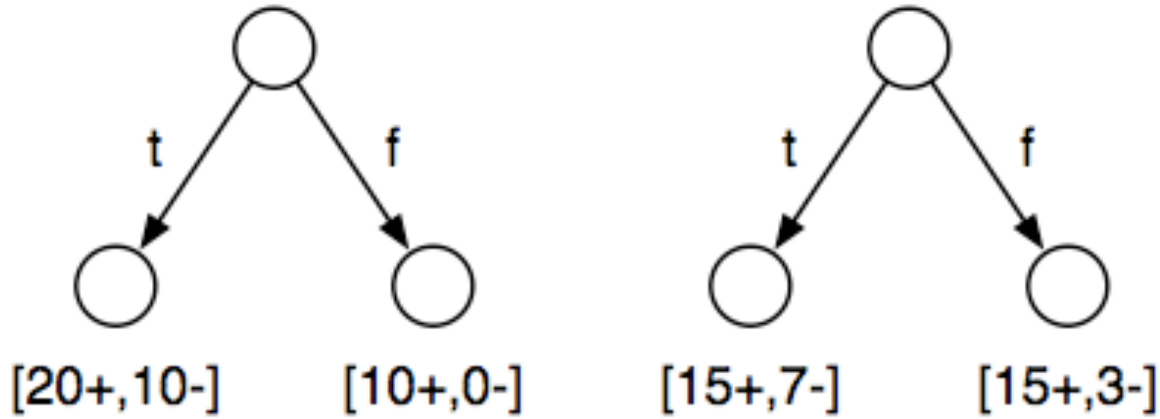
What is a good Test?

- The test should provide **information** about the class label.

  E.g. You are given 40 examples: 30 positive, 10 negative

  Consider two tests that would split the examples as follows:

  Which is best?



- Intuitively, we prefer an attribute that separates the training instances as well as possible. How do we quantify this (mathematically)?

What is information?

- Consider three cases:
  - You are about to observe the outcome of a dice roll
  - You are about to observe the outcome of a coin flip
  - You are about to observe the outcome of a biased coin flip

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

Information content

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

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

Information content

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

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

- You can also think of information as the amount of "surprise" in the outcome (e.g., consider $P(E) = 1$, then $I(E) \approx 0$)

  E.g.

  - fair coin flip provides $\log_2 2 = 1$ bit of information
  - fair dice roll provides $\log_2 6 \approx 2.58$ bits of information

Entropy

- Given 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** we expect from the output of *S*?

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

H(S) is the **entropy** of *S.*

- Note that this depends only on the probability distribution, and not on the actual alphabet. So we can write *H(P).*

Entropy

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

- Several ways to think about entropy:
  - 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.

Binary Classification

- We try to classify sample data using a decision tree.
- Suppose we have *p* positive samples and *n* negative samples.
- What is the entropy of this data set?

$$H(D) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

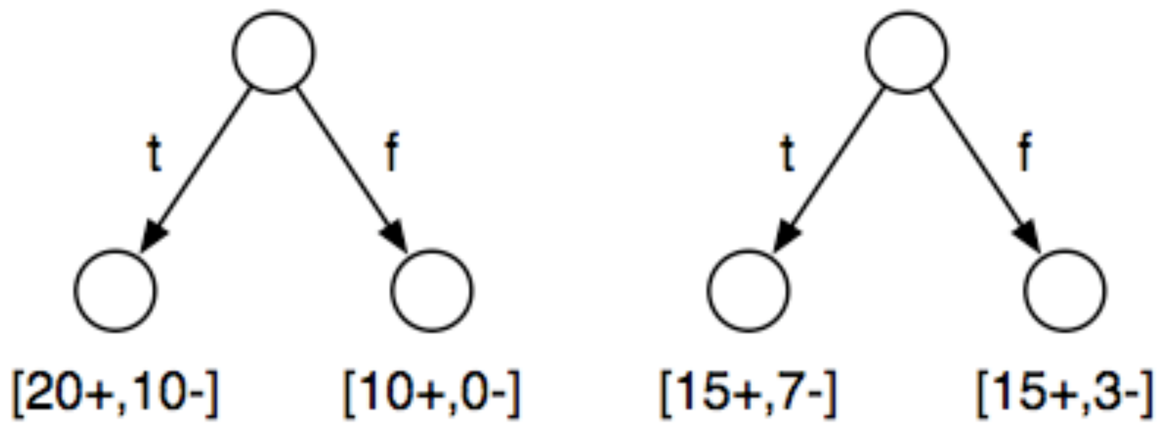
What is a good Test?

