CS485/685
Lecture 2: January 7th, 2016

Decision Trees

Readings: [RN] Sec. 18.1-18.4,
[HTF] Sec. 9.2, [D] Chapt. 1,
[B] Sec. 14.4, [M] Sec. 16.2

Inductive Learning (recap)

- Induction
  - Given a training set of examples of the form $(x, f(x))$
    - x is the input, $f(x)$ is the output
  - Return a function h that approximates f
    - h is called the hypothesis

Supervised Learning

- Two types of problems
  1. **Classification:**
  2. **Regression**

- NB: The nature (categorical or continuous) of the domain (input space) of f does not matter

Classification Example

- Problem: Will you enjoy an outdoor sport based on the weather?
- Training set:

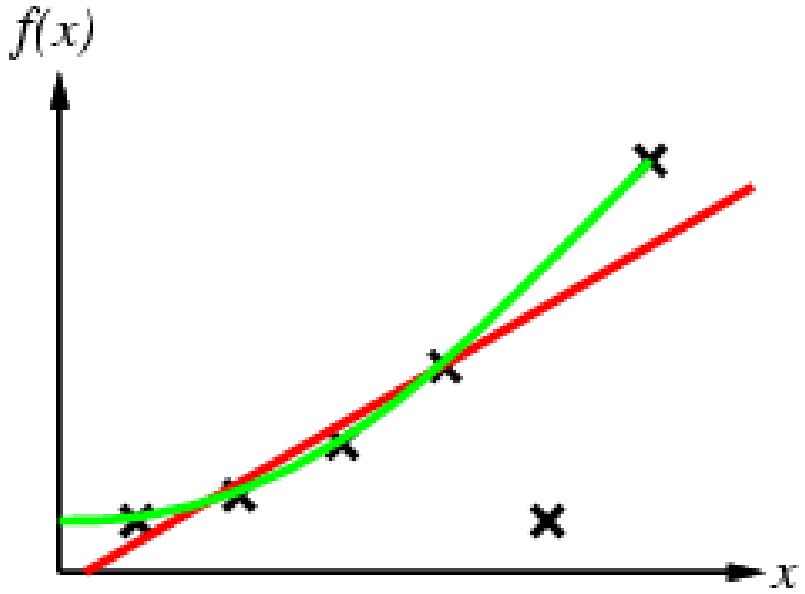
Sky	**Humidity**	**Wind**	**Water**	**Forecast**	**EnjoySport**
Sunny	Normal	Strong	Warm	Same	yes
Sunny	High	Strong	Warm	Same	yes
Sunny	High	Strong	Warm	Change	no
Sunny	High	Strong	Cool	Change	yes

x $f(x)$

- Possible Hypotheses:
  - $h_1: S = sunny \rightarrow ES = yes$
  - $h_2: Wa = cool$ or $F = same \rightarrow enjoySport$

Regression Example

- Find function h that fits f at instances x



More Examples

Problem	**Domain**	**Range**	**Classification / Regression**
Spam Detection			
Stock price prediction			
Speech recognition			
Digit recognition			
Housing valuation			
Weather prediction			

Hypothesis Space

- Hypothesis space H
  - Set of all hypotheses h that the learner may consider
  - Learning is a search through hypothesis space

- Objective: find h that minimizes
  - Misclassification
  - Or more generally some error function

  with respect to the training examples

- But what about unseen examples?

Generalization

- A good hypothesis will generalize well
  - i.e., predict unseen examples correctly

- Usually …
  - Any hypothesis h found to approximate the target function f well over a **sufficiently large set of training examples** will also approximate the target function well over any unobserved examples

Inductive learning

- Goal: find an h that agrees with f on training set
  - h is **consistent** if it agrees with f on all examples

- Finding a consistent hypothesis is not always possible
  - Insufficient hypothesis space:
    - E.g., it is not possible to learn exactly $f(x) = ax + b + xsin(x)$ when H = space of polynomials of finite degree
  - Noisy data
    - E.g., in weather prediction, identical conditions may lead to rainy and sunny days

Inductive Learning

- A learning problem is **realizable** if the hypothesis space contains the true function otherwise it is **unrealizable**.
  - Difficult to determine whether a learning problem is realizable since the true function is not known
- It is possible to use a very large hypothesis space
  - For example: H = class of all Turing machines
- But there is a **tradeoff** between **expressiveness** of a hypothesis class and the **complexity** of finding a good hypothesis

