Lecture 10: Decision Tree Learning
CS486/686 Intro to Artificial Intelligence

2023-6-13

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

Outline

- Inductive learning
- Decision trees

What is Machine Learning?

- Definition:
  - A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. [Tom Mitchell, 1997]

Examples

- Computer Go (reinforcement learning):
  - T: playing the game of Go
  - P: percent of games won against an opponent
  - E: playing practice games against itself
- Handwriting recognition (supervised learning):
  - T: recognize handwritten words within images
  - P: percent of words correctly recognized
  - E: database of handwritten words with given classifications
- Customer profiling (unsupervised learning):
  - T: cluster customers based on transaction patterns
  - P: homogeneity of clusters
  - E: database of customer transactions

Representation

- Representation of the learned information is important
  - Determines how the learning algorithm will work

- Common representations:
  - Neural networks
  - Weighted combination of basis functions
  - Graphical models (e.g., Bayesian networks)
  - Propositional logic (e.g., decision trees)
  - …

Inductive learning (also known as concept learning)

- 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

Classification

- 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

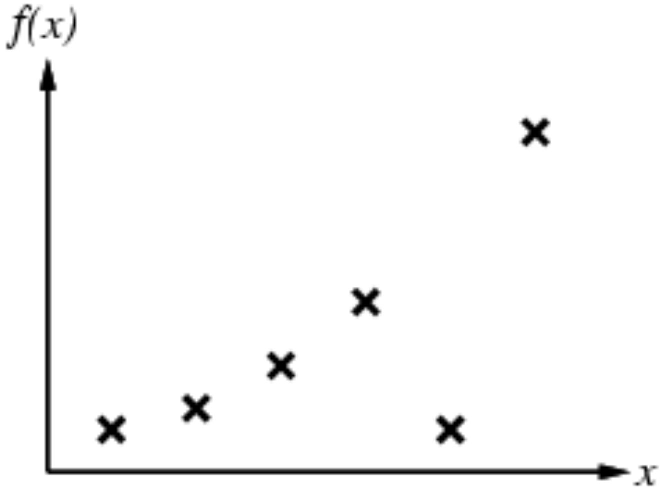
x

f(x)

- Possible hypotheses:
  - h_1: S=sunny → ES=yes
  - h_2: Wa=cool or F=same → enjoySport

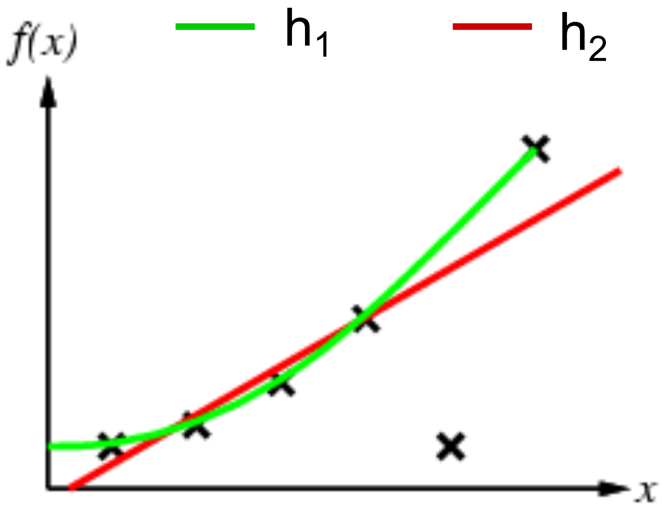
Regression

- Find function **h** that fits f at instances x



Regression

- Find function **h** that fits f at instances x



Hypothesis Space

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

- Objective:
  - Find hypothesis that agrees with training examples
  - But what about unseen examples?

