COMP 562: Introduction to Machine Learning

Lecture 19 : Decision Trees, k-Nearest Neighbor

Mahmoud Mostapha [1]

Department of Computer Science

University of North Carolina at Chapel Hill

mahmoudm@cs.unc.edu

October 29, 2018

[1] Slides adapted from Eric Eaton

COMP562 - Lecture 19

Plan for today:

- Supervised Learning and Decision Trees
- Decision Trees and Overfitting,
- k-Nearest Neighbor and Instance-based Learning

Function Approximation

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$

Input: Training examples of unknown target function f

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

Output: Hypothesis $h \in H$ that best approximates f

Based on slide by Tom Mitchell

Sample Dataset

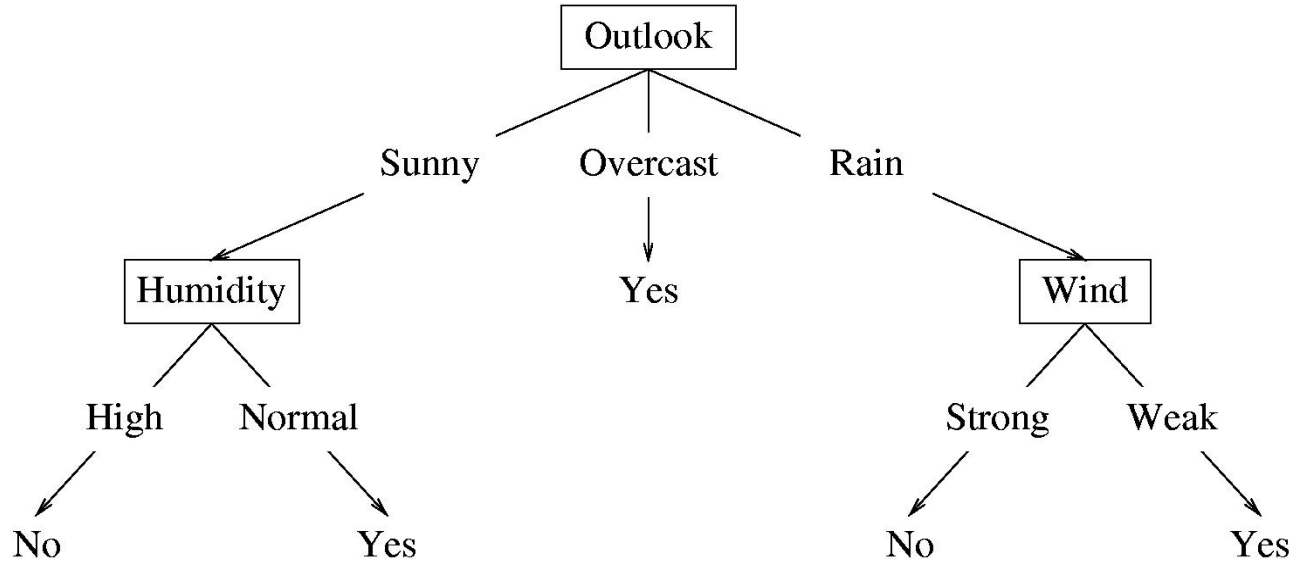
- Columns denote features X_i
- Rows denote labeled instances $\langle \boldsymbol{x}_i, y_i \rangle$
- Class label denotes whether a tennis game was played

$\langle \boldsymbol{x}_i, y_i \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision Tree

- A possible decision tree for the data:

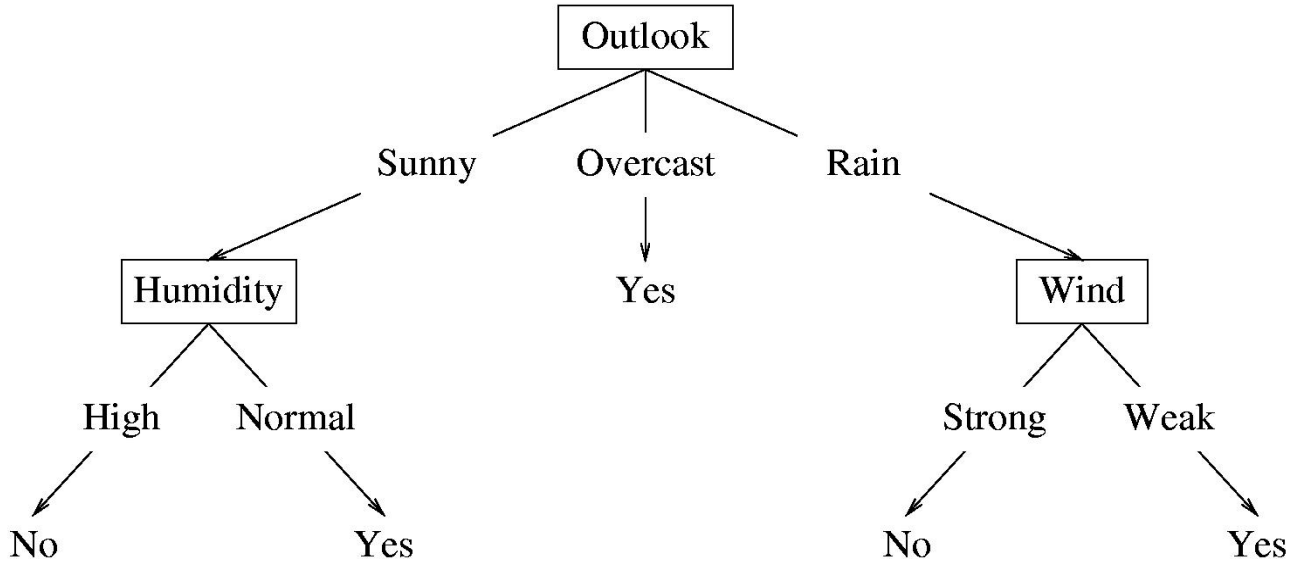


- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y \mid \boldsymbol{x} \in \text{leaf})$)

Based on slide by Tom Mitchell

Decision Tree

- A possible decision tree for the data:

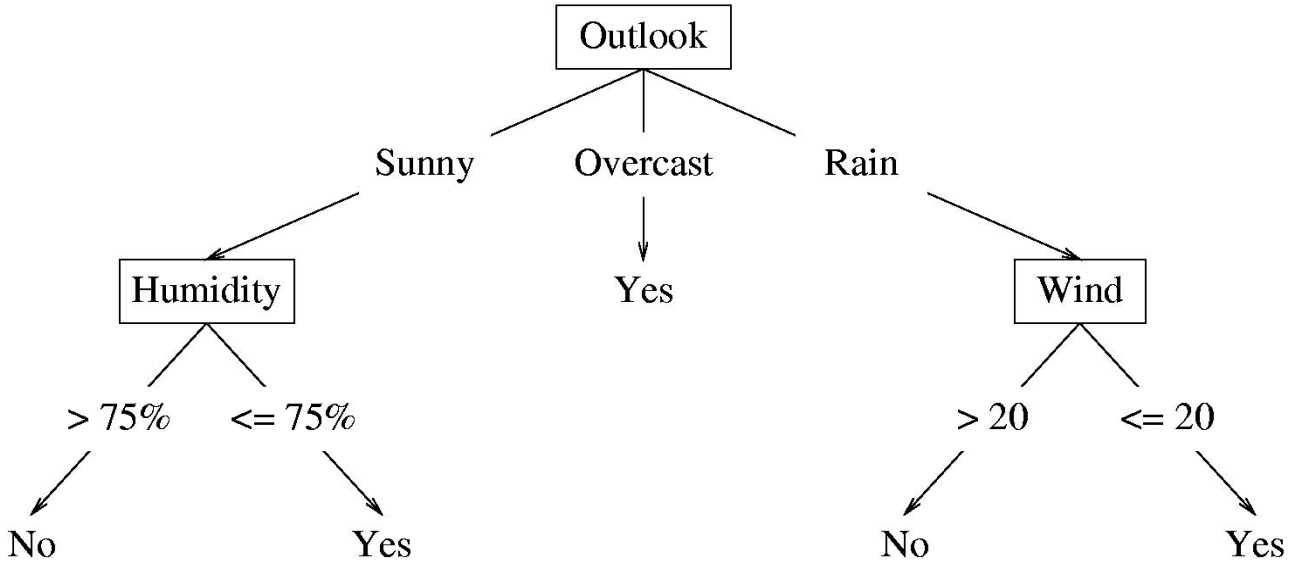


- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Based on slide by Tom Mitchell