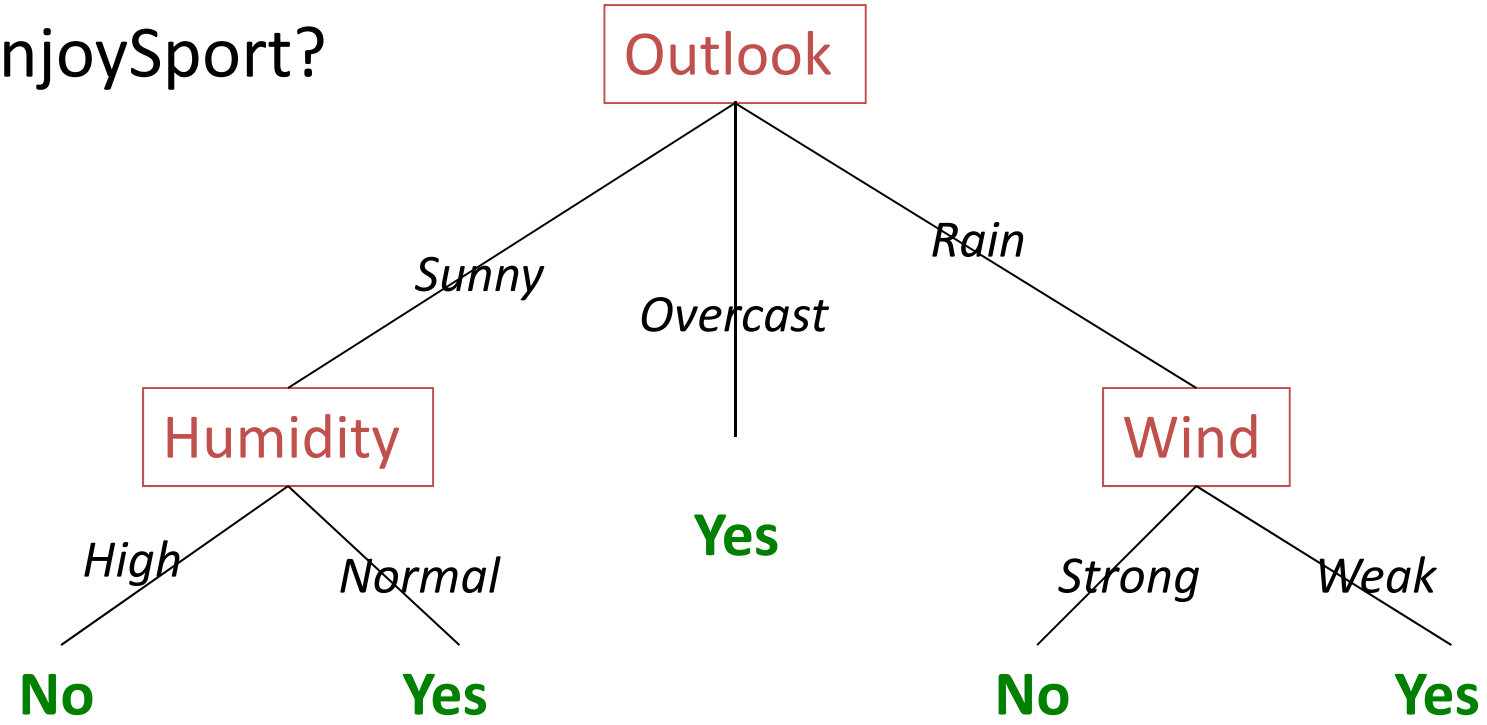
CART (Classification and Regression Trees)

- Tree
  - Nodes: labeled with attributes
  - Edges: labeled with attribute values
  - Leaves: labeled with
    - Classes (classification tree)
    - Values (Regression tree)

- Label an instance by following the branch consistent with the attribute values and returning the label stored in the resulting leaf.