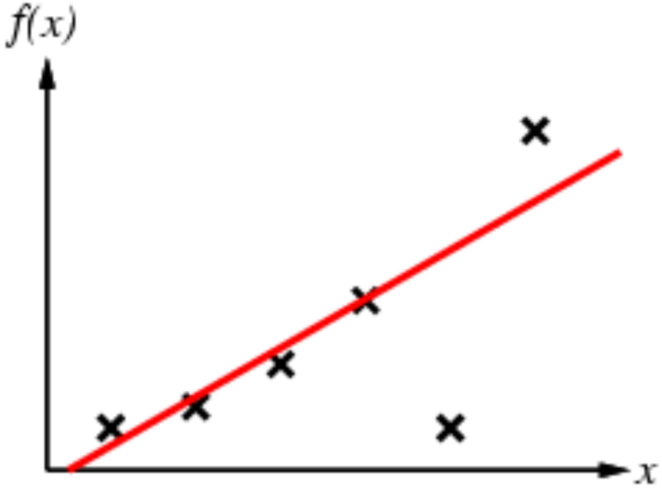
Generalization

- 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

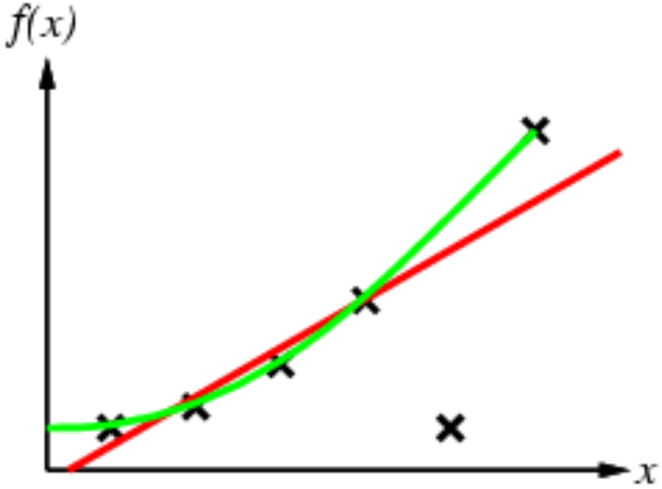
Inductive learning

- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



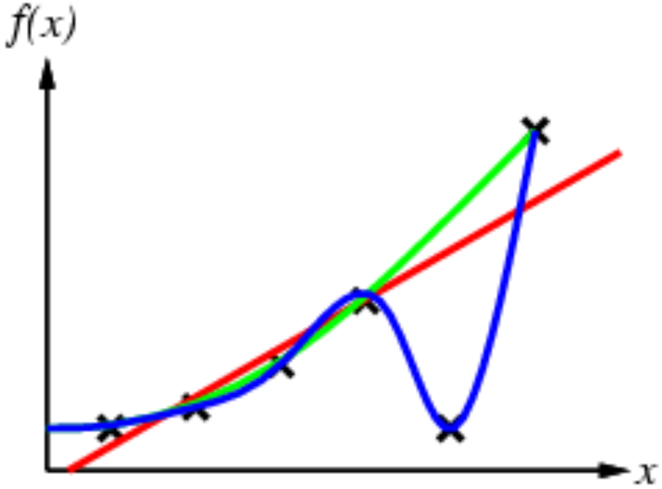
Inductive learning

- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



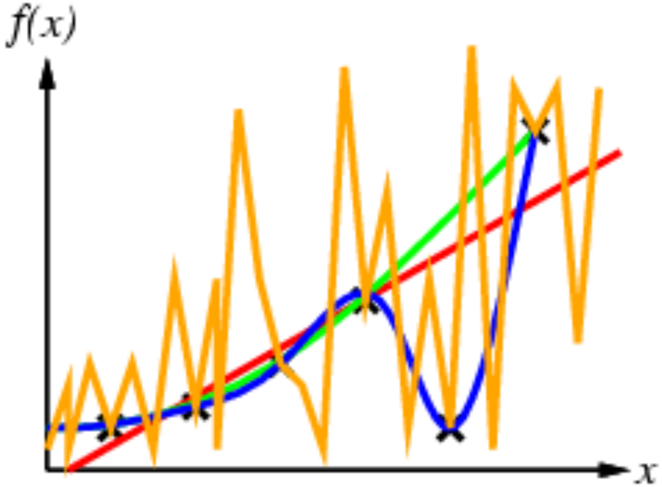
Inductive learning

- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



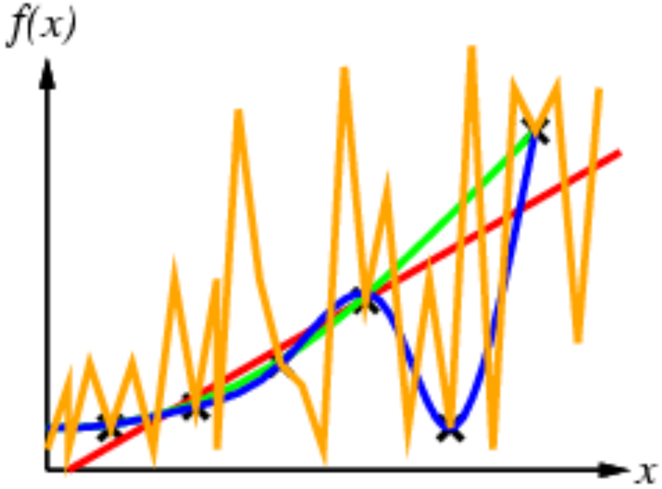
Inductive learning

- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



Inductive learning

- Construct/adjust h to agree with f on training set
- (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