- The test should provide **information** about the class label.

  E.g. You are given 40 examples: 30 positive, 10 negative

  Consider two tests that would split the examples as follows:

  Which is best?



$H(D) = -(3/4)\log_2(3/4)-(1/4)\log_2(1/4) = 0.811$

Conditional entropy

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

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

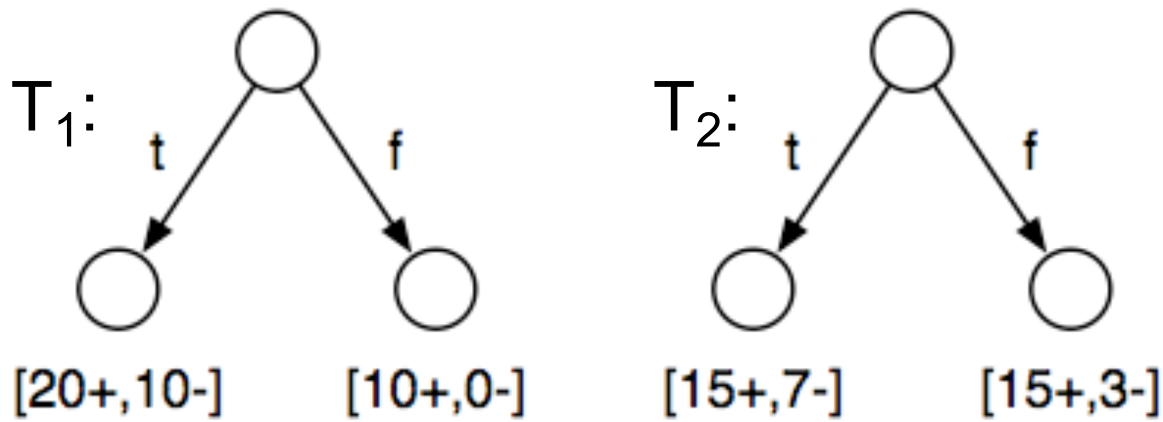
- **Interpretation**: the expected number of bits needed to transmit y if both the emitter and receiver know the possible values of *x* (but before they are told *x*'s specific value.)

What is a good Test?

- The test should provide **information** about the class label.

  E.g. You are given 40 examples: 30 positive, 10 negative

  Consider two tests that would split the examples as follows: Which is best?

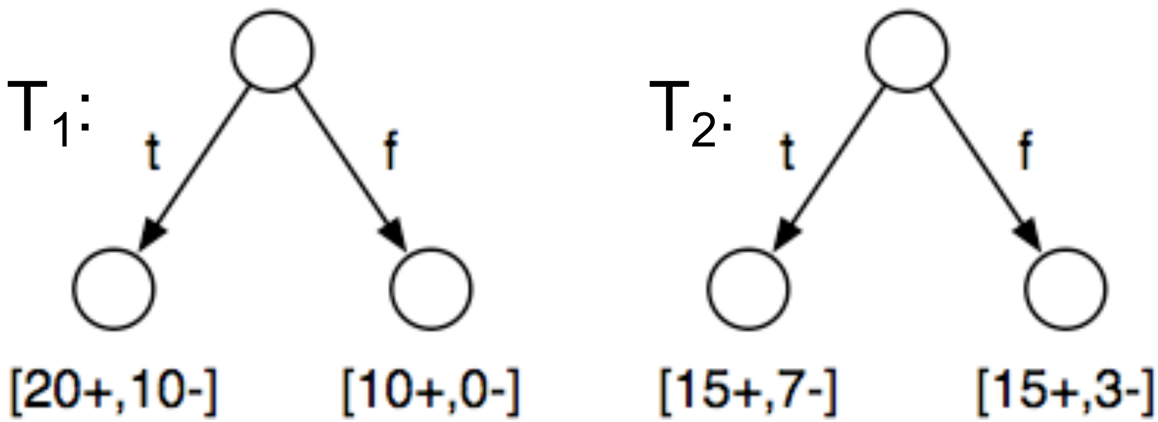


What is a good Test?