Example: classification

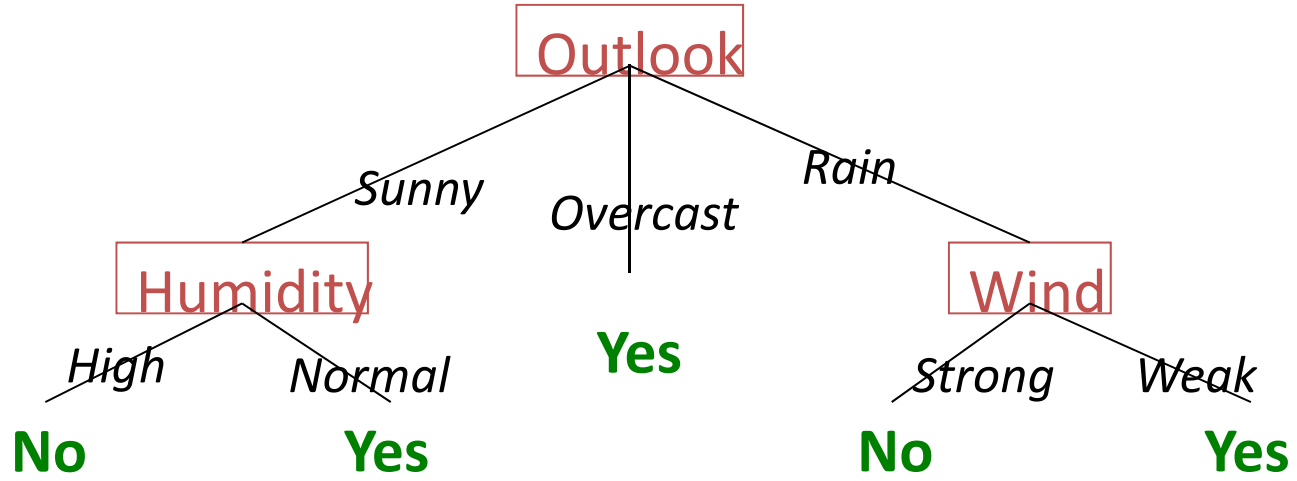
- EnjoySport?



- Instance:
- Classification:

Decision tree representation

- Decision trees can represent disjunctions of conjunctions of constraints on attribute values



Disjunction:

Decision tree representation

- Decision trees are fully expressive within the class of propositional languages
  - Any Boolean function can be written as a decision tree
    - Trivially by allowing each row in a truth table correspond to a path in the tree
    - Can often use small trees
    - Some functions require exponentially large trees (majority function, parity function)
  - However, there is no compact representation for all functions