Decision Tree

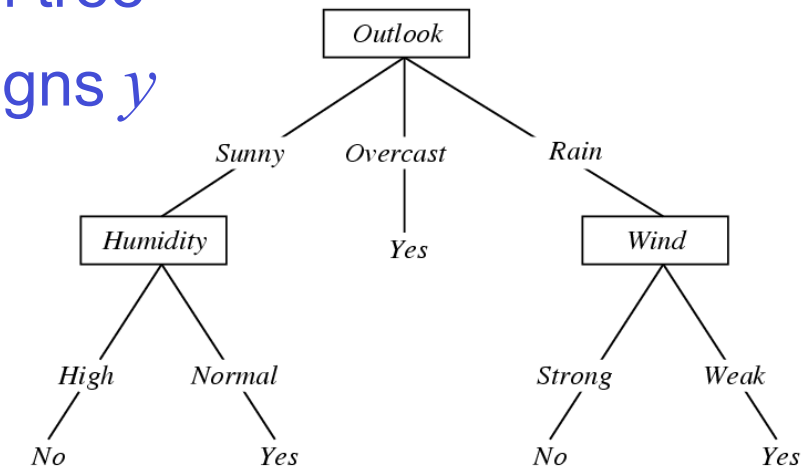
- If features are continuous, internal nodes can test the value of a feature against a threshold



Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y



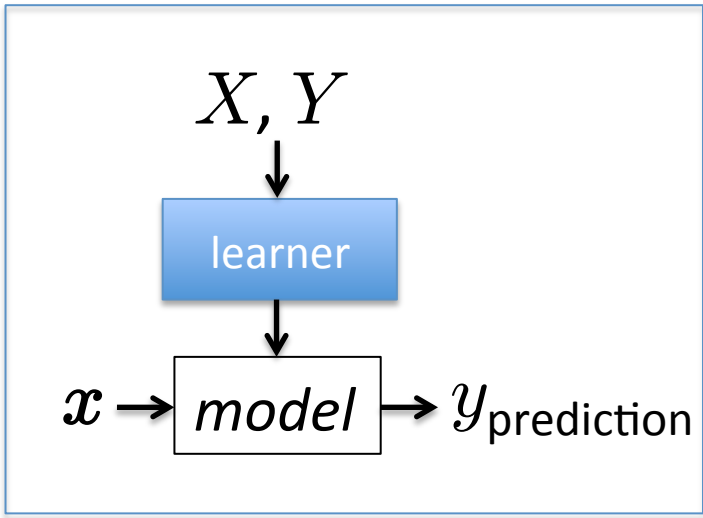
Stages of (Batch) Machine Learning

Given: labeled training data $X, Y = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n}$

- Assumes each $\boldsymbol{x}_i \sim \mathcal{D}(\mathcal{X})$ with $y_i = f_{target}(\boldsymbol{x}_i)$

Train the model:

model ← *classifier*.train(X, Y)



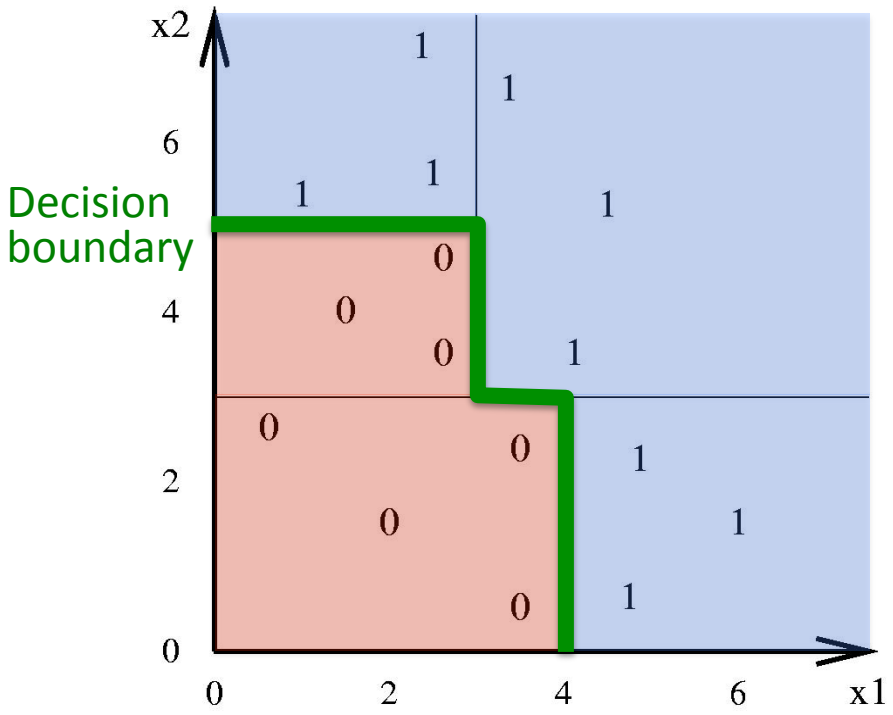
Apply the model to new data:

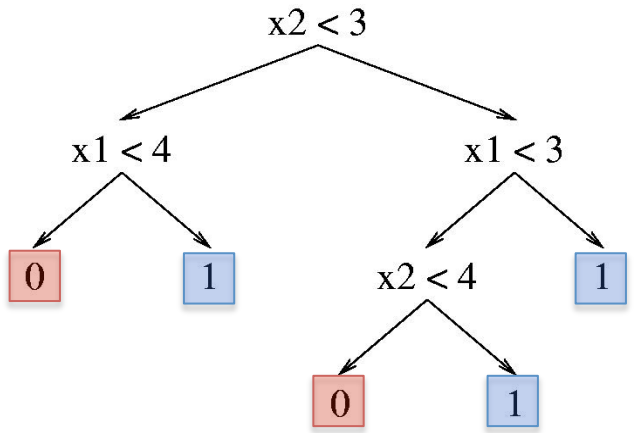
- Given: new unlabeled instance $\boldsymbol{x} \sim \mathcal{D}(\mathcal{X})$

$y_{\text{prediction}}$ ← *model*.predict($\boldsymbol{x}$)

Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels

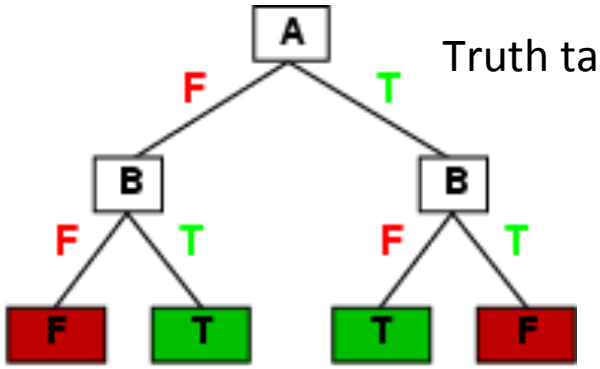




Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



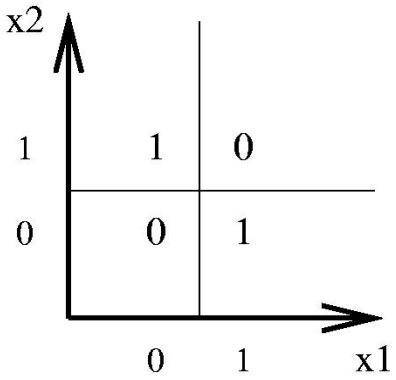
Truth table row → path to leaf

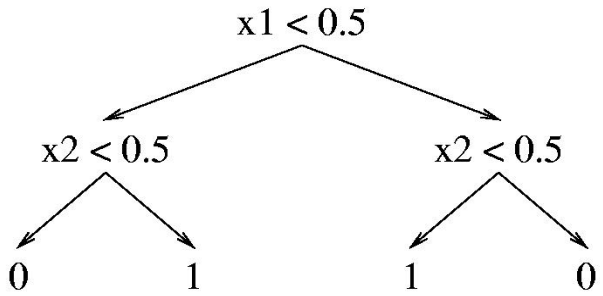
- In the worst case, the tree will require exponentially many nodes

Expressiveness

Decision trees have a variable-sized hypothesis space

- As the #nodes (or depth) increases, the hypothesis space grows
  - Depth 1 ("decision stump"): can represent any boolean function of one feature
  - Depth 2: any boolean fn of two features; some involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$)
  - etc.