- Ockham's razor: prefer the simplest hypothesis consistent with data

Inductive learning

- Finding a consistent hypothesis depends on the hypothesis space
  - For example, it is not possible to learn exactly f(x)=ax+b+xsin(x) when H=space of polynomials of finite degree
- 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

Inductive learning

- It is possible to use a very large hypothesis space
  - For example, H=class of all Turing machines

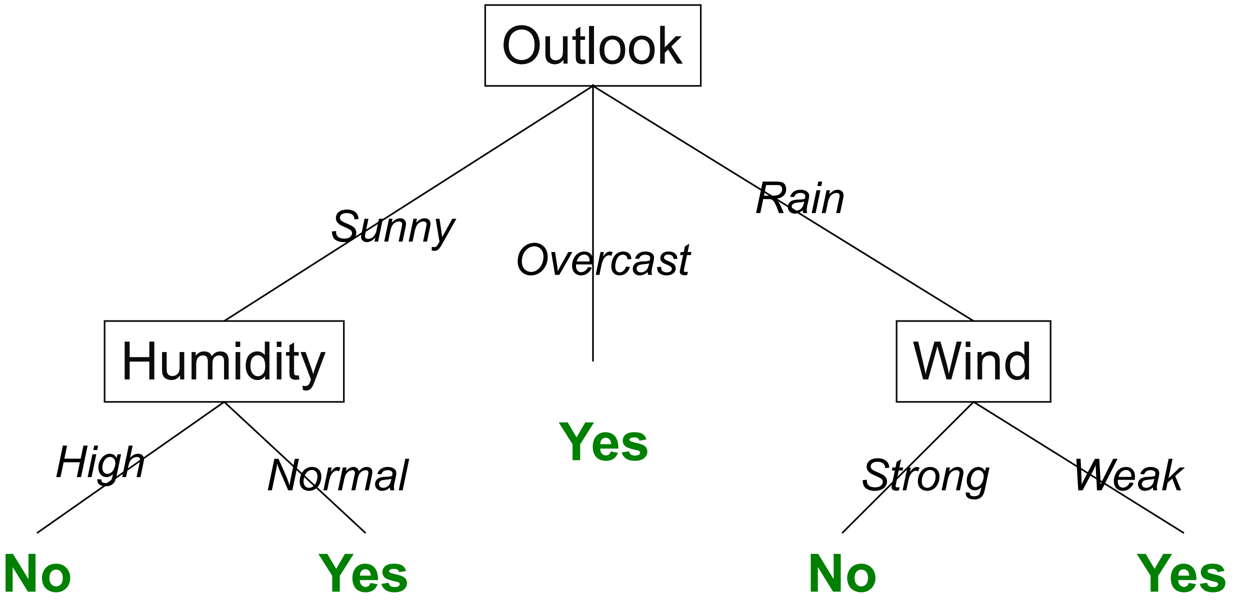
- But there is a tradeoff between expressiveness of a hypothesis class and complexity of finding simple, consistent hypothesis within the space
  - Fitting straight lines is easy, fitting high degree polynomials is hard, fitting Turing machines is very hard!

Decision trees

- Decision tree classification
  - Nodes: labeled with attributes
  - Edges: labeled with attribute values
  - Leaves: labeled with classes

- Classify an instance by starting at the root, testing the attribute specified by the root, then moving down the branch corresponding to the value of the attribute
  - Continue until you reach a leaf
  - Return the class

Decision tree (playing tennis)