Inducing a decision tree

- Aim: find a small tree consistent with the training examples

- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

Decision Tree Learning

```
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 attribute tests

- The central choice is deciding which attribute to test at each node

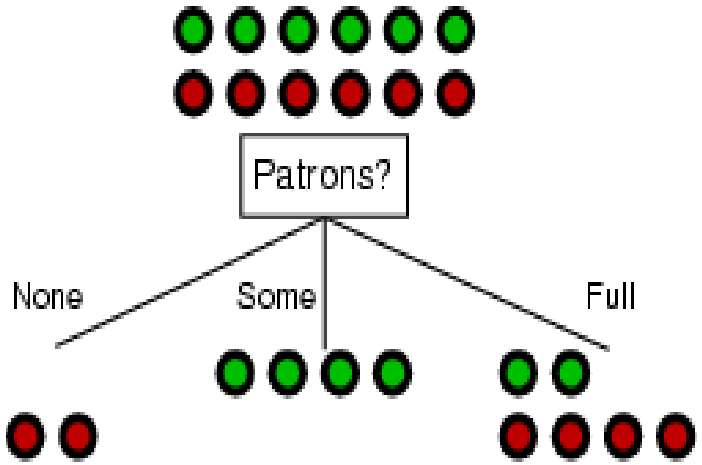
- We want to choose an attribute that is most useful for classifying examples

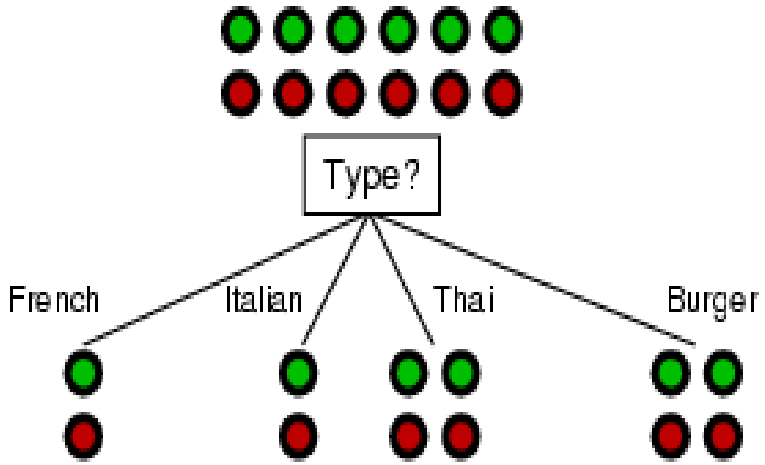
Example: Restaurant

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

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





- $Patrons?$ is better choice

Residual error for classification

- Let τ denote a leaf
- Let Q_τ denote the residual error at leaf τ
- Some residual error functions for classification
  - Error rate: $Q_\tau = (\#\tau - \max_k \#k)/\#\tau$
  - Gini Index: $Q_\tau = \sum_k p_\tau(k)(1 - p_\tau(k))$
  - Entropy: $Q_\tau = -\sum_k p_\tau(k) \log_2 p_\tau(k)$

Here k denotes the kth class

$\#\tau = amount\ of\ data\ in\ leaf\ \tau$

$\#k = amount\ of\ data\ in\ leaf\ \tau\ that\ belongs\ to\ class\ k$

and $p_\tau(k) = \frac{\#k}{\#\tau}$

Residual Error for Classification

Gini Index:

$Q_\tau = \sum_k p_\tau(k)(1 - p_\tau(k))$

Entropy:

$Q_\tau = \sum_k p_\tau(k)\,[-\log_2 p_\tau(k)]$

Expected misclassification when choosing the class according to $p_\tau(k)$

Expected #of bits to encode the class of an instance chosen at random according to $p_\tau(k)$

Residual Error for Regression

- Let $t_n = f(x_n)$ be the target for the nth example
- Let y_τ be the value returned by leaf τ
- Common residual error function for regression