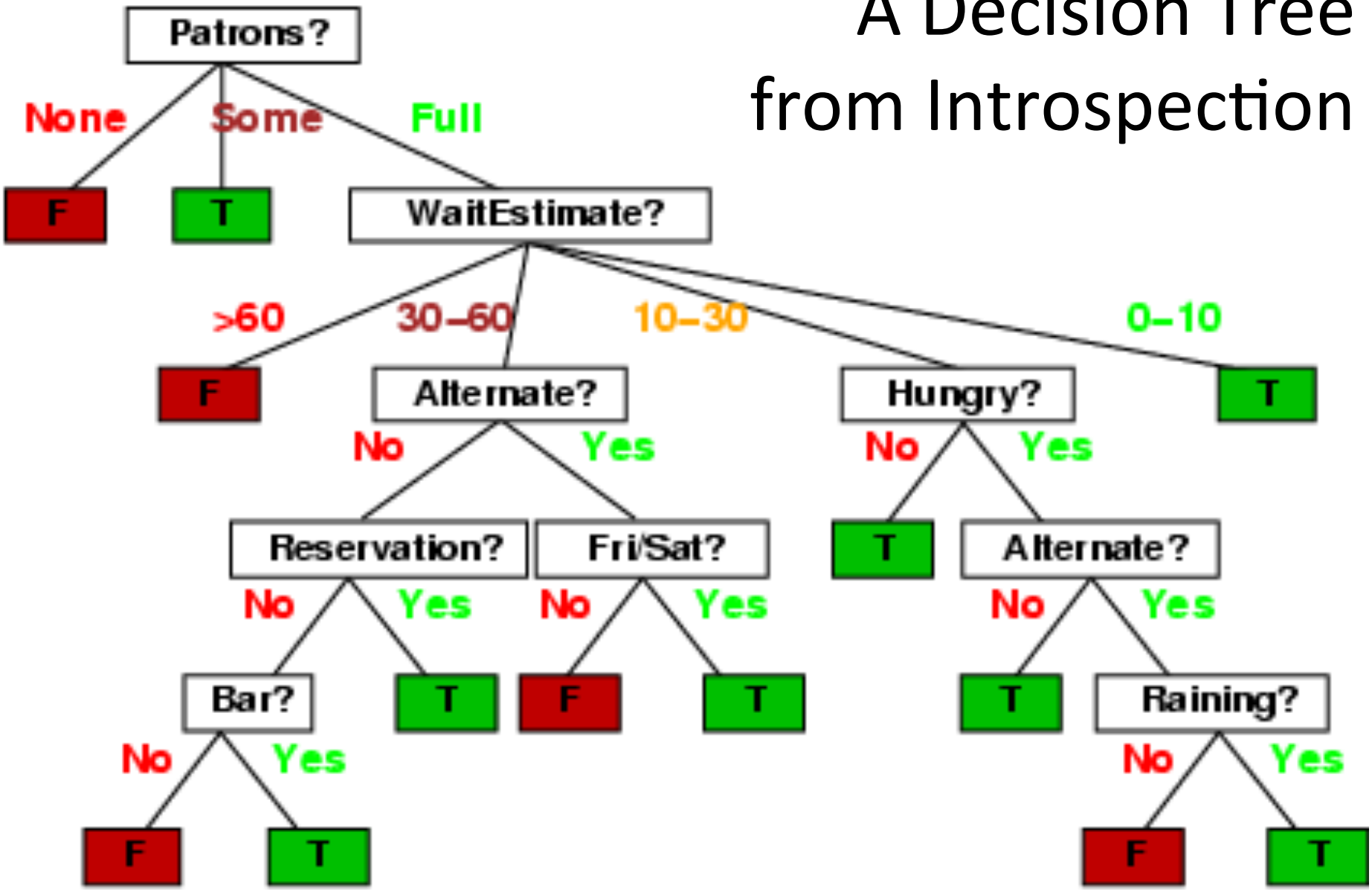
Another Example: Restaurant Domain (Russell & Norvig)

Model a patron's decision of whether to wait for a table at a 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

~7,000 possible cases

A Decision Tree from Introspection



Is this the best decision tree?

Preference bias: Ockham's Razor

- Principle stated by William of Ockham (1285-1347)
  - "*non sunt multiplicanda entia praeter necessitatem*"
  - entities are not to be multiplied beyond necessity
  - AKA Occam's Razor, Law of Economy, or Law of Parsimony

Idea: The simplest consistent explanation is the best

- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  - Finding the provably smallest decision tree is NP-hard
  - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

node = root of decision tree

Main loop:

1. $A \leftarrow$ the "best" decision attribute for the next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create a new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recurse over new leaf nodes.

How do we choose which attribute is best?

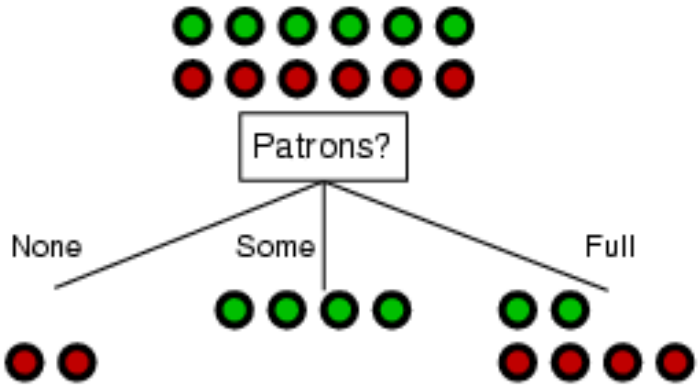
Choosing the Best Attribute

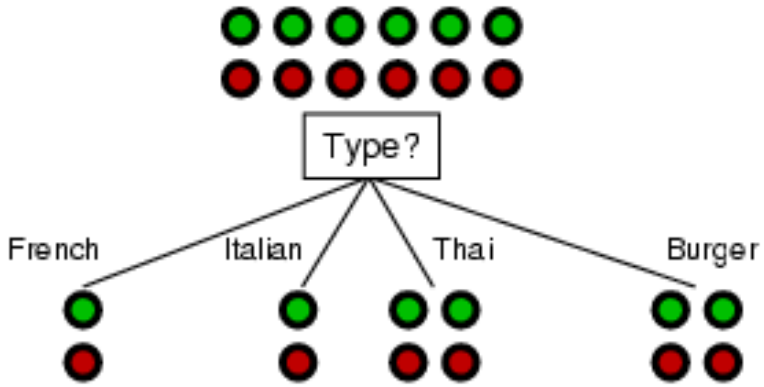
Key problem: choosing which attribute to split a given set of examples

- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

Choosing an Attribute

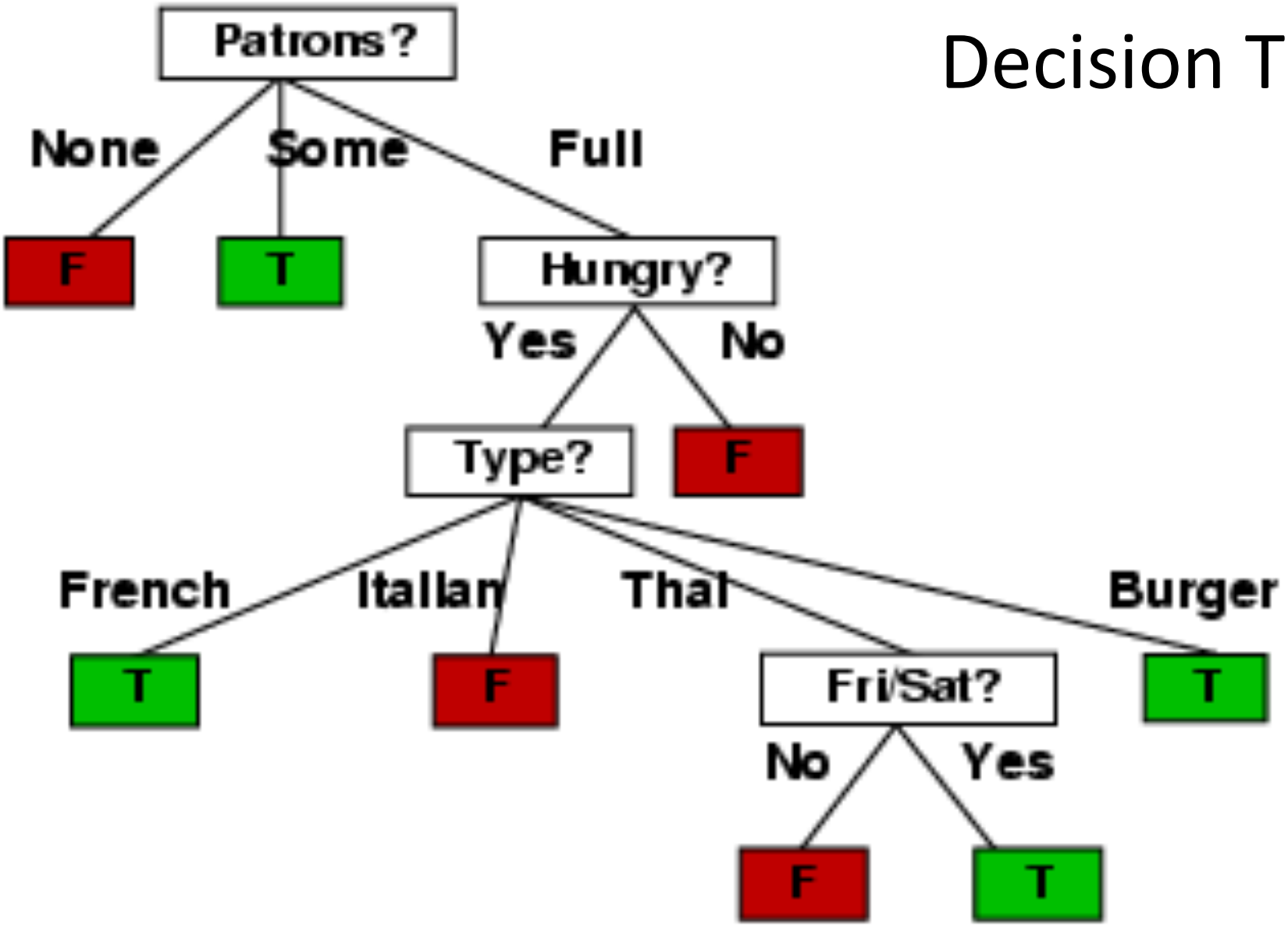
Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





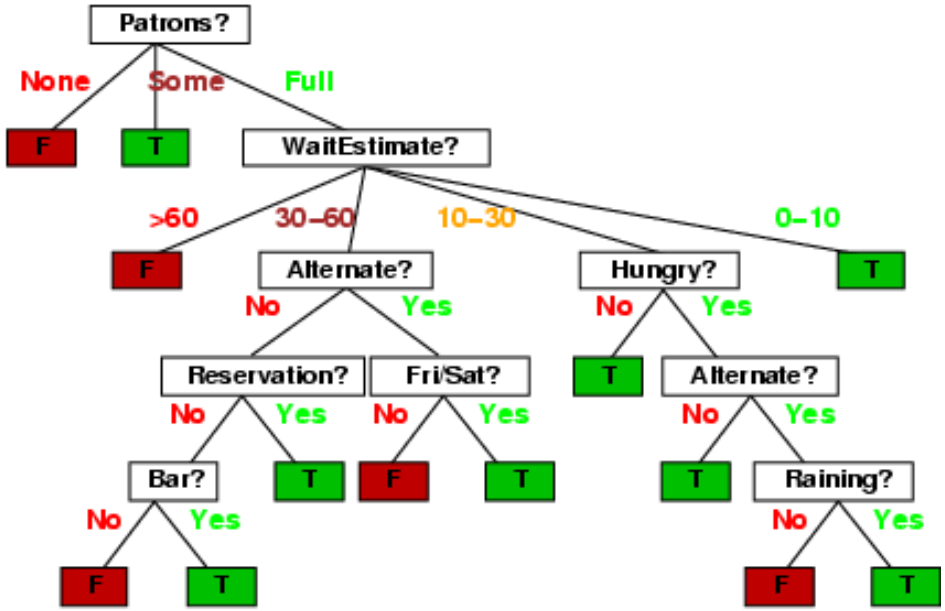
Which split is more informative: *Patrons?* or *Type?*

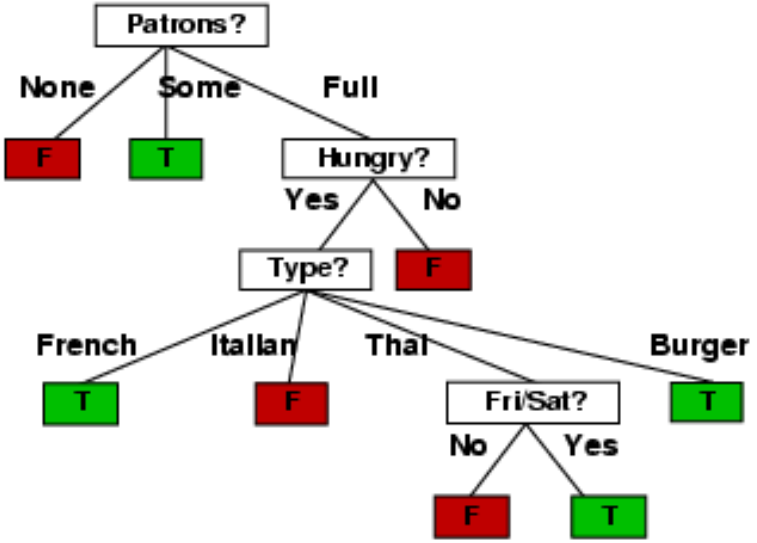
ID3-induced Decision Tree



Based on Slide from M. desJardins & T. Finin

Compare the Two Decision Trees



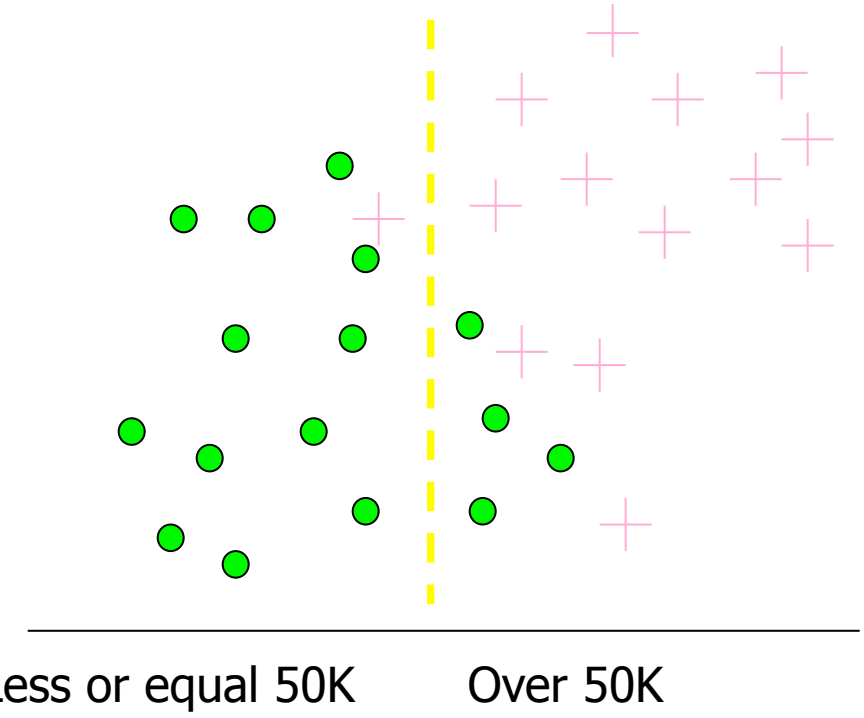


Based on Slide from M. desJardins & T. Finin

Information Gain

Which test is more informative?