- The test should provide **information** about the class label.

  E.g. You are given 40 examples: 30 positive, 10 negative

  Consider two tests that would split the examples as follows: Which is best?



$H(D) = -(3/4)\log_2(3/4)-(1/4)\log_2(1/4) = 0.811$

$H(D|T_1) = (30/40)[-(20/30)\log_2(20/30)-(10/30)\log_2(10/30)]+(10/40)[0] = 0.688$

$H(D|T_2) = (22/40)[-(15/22)\log_2(15/22)-(7/22)\log_2(7/22)]$

$\quad + (18/40)[-(15/18)\log_2(15/18)-(3/18)\log_2(3/18)] = 0.788$

Information gain

- The reduction in entropy that would be obtained by knowing *x* :

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

- Equivalently, suppose one has to transmit y. How many bits (on average) would it save if both the transmitter and emitter knew *x*?

Recursive learning of decision trees

Given a set of labeled training instances:

1. If all training instances have the same class, create a leaf with that class label and exit.

Else:

2. Pick the **best test** to split the data on.
3. Split the training set according to the value of the test outcome.
4. Recursively repeat steps 1 to 3 on each subset of training data.

How do we pick the **best test**?

Recursive learning of decision trees

Given a set of labeled training instances:

1. If all training instances have the same class, create a leaf with that class label and exit.

Else:

2. Pick the **best test** to split the data on.
3. Split the training set according to the value of the test outcome.
4. Recursively repeat steps 1 to 3 on each subset of training data.

How do we pick the **best test**?

For **classification**: choose the test with highest information gain.

For **regression**: choose the test with lowest mean-squared error.

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!
  - Nonetheless, the attribute could be irrelevant
- Could transform attribute in one (or many) binary attributes.

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!
  - Nonetheless, the attribute could be irrelevant
- Could transform attribute in one (or many) binary attributes.

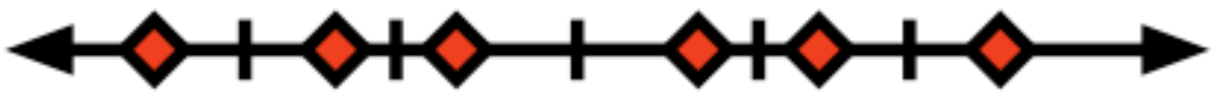
- C4.5 (the most popular decision tree construction algorithm in ML community) uses only binary tests:
  - Attribute =Value (discrete) or Attribute<Value (continuous)
- Other approaches consider smarter metrics which account for the number of possible outcomes.

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 $x_j > \tau$?