  - Euclidean error: $E_\tau = \sum_{n \in R_\tau} (t_n - y_\tau)^2$

Choosing attribute tests

- In leaf τ, choose attribute A that reduces residual error the most when expanded

$$A^* = argmax_A \; Q_\tau - \sum_a p_\tau(A = a)\, Q_{\tau a}$$

Where $p_\tau(A = a) = \frac{\#(A=a)}{\#\tau}$

Example

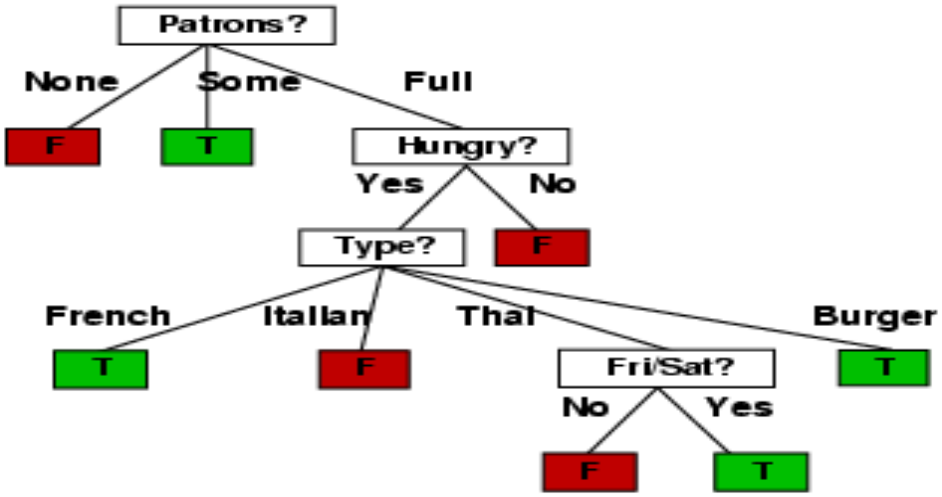
- Consider restaurant example (slide 19)
- Error rate reduction:

- Gini index reduction:

- Entropy reduction:

Example

- Decision tree learned from the 12 examples:

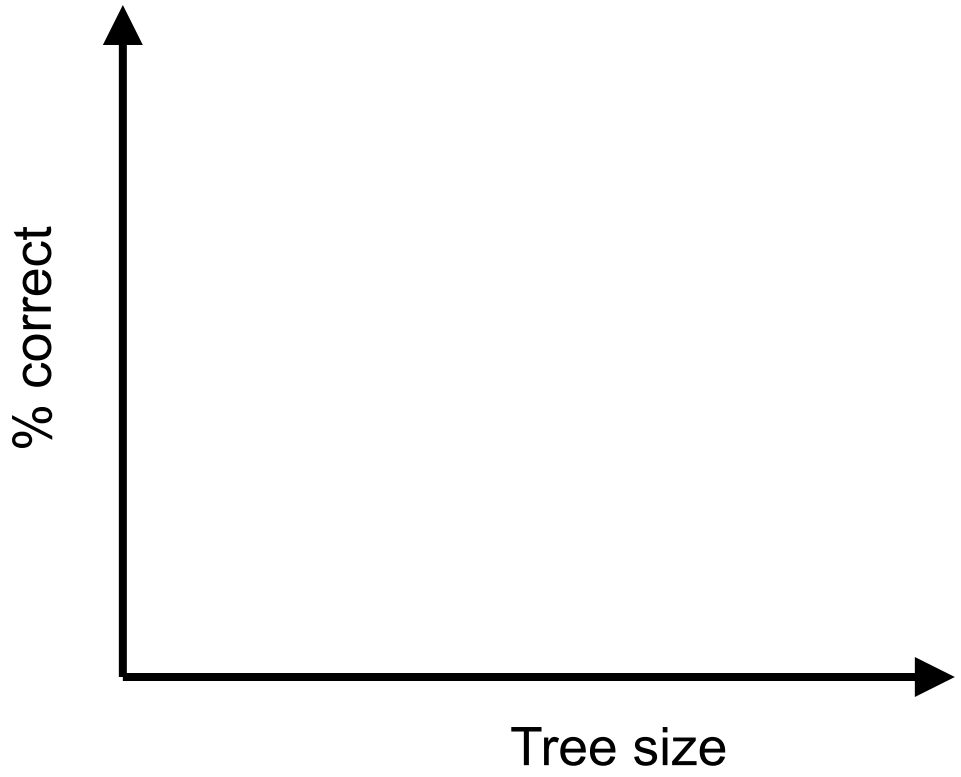


- Substantially simpler than “true” tree
  - a more complex hypothesis isn’t justified by small amount of data

Performance of a learning algorithm

- A learning algorithm is good if it produces a hypothesis that does a good job of predicting classifications of unseen examples
- Verify performance with a **test set**
  1. Collect a large set of examples
  2. Divide into 2 disjoint sets: training set and test set
  3. Learn hypothesis h with training set
  4. Measure percentage of correctly classified examples by h in the test set
  5. Repeat 2-4 for different randomly selected training sets of varying sizes

Learning curves



Overfitting

- Decision-tree grows until all training examples are perfectly classified

- But what if…
  - Data is noisy
  - Training set is too small to give a representative sample of the target function

- May lead to **Overfitting!**
  - Common problem with most learning algorithms

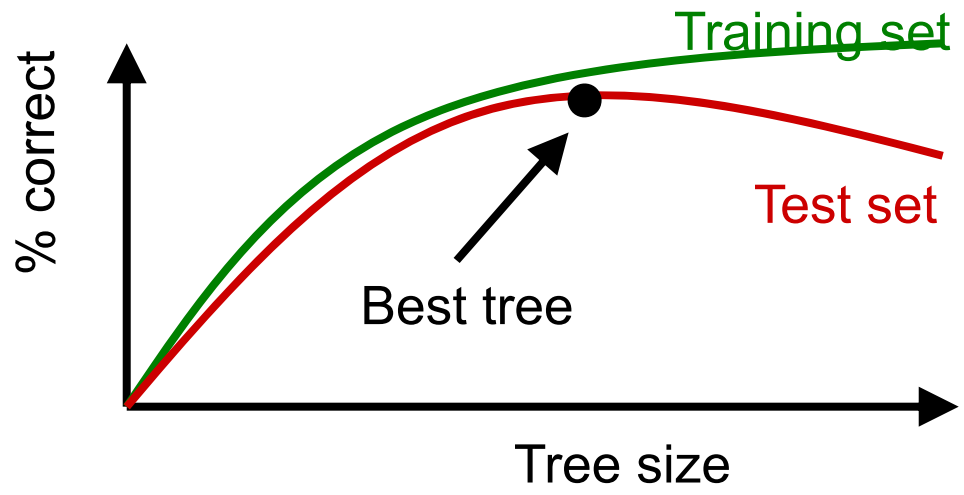
Overfitting

- **Definition**: Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$ such that h has smaller error than h' over the training examples but h' has smaller error than h over the entire distribution of instances