Split over whether Balance exceeds 50K



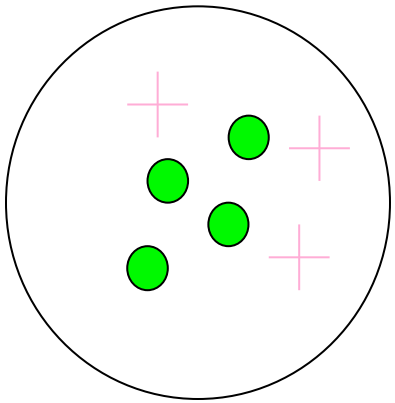
Split over whether applicant is employed

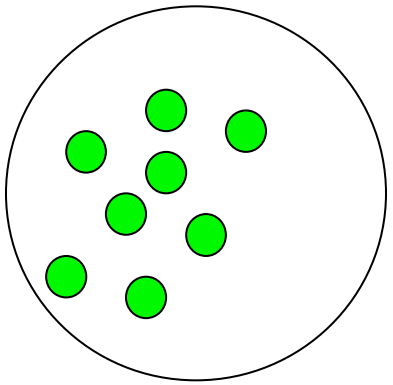
Based on slide by Pedro Domingos

Information Gain

Impurity/Entropy (informal)

- Measures the level of **impurity** in a group of examples

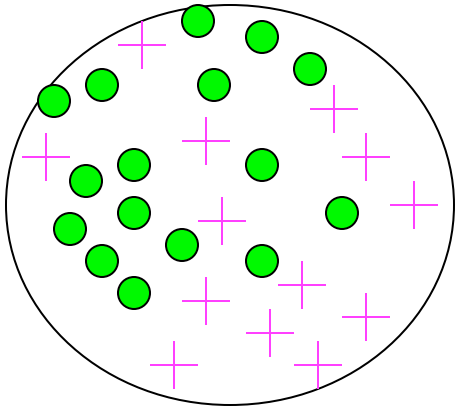




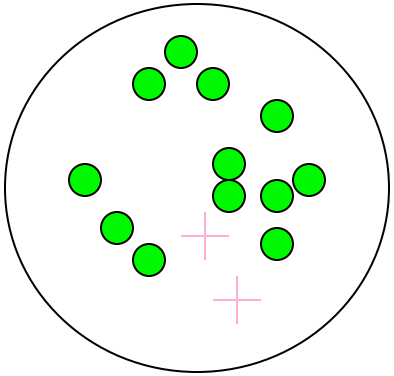
Based on slide by Pedro Domingos

Impurity

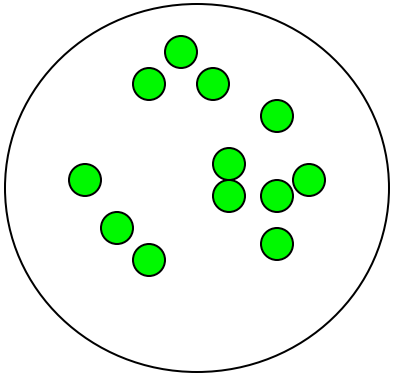
Very impure group



Less impure



Minimum impurity



Based on slide by Pedro Domingos

Entropy: a common way to measure impurity

Entropy $H(X)$ of a random variable X

# of possible values for X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Entropy: a common way to measure impurity

Entropy $H(X)$ of a random variable X

# of possible values for X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Why? Information theory:

- Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

Slide by Tom Mitchell

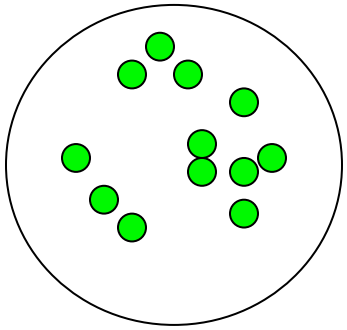
2-Class Cases:

Entropy $$H(x) = -\sum_{i=1}^{n} P(x=i)\log_2 P(x=i)$$

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1 = 0$

  not a good training set for learning

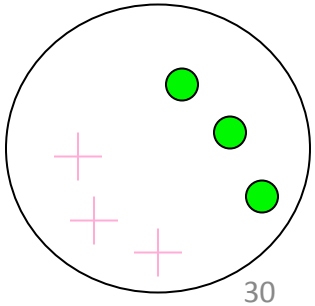
Minimum impurity



- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

  good training set for learning

Maximum impurity



Based on slide by Pedro Domingos

Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Based on slide by Pedro Domingos

From Entropy to Information Gain

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

From Entropy to Information Gain

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v) \log_2 P(X=i|Y=v)$$

From Entropy to Information Gain

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v) \log_2 P(X=i|Y=v)$$

Conditional entropy $H(X|Y)$ of X given Y :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v) H(X|Y=v)$$

Slide by Tom Mitchell

From Entropy to Information Gain

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v) \log_2 P(X=i|Y=v)$$

Conditional entropy $H(X|Y)$ of X given Y :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v) H(X|Y=v)$$

Mututal information (aka Information Gain) of X and Y :

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

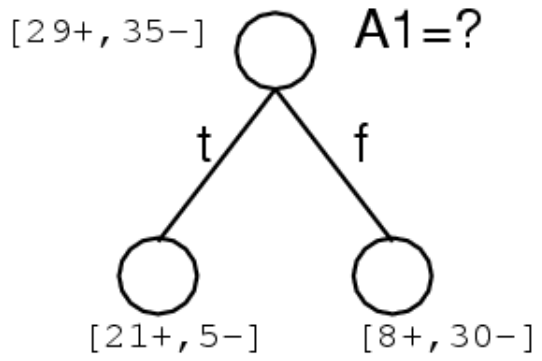
Slide by Tom Mitchell

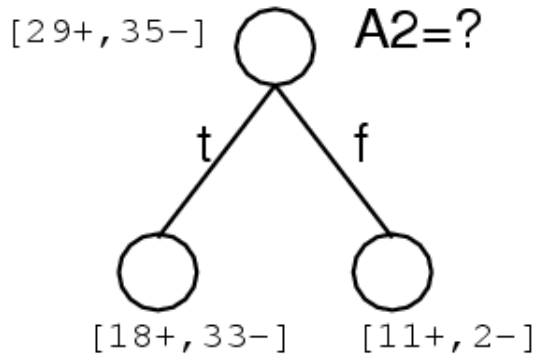
Information Gain

Information Gain is the mutual information between input attribute A and target variable Y

Information Gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A

$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$





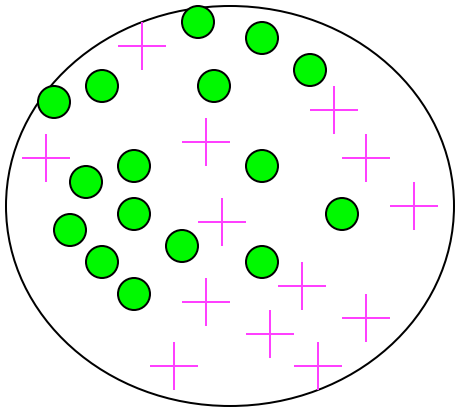
Slide by Tom Mitchell

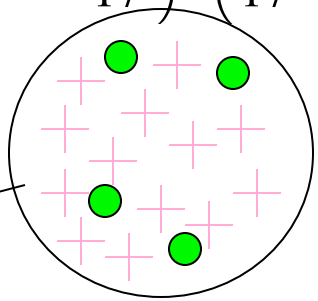
Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

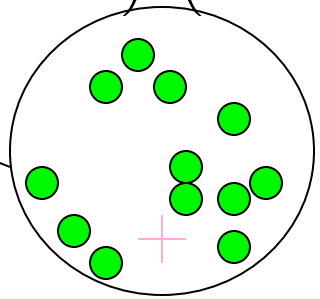
Entire population (30 instances)





17 instances

child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



13 instances

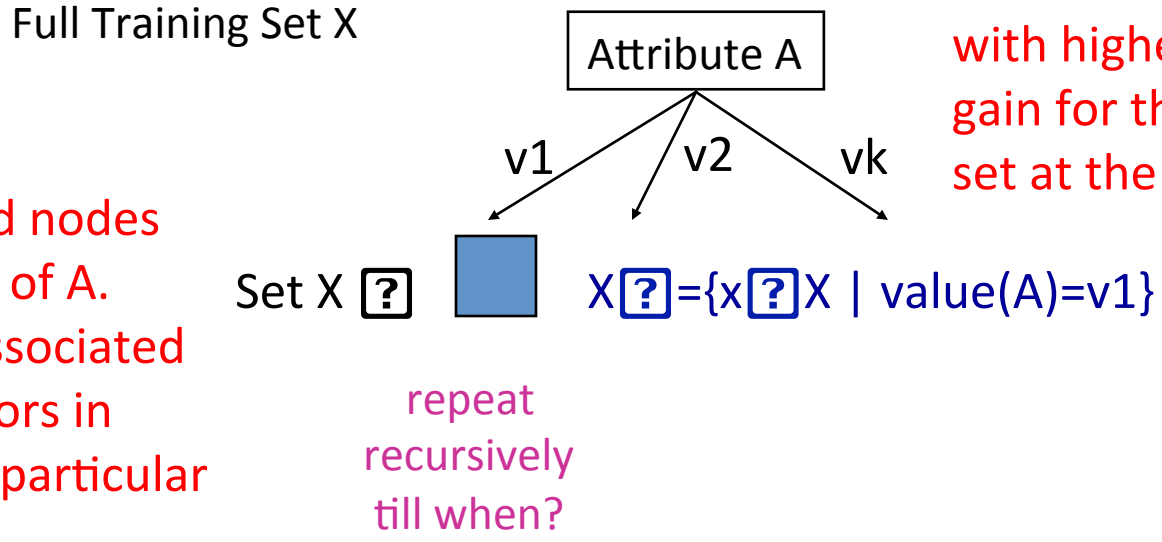
parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Based on slide by Pedro Domingos