Tests for real-valued features

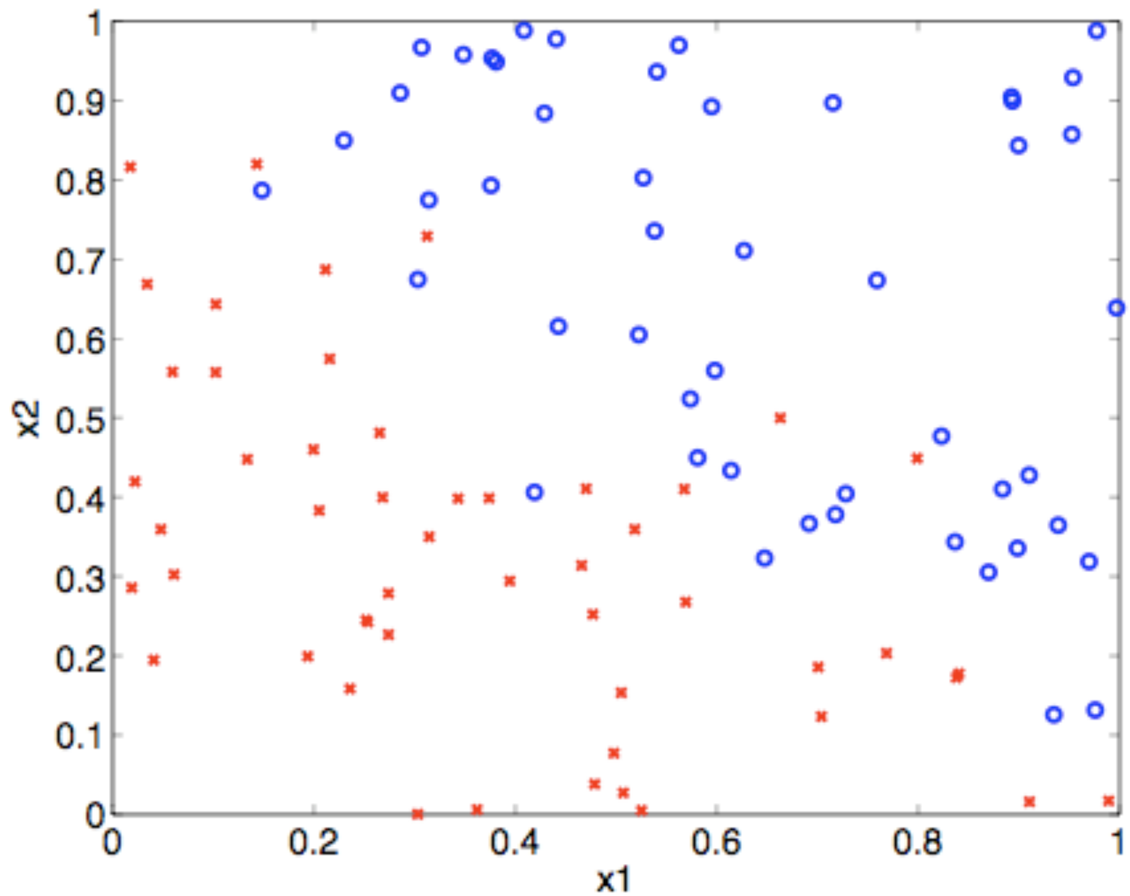
- Suppose feature j is real-valued
- How do we choose a finite set of possible thresholds, for tests of the form $x_j > \tau$?
- Choose midpoints of the observed data values, $x_{1,j}, \ldots, x_{m,j}$



