CS 383 – Machine Learning

Decision Trees

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

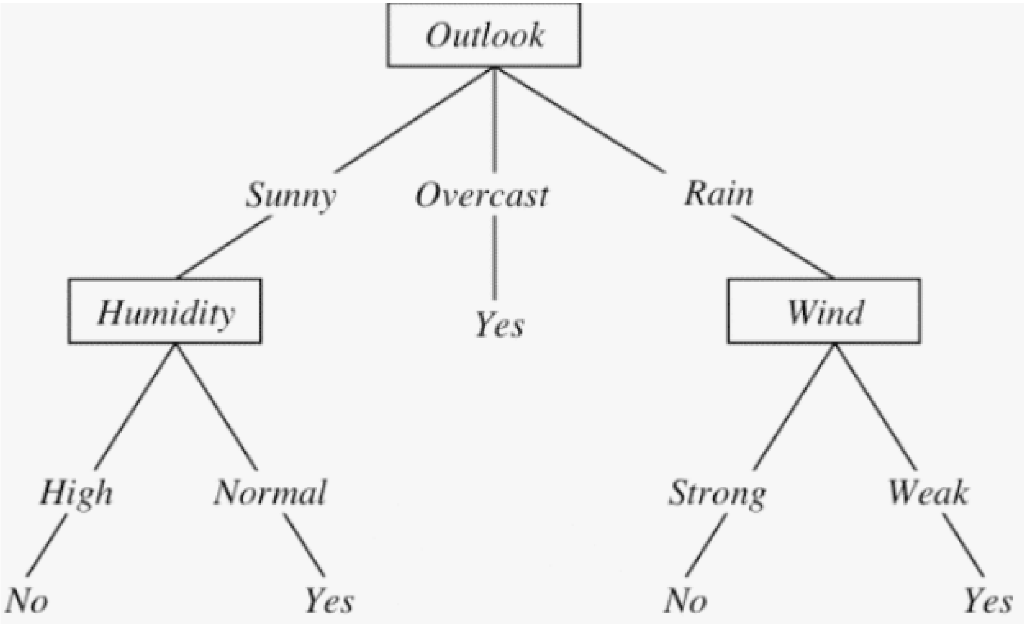
Objectives

- Decision Trees

Decision Trees

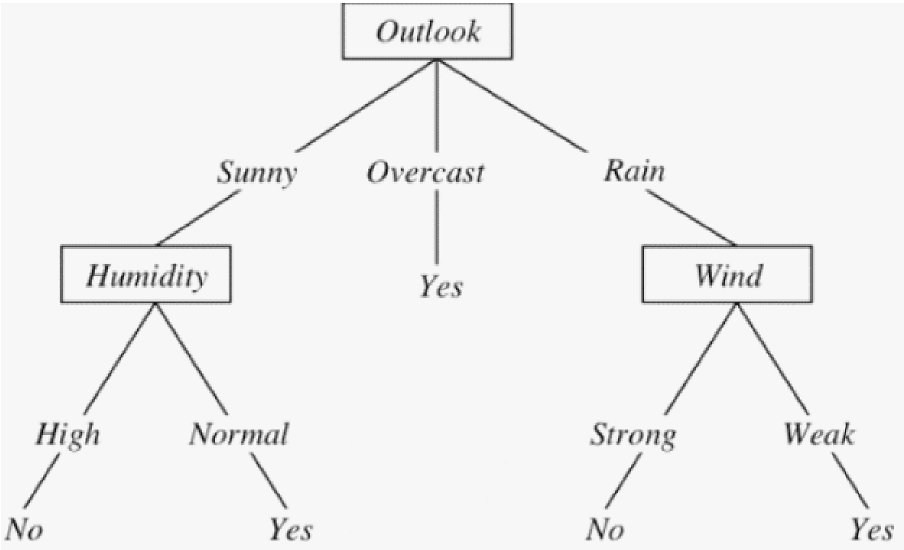
Decision Trees

- Building Decision Trees: Hierarchical and Recursive partitioning of the feature space
- Example: Play tennis?



Decision Trees

- Representation
  - Each internal node tests an attribute
  - Each branch is an attribute value
  - Each leaf assigns a classification



Consistent Decision Trees

- If we have D binary features and a binary classification system then there are 2^{2^D} decision trees
  - Yikes
- If our data is noiseless and without any randomness, then there is at least one trivially **consistent** decision tree for the data set
  - Fits the data perfectly
- In fact there's many potential ones
- But we'd prefer to find a *compact* one.
  - That is one with a minimal height.