Using Information Gain to Construct a Decision Tree



Disadvantage of information gain:

- It prefers attributes with large number of values that split the data into small, pure subsets
- Quinlan's gain ratio uses normalization to improve this

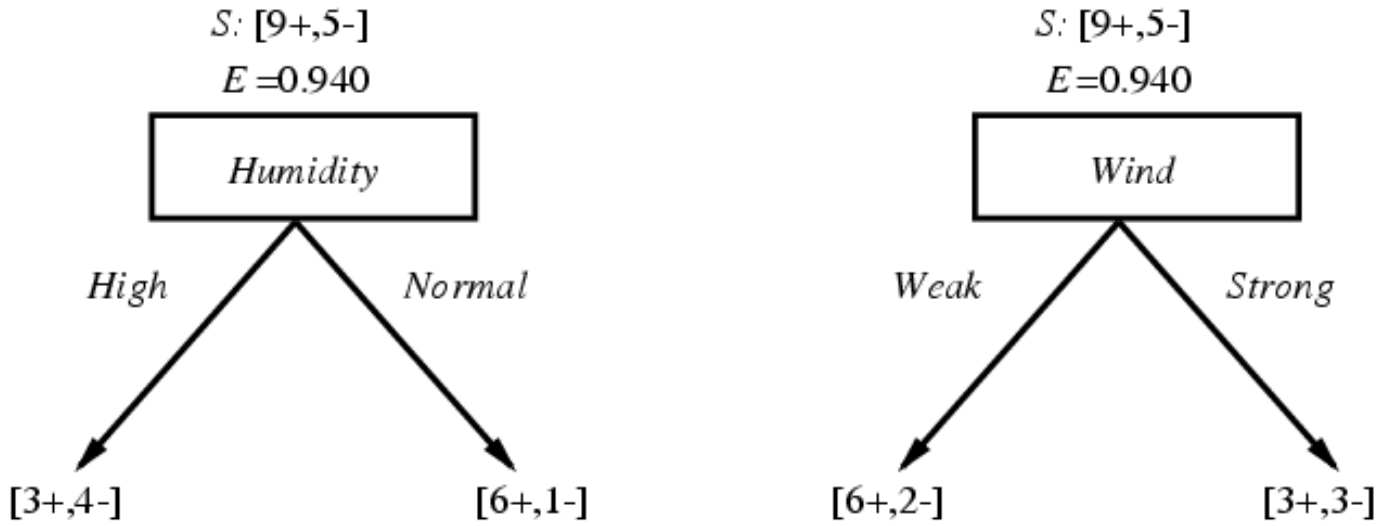
Based on slide by Pedro Domingos

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the Next Attribute

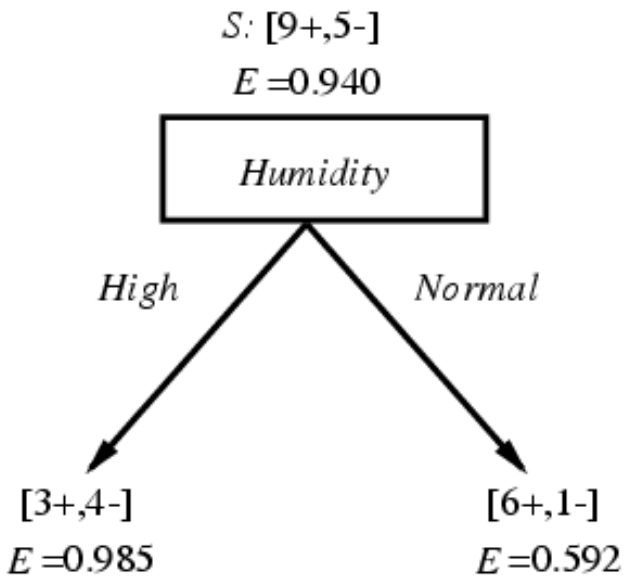
Which attribute is the best classifier?

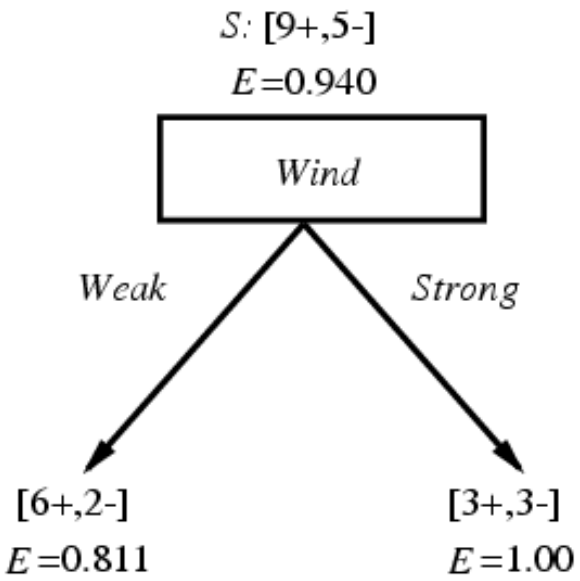


Slide by Tom Mitchell

Selecting the Next Attribute

Which attribute is the best classifier?





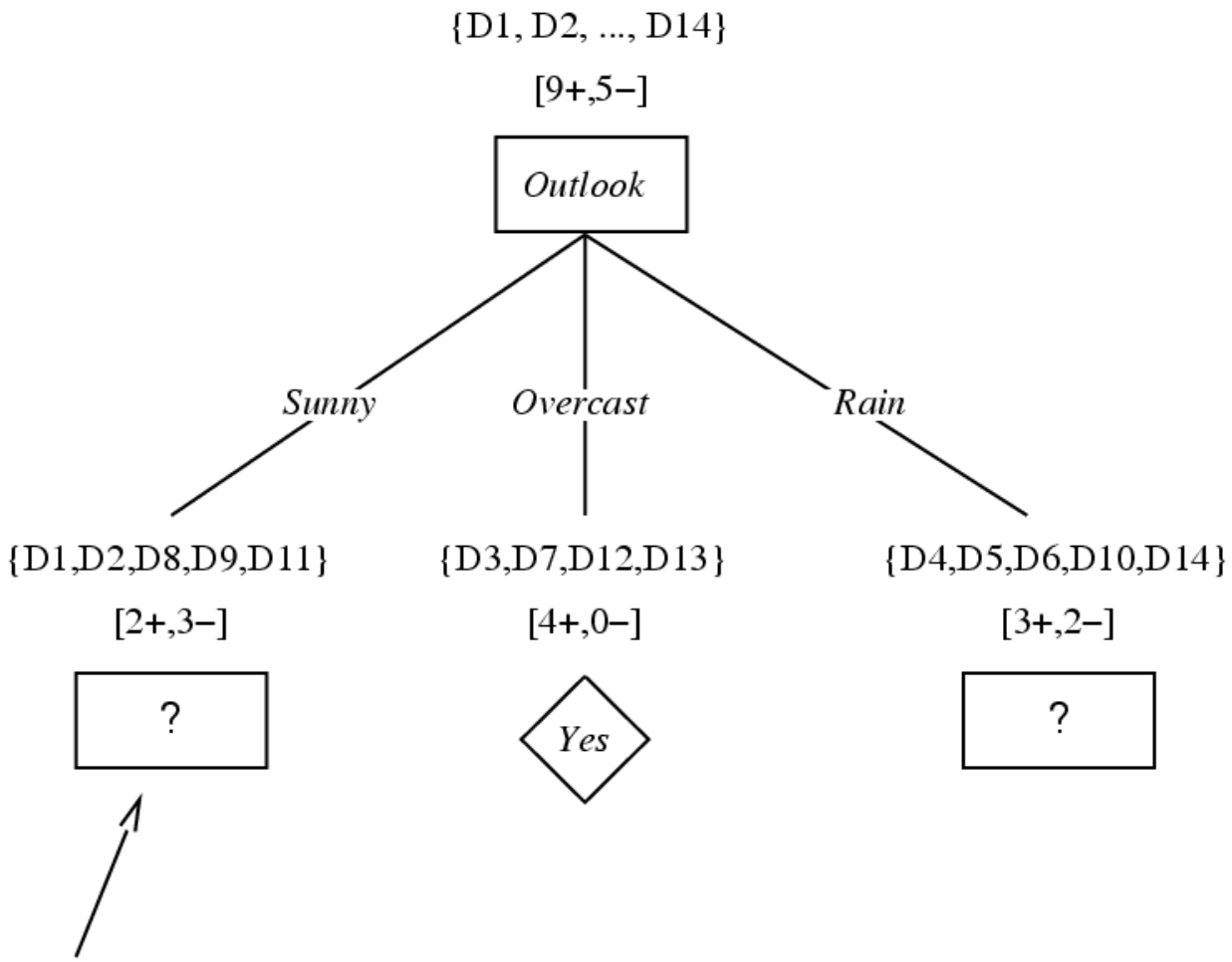
Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Slide by Tom Mitchell



Which attribute should be tested here?