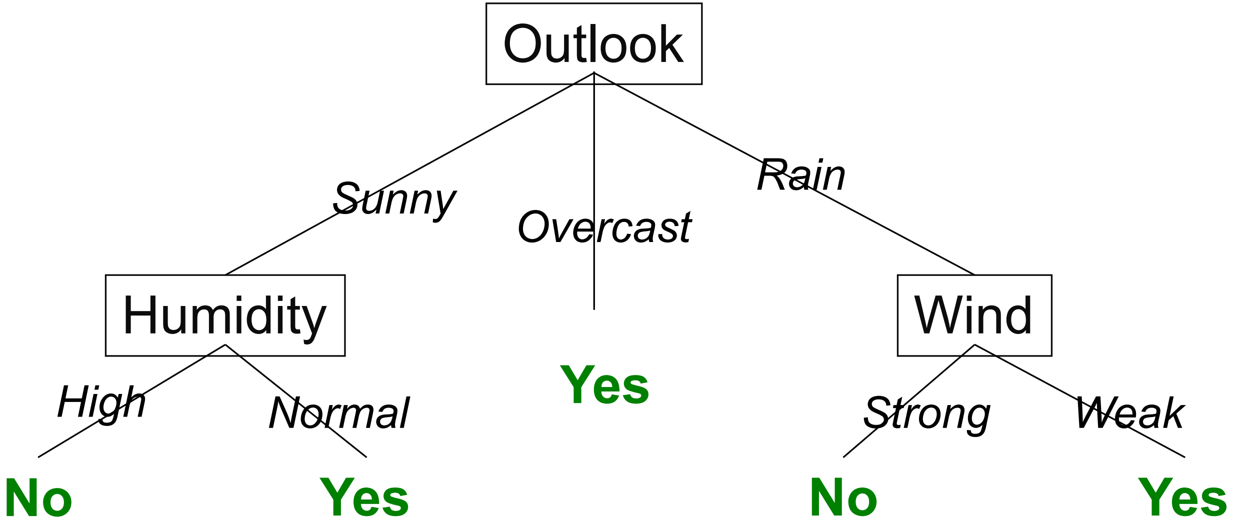
An instance

<Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong>

Classification: No

Decision tree representation

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



(Outlook=Sunny ∧ Humidity=Normal) ∨ (Outlook=Overcast) ∨ (Outlook=Rain ∧ Wind=Weak)

Decision tree representation

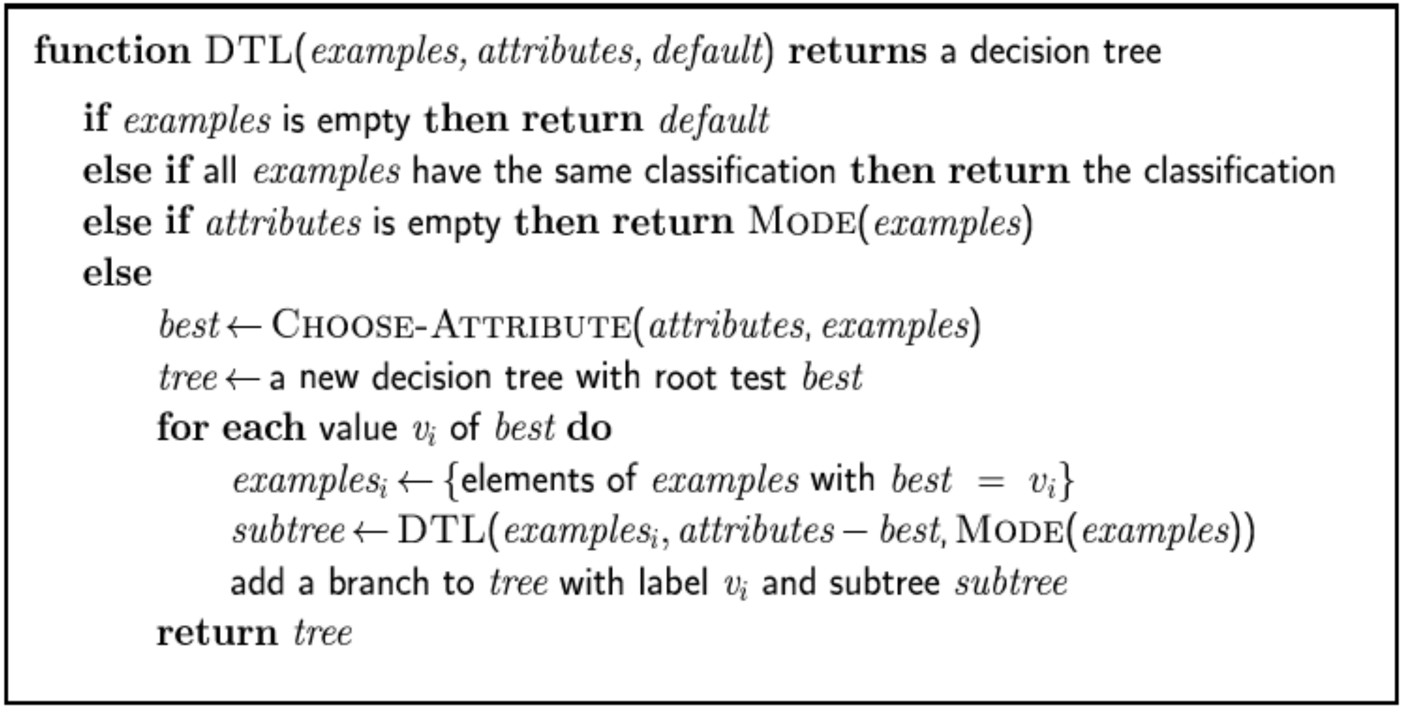
- Decision trees are fully expressive within the class of propositional languages
  - Any Boolean function can be written as a decision tree
    - Trivially by allowing each row in a truth table correspond to a path in the tree
    - Can often use small trees
    - Some functions require exponentially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