- Choose midpoints of data values with different y values

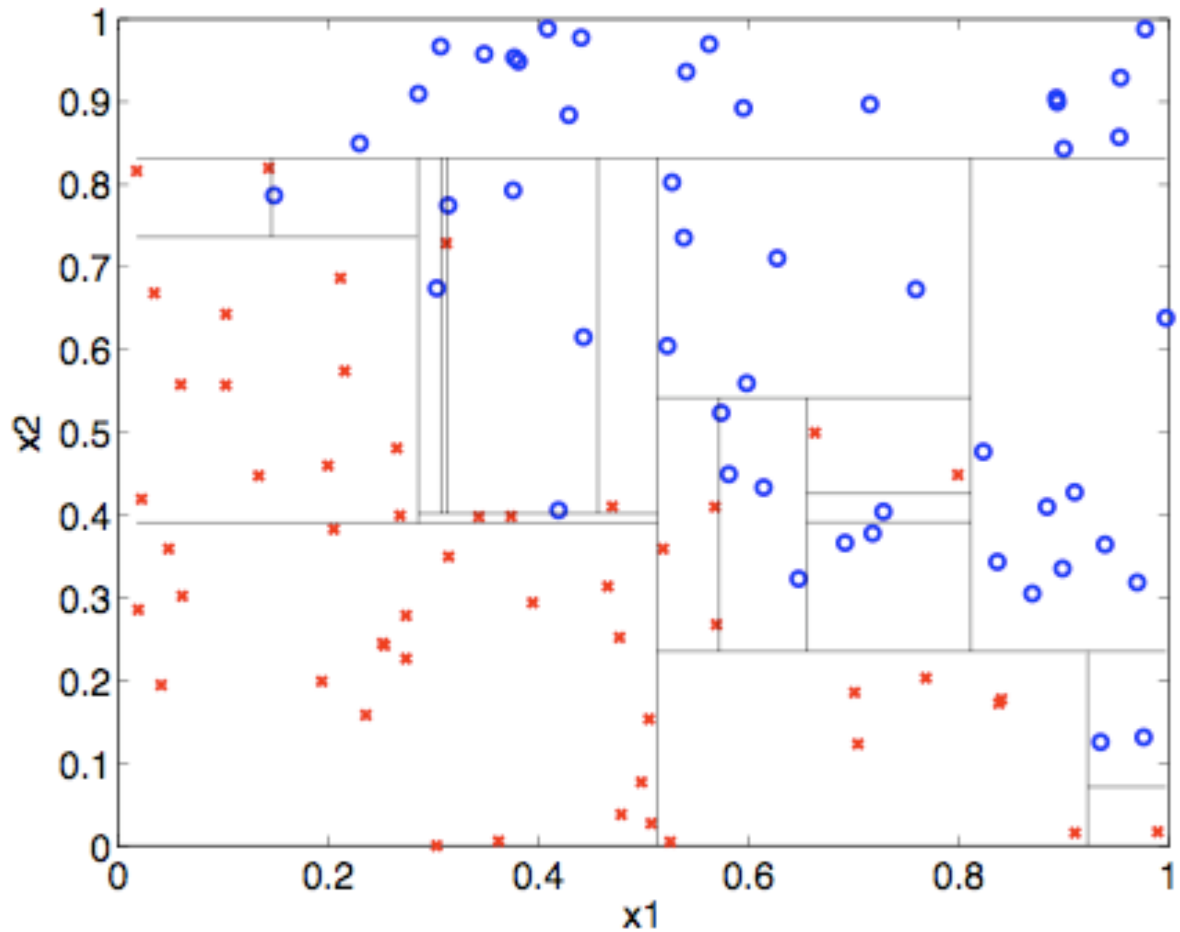
A complete (artificial) example

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



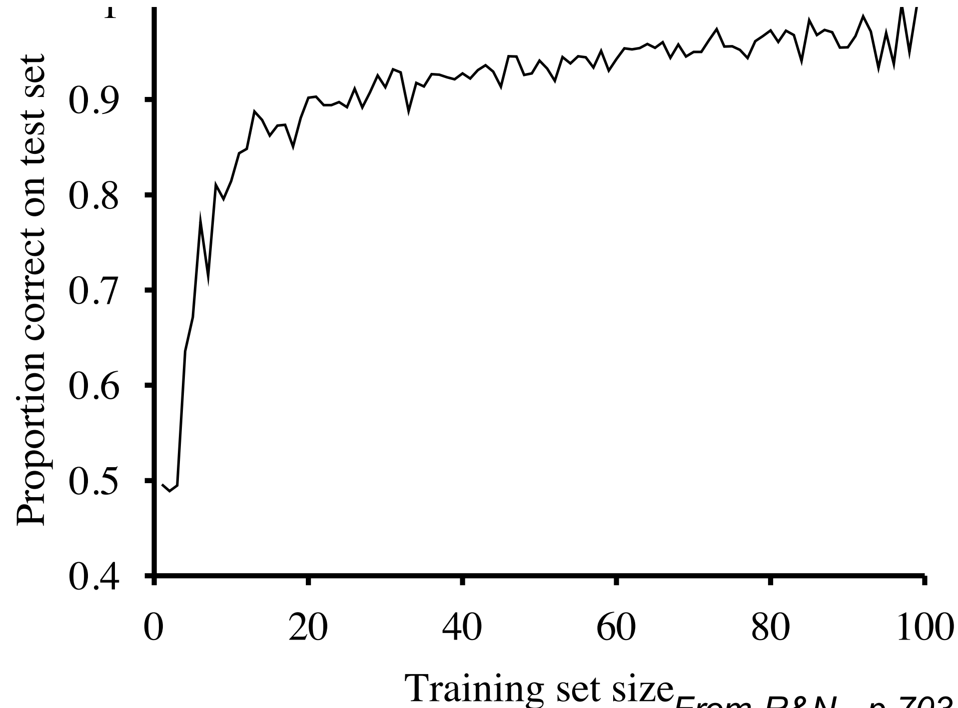
A complete (artificial) example

- The decision tree, graphically:



Assessing Performance

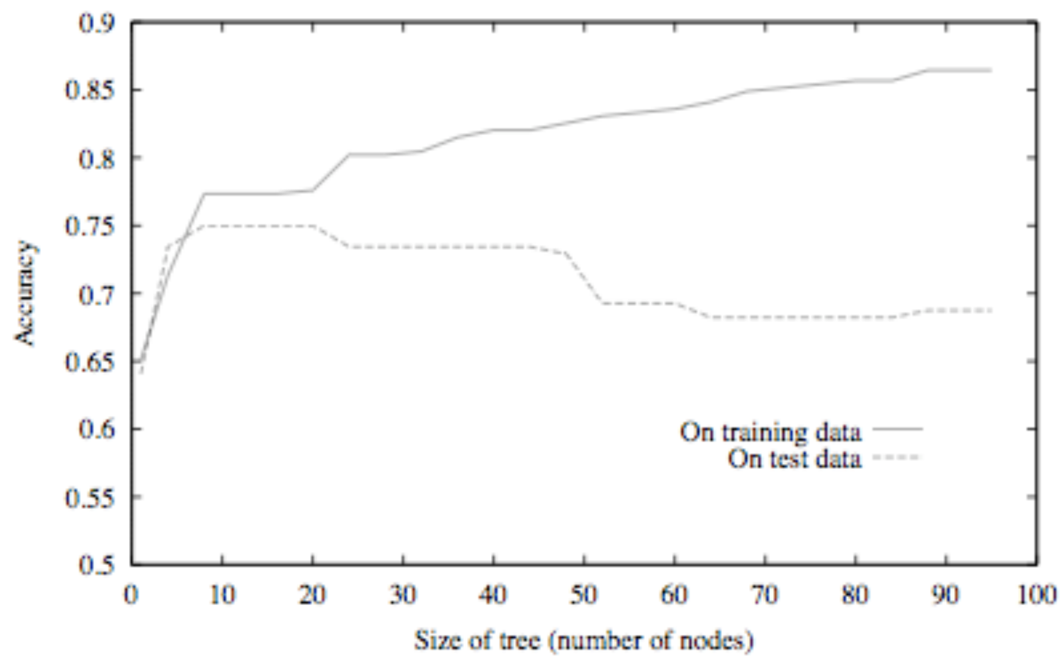
- Split data into a **training set** and a **validation set**
- Apply learning algorithm to training set.
- Measure error with the validation set.



From R&N , p.703

Overfitting in decision trees

- Decision tree construction proceeds until all leaves are "pure" , i.e. all examples are from the same class.
- As the tree grows, the generalization performance can start to degrade, because the algorithm is including irrelevant attributes/tests/outliers.
- How can we avoid this?



Avoiding Overfitting

- Objective: Remove some nodes to get better generalization.

Avoiding Overfitting

- Objective: Remove some nodes to get better generalization.
- Early stopping: Stop growing the tree when further splitting the data does not improve information gain of the validation set.
- Post pruning: Grow a full tree, then prune the tree by eliminating lower nodes that have low information gain on the validation set.