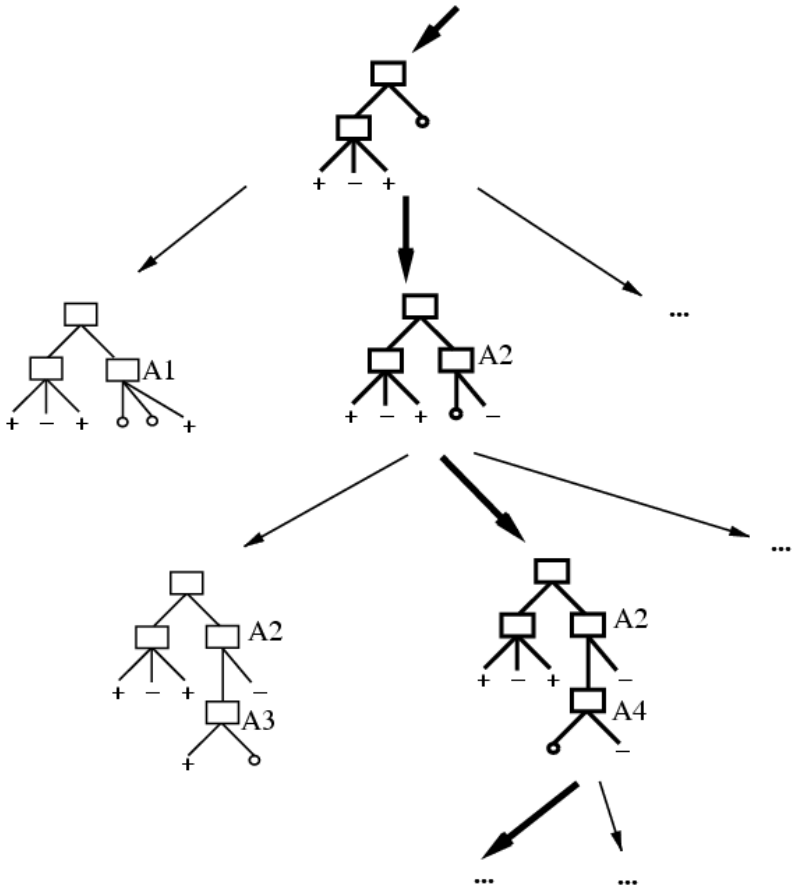
S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Which Tree Should We Output?



- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

Occam's razor: prefer the simplest hypothesis that fits the data

Slide by Tom Mitchell

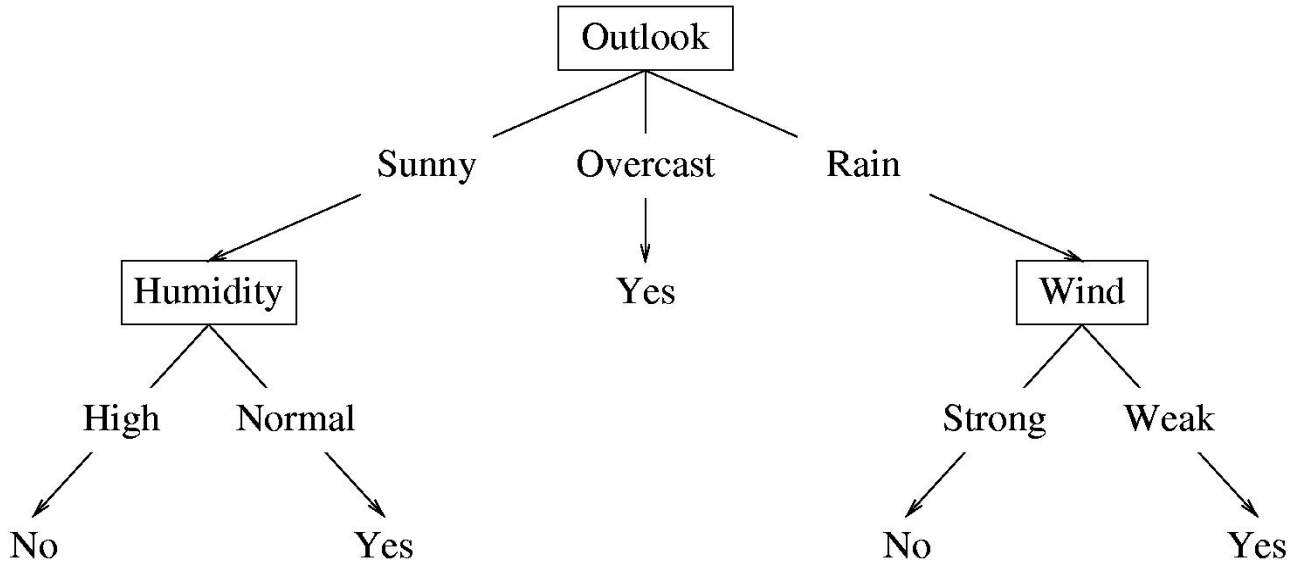
How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

Based on Slide from M. desJardins & T. Finin

Summary of Decision Trees (so far)



- Decision tree induction → choose the best attribute
  - Choose split via information gain
  - Build tree greedily, recursing on children of split
  - Stop when we achieve homogeny
    - i.e., when all instance in a child have the same class

Summary of Decision Trees (so far)

Information Gain: Mutual information of attribute A and the class variable of data set X

$$InfoGain(X, A) = H(X) - H(X \mid A)$$

$$= H(X) - \sum_{v \in values(A)} \frac{|\{x \in X \mid x_A = v\}|}{|X|} \times H(\{x \in X \mid x_A = v\})$$