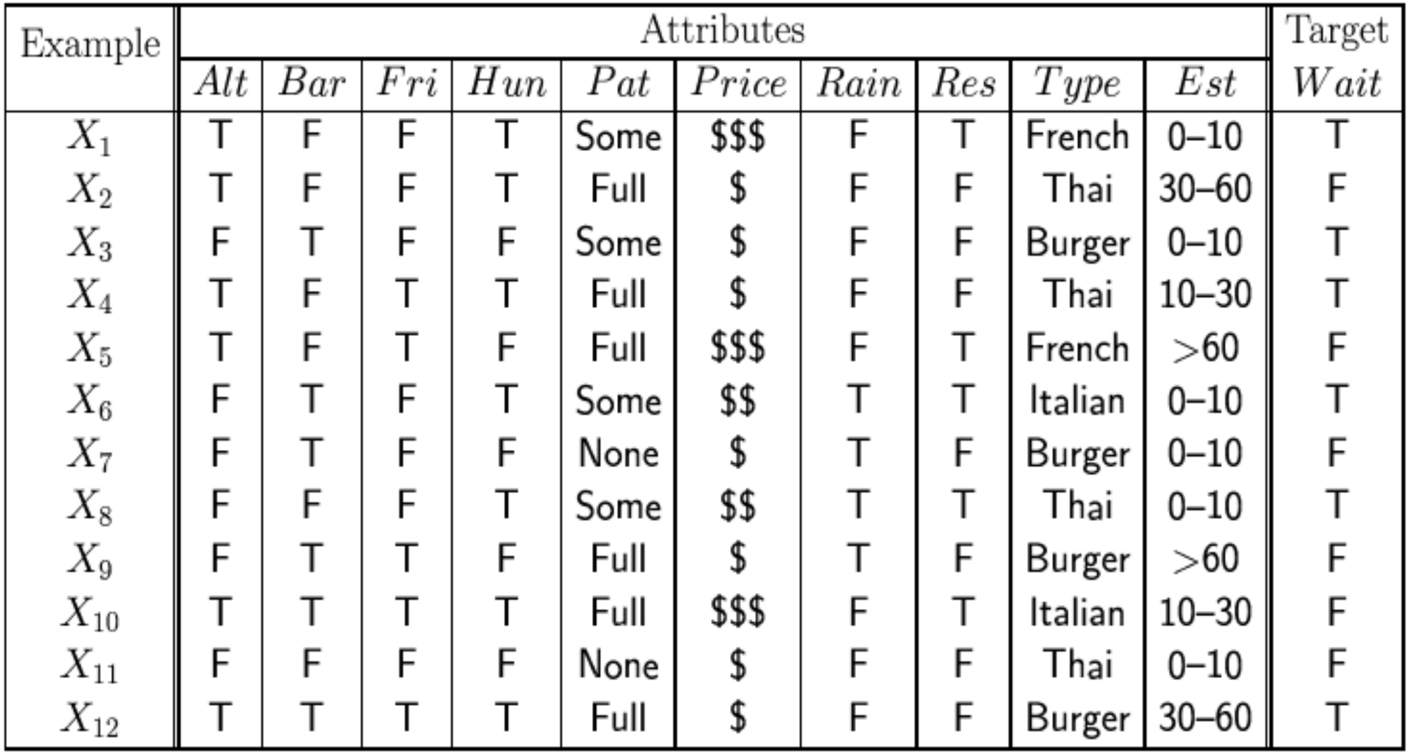
```
function DTL($examples, attributes, default$) returns a decision tree
    if $examples$ is empty then return $default$
    else if all $examples$ have the same classification then return the classification
    else if $attributes$ is empty then return MODE($examples$)
    else
        $best \leftarrow$ CHOOSE-ATTRIBUTE($attributes, examples$)
        $tree \leftarrow$ a new decision tree with root test $best$
        for each value $v_i$ of $best$ do
            $examples_i \leftarrow$ {elements of $examples$ with $best = v_i$}
            $subtree \leftarrow$ 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

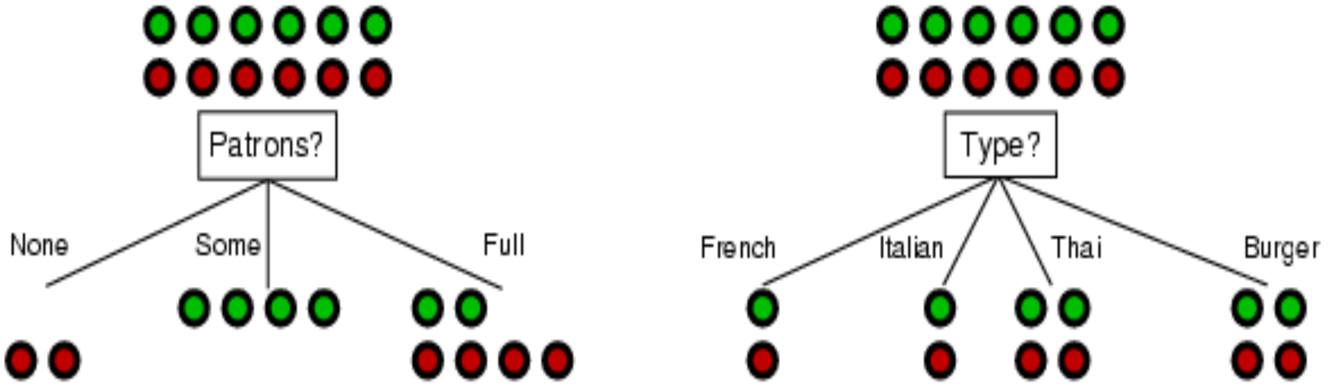
Example -- Restaurant



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"

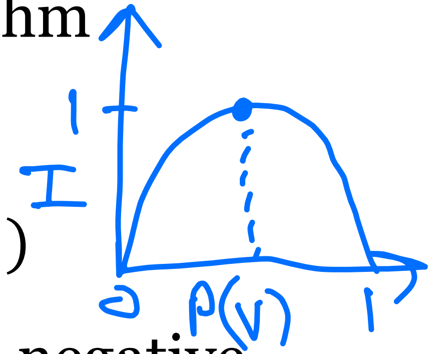


- *Patrons?* is a better choice

Using information theory

- To implement **Choose-Attribute** in the DTL algorithm
- Measure uncertainty (Entropy):

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



- For a training set containing p positive examples and n negative examples:

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information gain

- A chosen attribute *A* divides the training set *E* into subsets $E_1, \ldots, E_v$ according to their values for *A*, where *A* has *v* distinct values.

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

- Information Gain (IG) or reduction in uncertainty from the attribute test:

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

- Choose the attribute with the largest IG

Information gain

For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

Consider the attributes *Patrons* and *Type* (and others too):

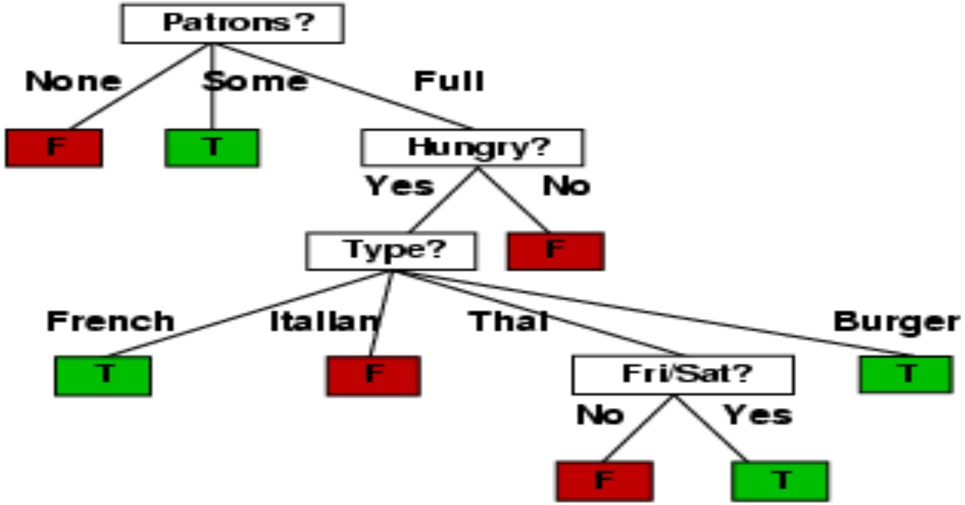
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Example

- Decision tree learned from the 12 examples:



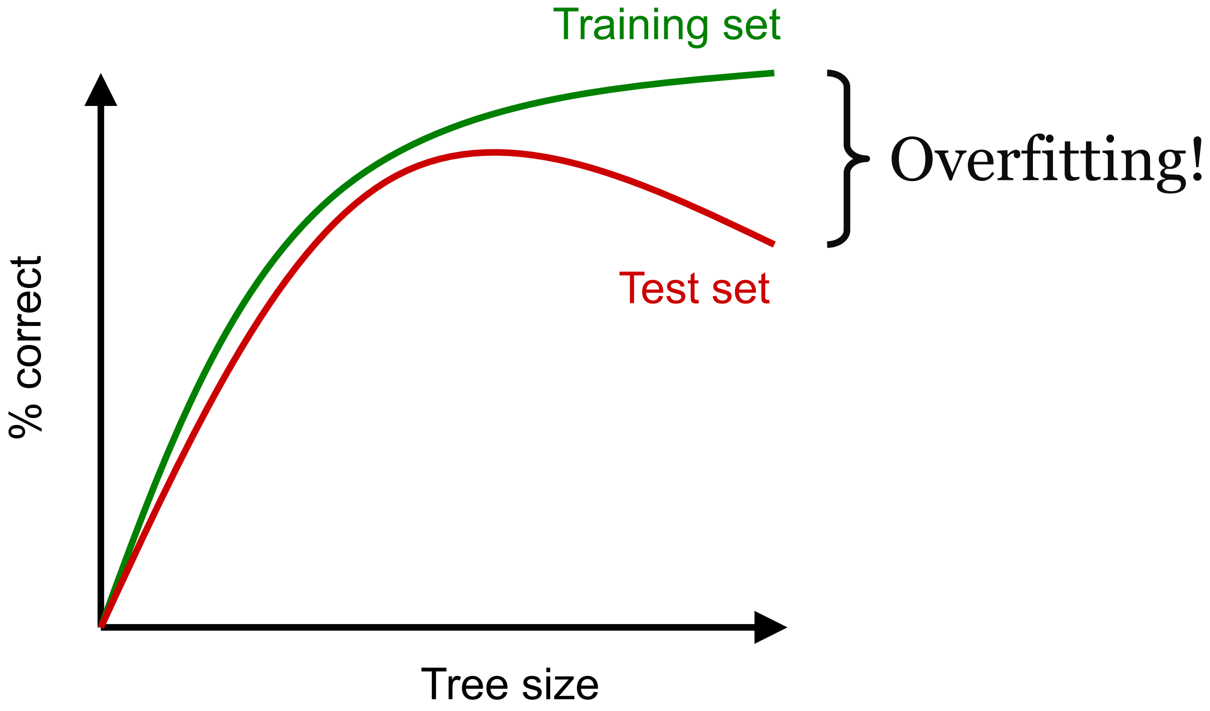
- Substantially simpler than "true" tree---a more complex hypothesis isn't justified by small amount of data

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

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

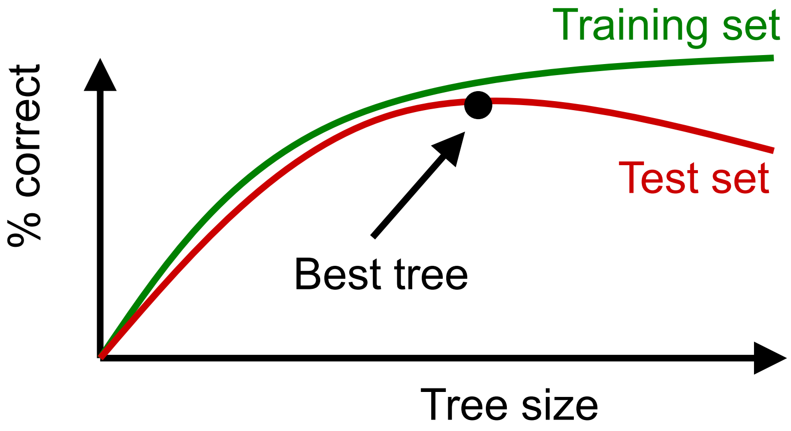
- May lead to **Overfitting!**
  - Definition: Given a hypothesis space H, a hypothesis h ∊ H is said to overfit the training data if there exists some alternative hypothesis h’ ∊ H such that h has smaller error than h’ over the training examples but h’ has smaller error than h over the entire distribution of instances

  - Overfitting can decrease accuracy of decision trees by 10-25%

Avoiding overfitting

Two popular techniques:

1. Prune statistically irrelevant nodes
   - Measure irrelevance with χ^2 test
2. Ideally: stop growing tree when test set performance decreases
   - Use cross-validation



Choosing Tree Size

- **Problem:** since we are choosing Tree Size based on the test set, the test set effectively becomes part of the training set when optimizing Tree Size. Hence, we cannot trust anymore the test set accuracy to be representative of future accuracy.

- Solution: split data into **training, validation and test sets**
  - **Training set:** compute decision tree
  - **Validation set:** optimize hyperparameters such as Tree Size
  - **Test set:** measure performance

Choosing Tree Size based on Validation Set