Example

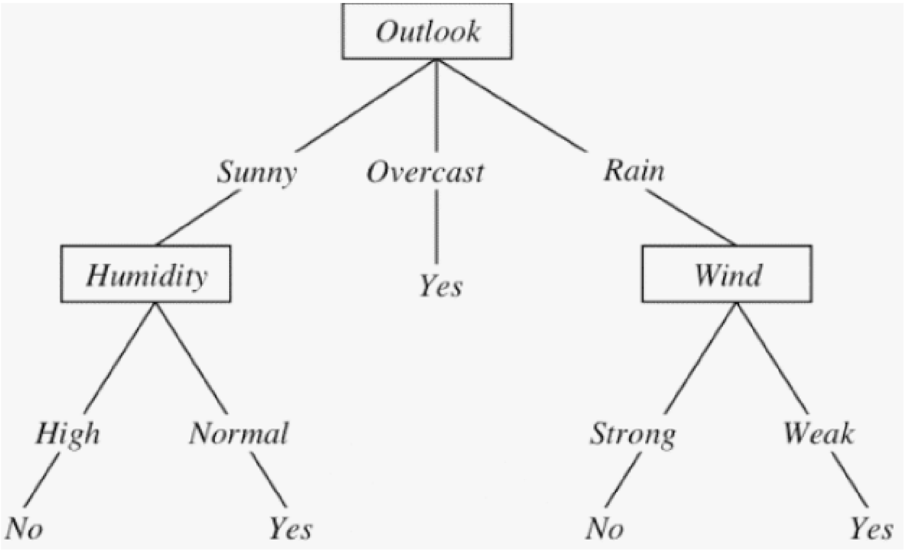
- Example: Situations for which I will/won't play tennis
- **Exercise:** Let's (painfully) build a consistent decision tree from these examples
  - There's a lot of different possible ones

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes

Example: Consistent Tree

- Here's one!

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes



Decision Tree Learning

- In practice it may be impossible to build a consistent decision tree
  - There may be noise
  - And/or there may be randomness
  - Or we don’t have enough information/features.
- Ultimately we’d like to find a small (compact) tree consistent with as many training examples as possible.

Decision Tree Learning

- Idea: (recursively) choose "most significant" attribute/feature as root of (sub)tree
  - A greedy algorithm

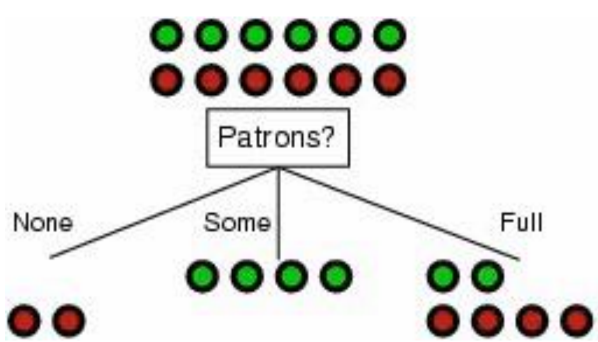
function DTL(*examples*, *attributes*, *default*) **returns** a decision tree

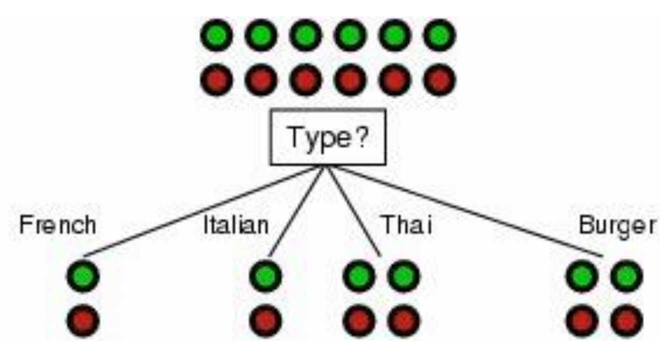
    if *examples* is empty **then return** *default*
    else if all *examples* have the same classification **then return** the classification
    else if *attributes* is empty **then return** MODE(*examples*)
    else
        best ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        tree ← a new decision tree with root test *best*
        for each value v_i of *best* **do**
            $examples_i$ ← {elements of *examples* with $best = v_i$}
            subtree ← DTL($examples_i$, *attributes* − *best*, MODE(*examples*))
            add a branch to *tree* with label v_i and subtree *subtree*
        return *tree*

Choosing an Attribute

- Idea: A good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”
- Which choice is better? (Patrons? Or Type?)

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T





Choosing an Attribute

- While there's many ideas on how to choose the next attribute, let's use something we already talked about!
- Information gain!