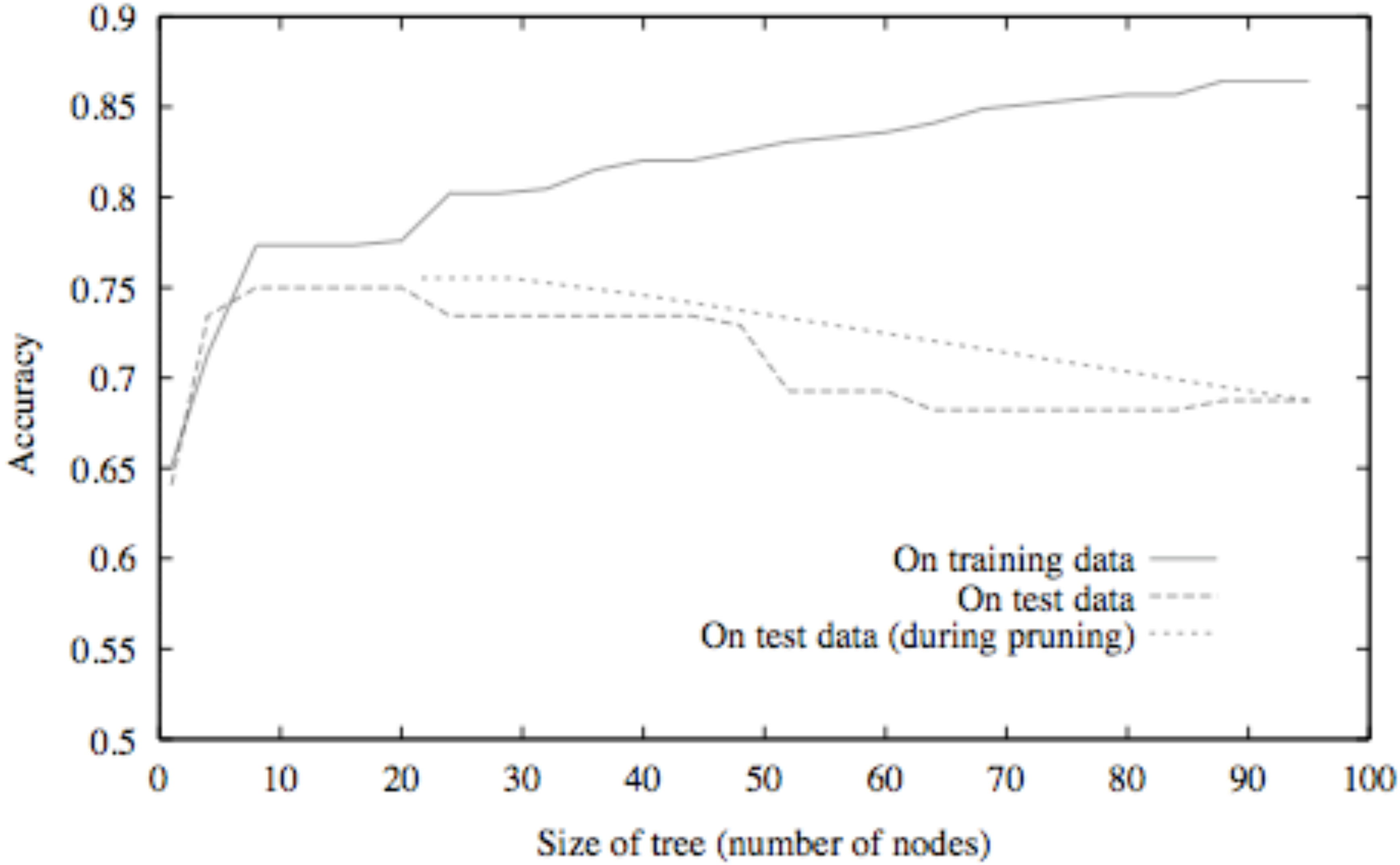
Avoiding Overfitting

- Objective: Remove some nodes to get better generalization.
- Early stopping: Stop growing the tree when further splitting the data does not improve information gain of the validation set.
- Post pruning: Grow a full tree, then prune the tree by eliminating lower nodes that have low information gain on the validation set.
- In general, post pruning is better. It allows you to deal with cases where a single attribute is not informative, but a combination of attributes is informative.

Reduced-error pruning (aka post pruning)

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

Example: Effect of reduced-error pruning



Advantages of decision trees

- Provide a general representation of classification rules
  - Scaling / normalization not needed, as we use no notion of "distance" between examples.
- The learned function, $y = h(x)$, is easy to interpret.
- Fast learning algorithms (e.g. C4.5, CART).
- Good accuracy in practice – many applications in industry!

Limitations

- Sensitivity:
  - Exact tree output may be sensitive to small changes in data
  - With many features, tests may not be meaningful
- Good for learning (nonlinear) piecewise axis-orthogonal decision boundaries, but not for learning functions with smooth, curvilinear boundaries.
  - Some extensions use non-linear tests in internal nodes.

What you should know

- How to use a decision tree to classify a new example.
- How to build a decision tree using an information-theoretic approach.
- How to detect (and fix) overfitting in decision trees.
- How to handle both discrete and continuous attributes and outputs.