```
Let $TS$ be the Tree Size
For $TS = 1$ to max value
        $decisionTree_{TS} \leftarrow train(TS, trainingData)$
        $accuracy_{TS} \leftarrow eval(decisionTree_{TS}, validationData)$
$TS^* \leftarrow argmax_{TS}\ accuracy_{TS}$
$decisionTree_{TS^*} \leftarrow train(TS^*, trainingData \cup validationData)$
$accuracy \leftarrow eval(decisionTree_{TS^*}, testData)$
Return $k^*, accuracy$
```

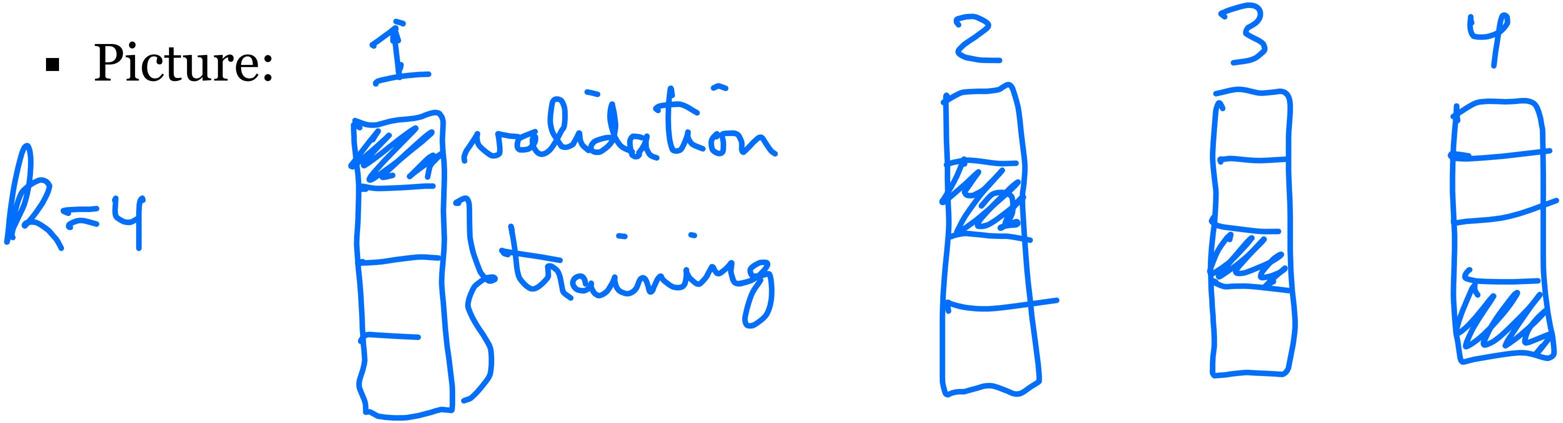
```
$eval(decisionTree, dataset)$
        $correct \leftarrow 0$
        For each $(x, y) \in dataset$
                if $y = decisionTree(x)$ then $correct \leftarrow correct + 1$
        $accuracy \leftarrow \frac{correct}{|dataset|}$
        return $accuracy$
```

Robust validation

- How can we ensure that validation accuracy is representative of future accuracy?
  - Validation accuracy becomes more reliable as we increase the size of the validation set
  - However, this reduces the amount of data left for training
- Popular solution: **cross-validation**

Cross-Validation

- Repeatedly split training data in two parts, one for training and one for validation. Report the average validation accuracy.
- ***k*-fold cross validation**: split training data in *k* equal size subsets. Run *k* experiments, each time validating on one subset and training on the remaining subsets. Compute the average validation accuracy of the *k* experiments.
- Picture:



Selecting Tree Size by Cross-Validation

```
Let $TS$ be the Tree Size
Let $k$ be the number of $trainData$ splits
For $TS = 1$ to max value
        For $i = 1$ to $k$ do   (where $i$ indexes $trainData$ splits)
                $decisionTree_{TS} \leftarrow train(TS, trainData_{1..i-1,i+1..k})$
                $accuracy_{TS,i} \leftarrow eval(decisionTree_{TS}, trainData_i)$
        $accuracy_{TS} \leftarrow average(\{accuracy_{TS,i}\}_{\forall i})$
$TS^* \leftarrow argmax_{TS}\, accuracy_{TS}$
$decisionTree_{TS^*} \leftarrow train(TS^*, trainData)$
$accuracy \leftarrow eval(decisionTree_{TS^*}, testData)$
Return $TS^*, accuracy$
```