Information Gain (again)

- Given probability of events $v_1, \dots, v_n$ as $P(v_1), \dots, P(v_n)$ we can compute the entropy as

$$H\big(P(v_1), \dots, P(v_n)\big) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

- Information Gain (IG) or reduction in entropy based on attributed A:

$$IG(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- Where $remainder(A)$ is the weighted average entropy after splitting on attribute A

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} H\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

Example: Information gain

- Overall Entropy:
  - $p = n = 6$
  - $H(6/12, 6/12) = 1$
- Consider the attributes Patrons and Type (really we should look at all the attributes)
- Patrons
  - 2 none instances
    - 0 true wait
    - 2 false wait
  - 4 some instances
    - 4 true wait
    - 0 false wait
  - 6 full instances
    - 2 true wait
    - 4 false wait
  - $IG(Patrons) = 1 - \left[\frac{2}{12} H\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} H\left(\frac{4}{4}, \frac{0}{4}\right) + \frac{6}{12} H\left(\frac{2}{6}, \frac{4}{6}\right)\right] = 0.54$

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example: Information gain

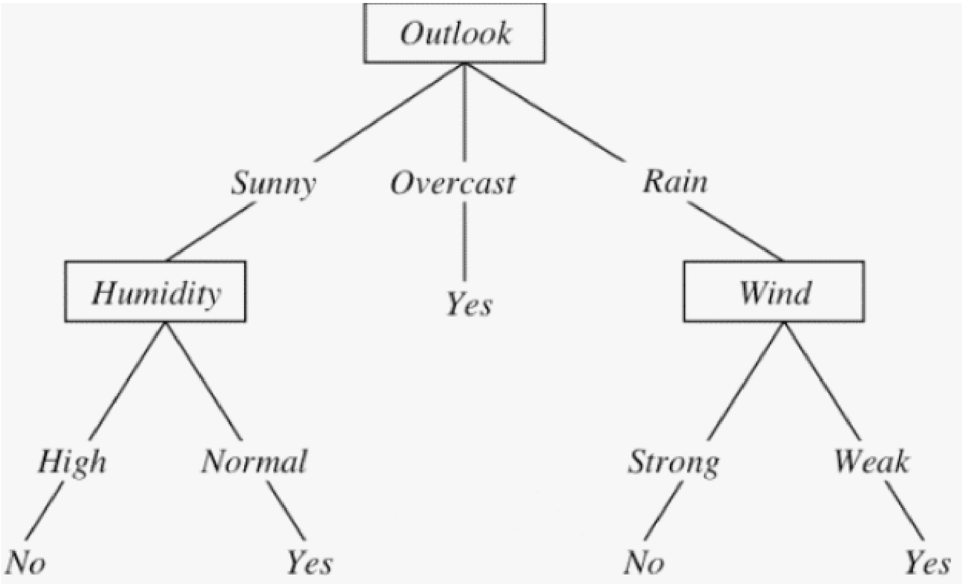
- Patrons
  - $IG(Patrons) = 1 - \left[\frac{2}{12}H\left(\frac{0}{2},\frac{2}{2}\right) + \frac{4}{12}H\left(\frac{4}{4},\frac{0}{4}\right) + \frac{6}{12}H\left(\frac{2}{6},\frac{4}{6}\right)\right] = 0.54$
- Type
  - $IG(Type) = 1 - \left[\frac{2}{12}H\left(\frac{1}{2},\frac{1}{2}\right) + \frac{4}{12}H\left(\frac{2}{4},\frac{2}{4}\right) + \frac{4}{12}H\left(\frac{2}{4},\frac{2}{4}\right) + \frac{2}{12}H\left(\frac{1}{2},\frac{1}{2}\right)\right] = 0$
- So which should we split on?

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example

- Building a tree completely using information gain is called the ID3 decision tree algorithm