- Overfitting has been found to decrease accuracy of decision trees by 10-25%

Avoid overfitting

- Two popular techniques
  - Stop growing tree when test set performance starts decreasing
    - Use cross-validation
  - Prune statistically irrelevant nodes



Cross-validation

- Split data in two parts, one for training, one for testing the accuracy of a hypothesis

- K-fold cross validation means you run k experiments, each time putting aside $1/k$ of the data to test on and compute average accuracy of the k experiments.

Early stopping is difficult

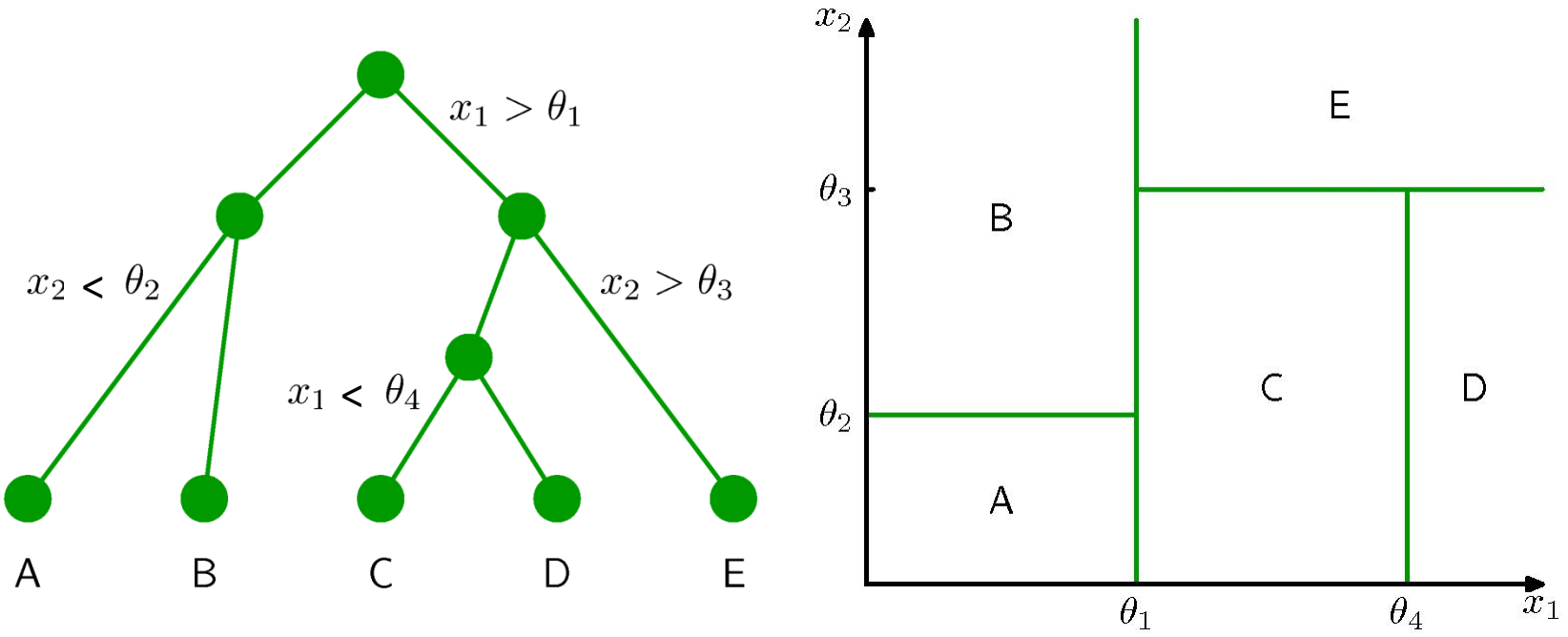
- Performance curves:
  - Train accuracy: monotonically increasing curve, but not smooth
  - Test accuracy: curve is not concave typically

Pruning

- Pruning is more common in practice
- Prune nodes in a bottom up fashion
- Two approaches:
  - Remove nodes that improve test accuracy by less than some threshold based on cross-validation
  - Regularization: add penalty term that reflects tree complexity (e.g., $|T| = $ #leaves in the tree)
    - $Q_\tau - \sum_a p_\tau(A = a) Q_{\tau a} - \lambda |T|$ is a weight that adjusts the importance of the penalty
    - Remove leaves with negative "regularized" error reduction

Decision tree with continuous attributes

- Tree partitions the input space



Decision tree with continuous attributes

- How do we come up with good partitions?
- Common approach: thresholding
  - Single attribute: $x_j > \theta_j$
  - Multi-attribute: $f(x_1, \ldots, x_M) > \theta$
    - Where f can be linear or non-linear
- Alternative: nearest neighbour (next class)