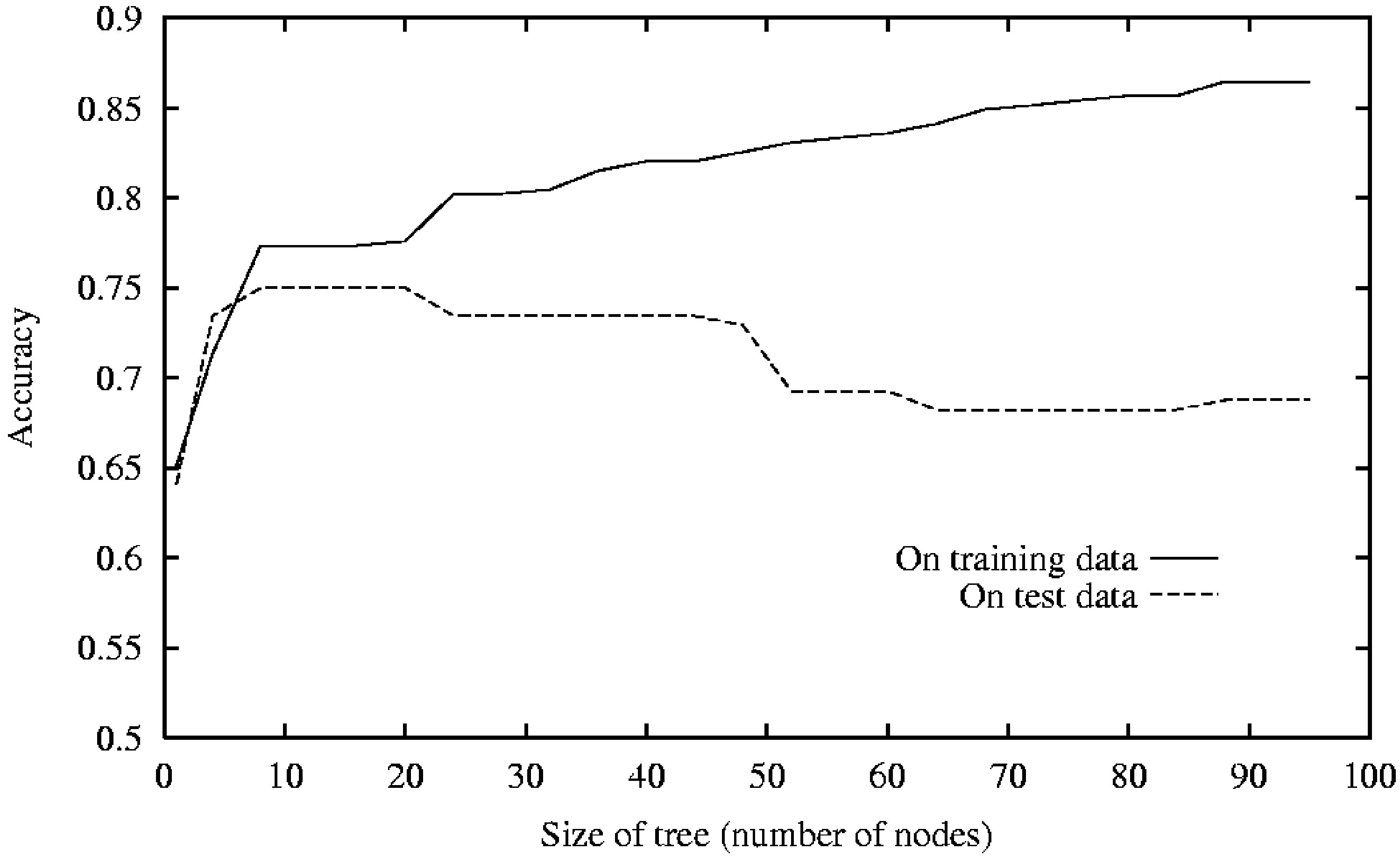
Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



Overfitting

- What's the problem with fitting our data as closely as possible?
  - We may overfit the data!
- There are two different approaches to dealing with this
  1. Stop growing based on some condition
  2. Grow "fully" then "prune" (again based on some condition)
- For each of these we can make our decision
  - Based on some statistic
  - Based on error with a new set (validation set)

Overfitting



http://jmvidal.cse.sc.edu/talks/decisiontrees/allslides.html

Statistical Approaches

- Statistics-Based **Growth**
  - Maybe you already know the ideal "height" (based on other stuff you've done). So just grow to that height

OR

  - You can set a threshold on information gain
    - When it comes time to split, only split if best IG is above some threshold

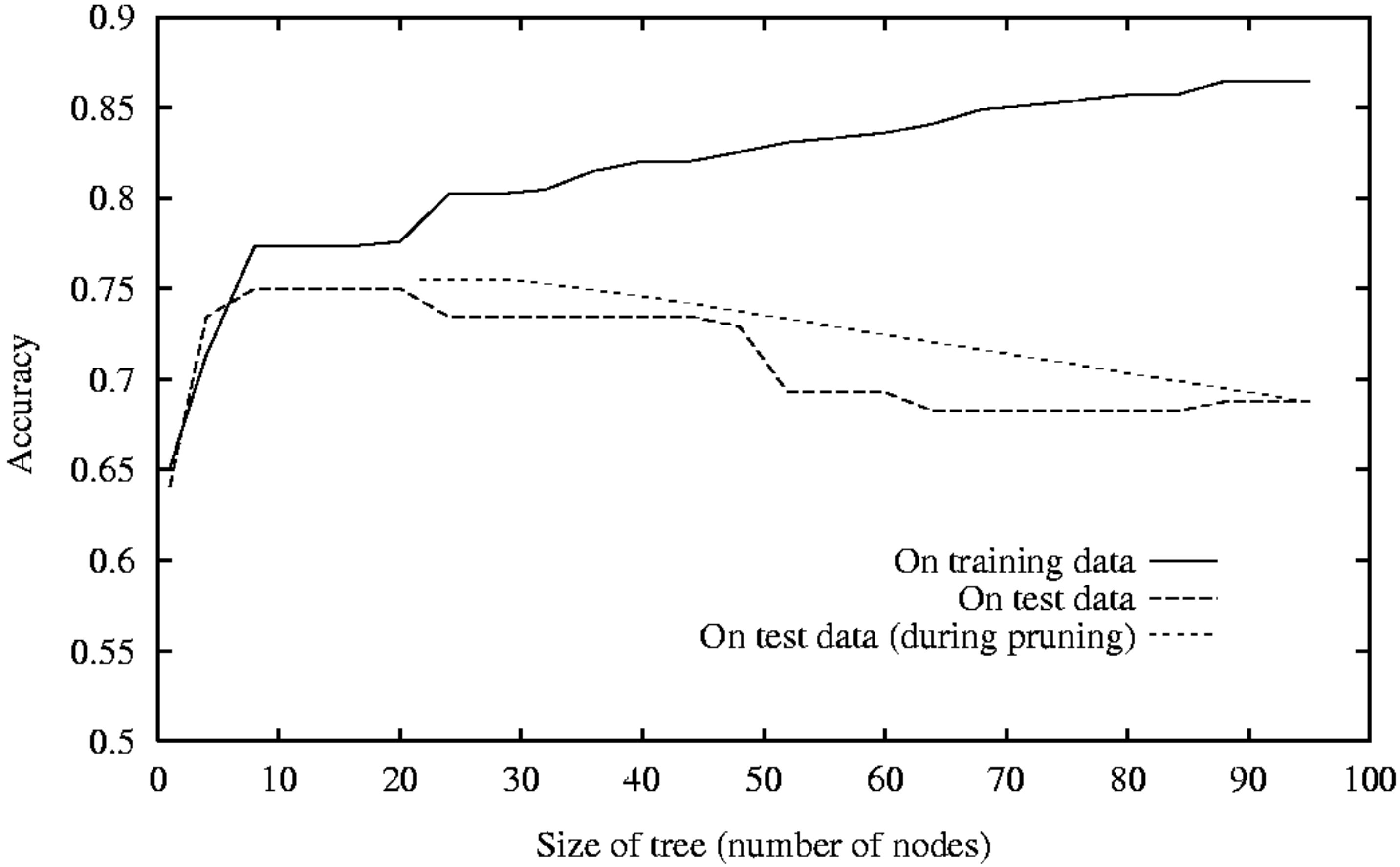
Statistical Approaches

- Statistics-Based **Pruning**
  - After having fully grown tree, find the node that has the lowest (or largest) statistic
  - If it’s below (or above) some threshold make it a leaf with it’s class equal to the mode of the labels passing through it
  - Update statistics in tree
  - Continue until no split nodes meet criteria

Reduced Error Approaches

- Split data into training, validation, and testing
- Reduced Error Growing
  - When you look to split a node based on training data, evaluating after using *validation set*
  - If things get worse, stop
- Reduced Error Pruning
  - Evaluate impact on *validation* set of pruning each possible node (plus those below it)
  - Greedily remove the one that most improve *validation* set accuracy
  - Continue until none help

Reduced Error Pruning



http://jmvidal.cse.sc.edu/talks/decisiontrees/allslides.html

Continuous Valued Inputs

- What if we have features that have continuous values?
  - One branch for each value?
    - Bad idea!!!! (impossible?)
- If we know the range of our values and we set how many branches the node should make then
  - Divide range evenly?
  - Somehow do it more intelligently?

Final Observations

- Let's think about this algorithm
  - Supervised or un-supervised?
  - Classification or regression?
  - Model-based or instance-based?
    - When it comes time to test/use, are we using the original data?
  - Linear vs Non-Linear?
  - Can this work on categorical data?
  - Can this work on continuous valued data?
  - Training Complexity?
  - Testing Complexity?
  - How to deal with overfitting?
  - Directly handles multi-class?