fraction of instances with value v in attribute A

entropy of those instances

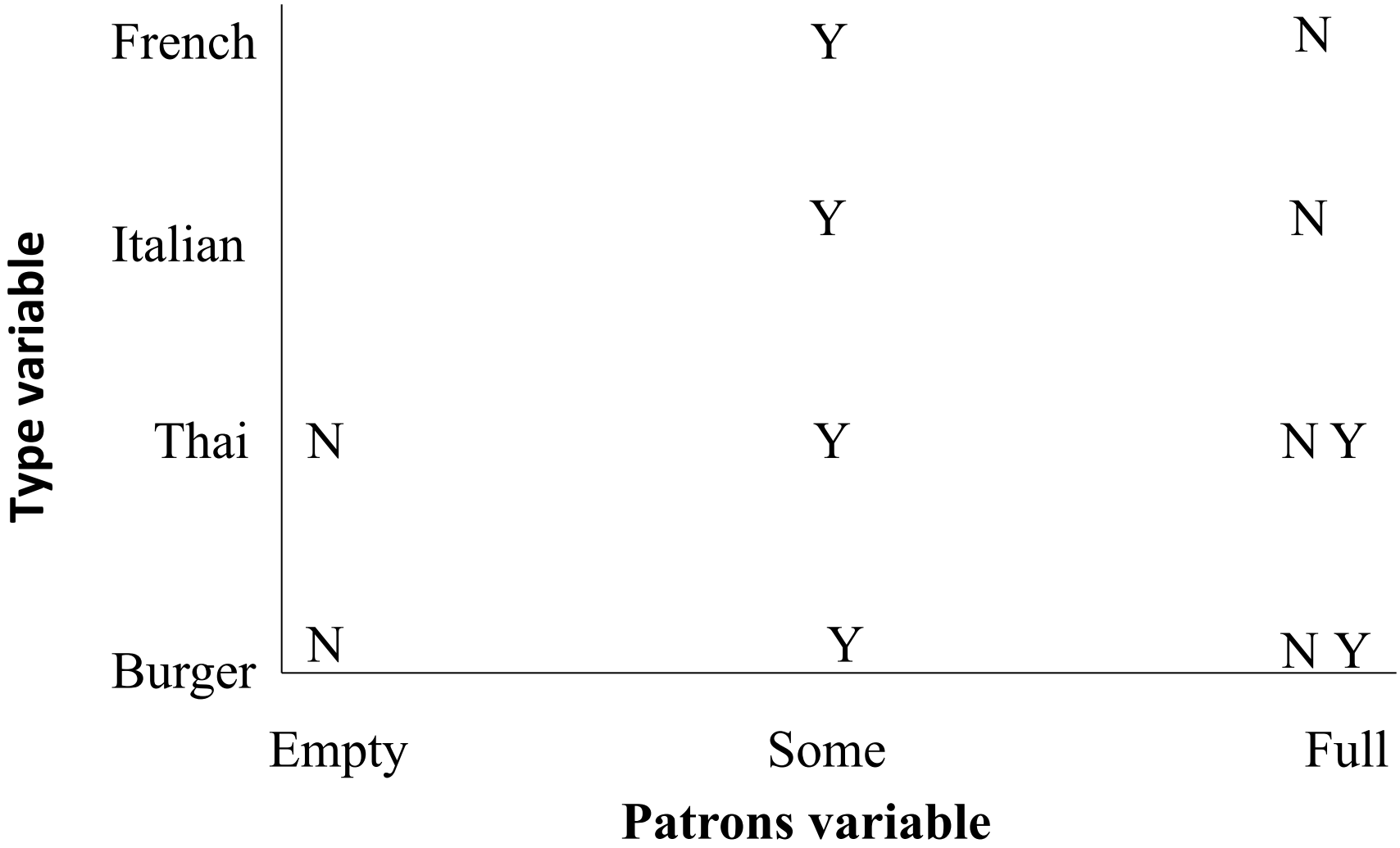
Entropy:

$$H(X) = -\sum_{c \in Classes} \frac{|\{x \in X \mid class(x) = c\}|}{|X|} \log_2 \frac{|\{x \in X \mid class(x) = c\}|}{|X|}$$

fraction of instances of class c

Restaurant Example

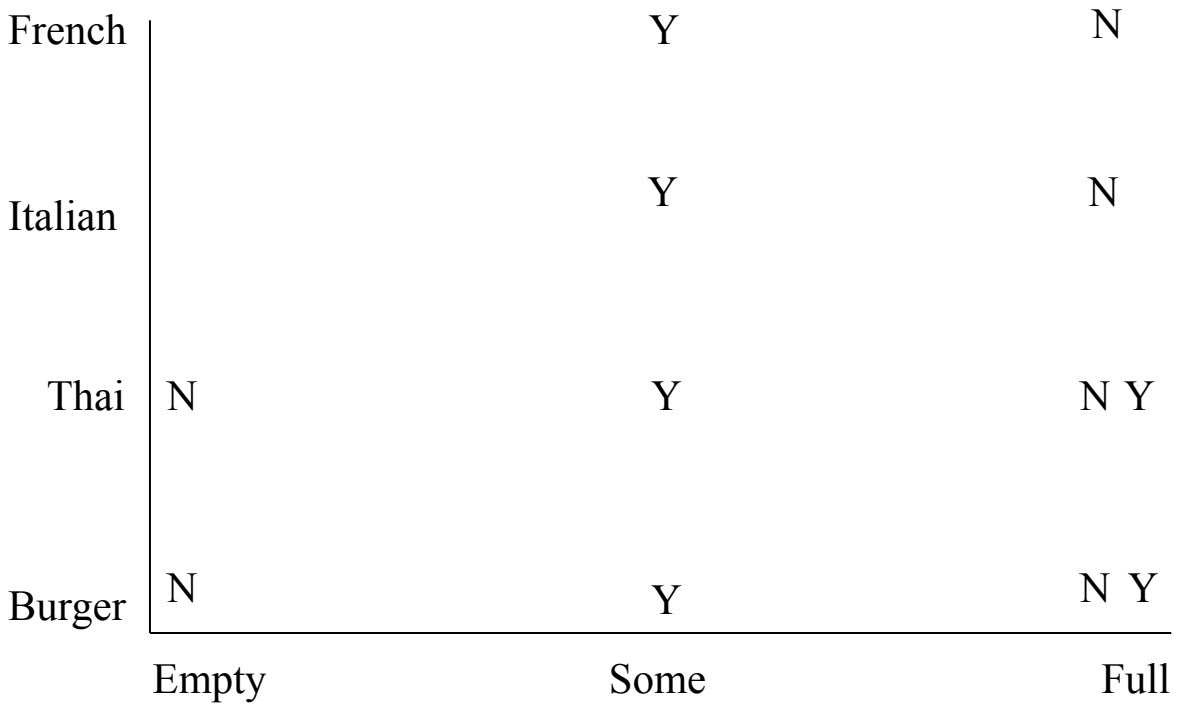
Random: Patrons or Wait-time; **Least-values**: Patrons; **Most-values**: Type; **Max-gain**: ???



Computing information gain

I(X) = ?

I (Pat, X) = ?



I (Type, X) = ?

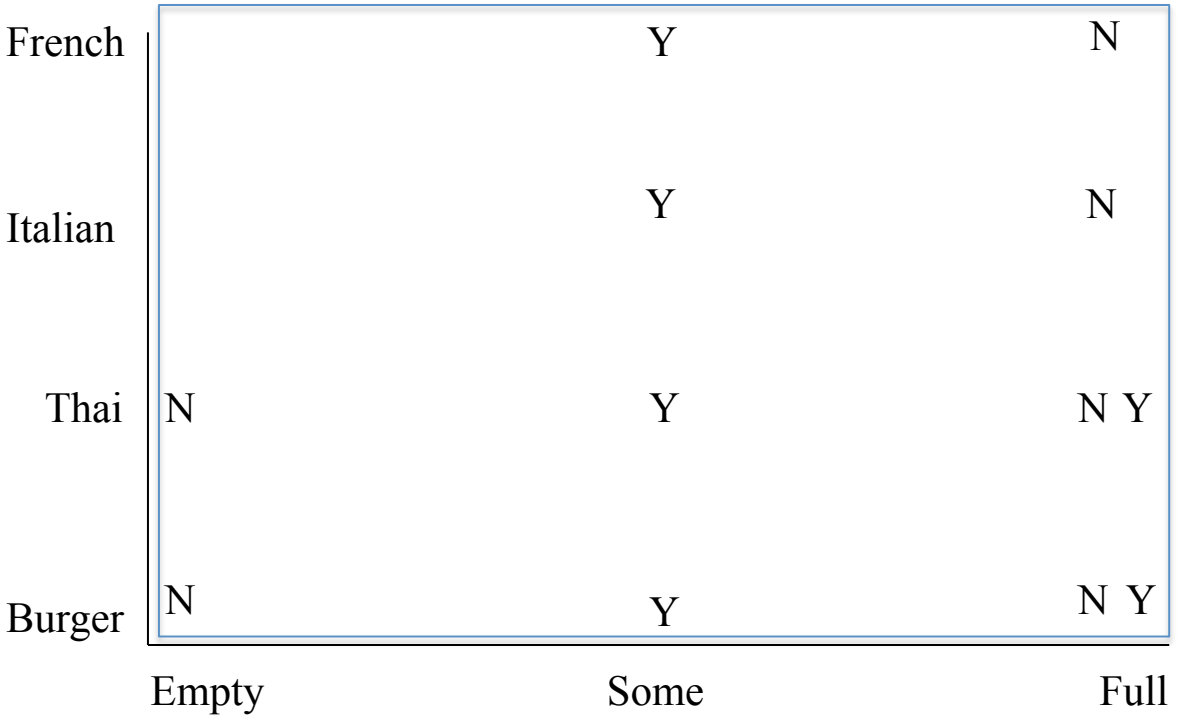
Gain (Pat, X) = ?

Gain (Type, X) = ?

Computing information gain

I(X) =
- (.5 log .5 + .5 log .5)
= .5 + .5 = 1

I (Pat, X) = ?



I (Type, X) = ?

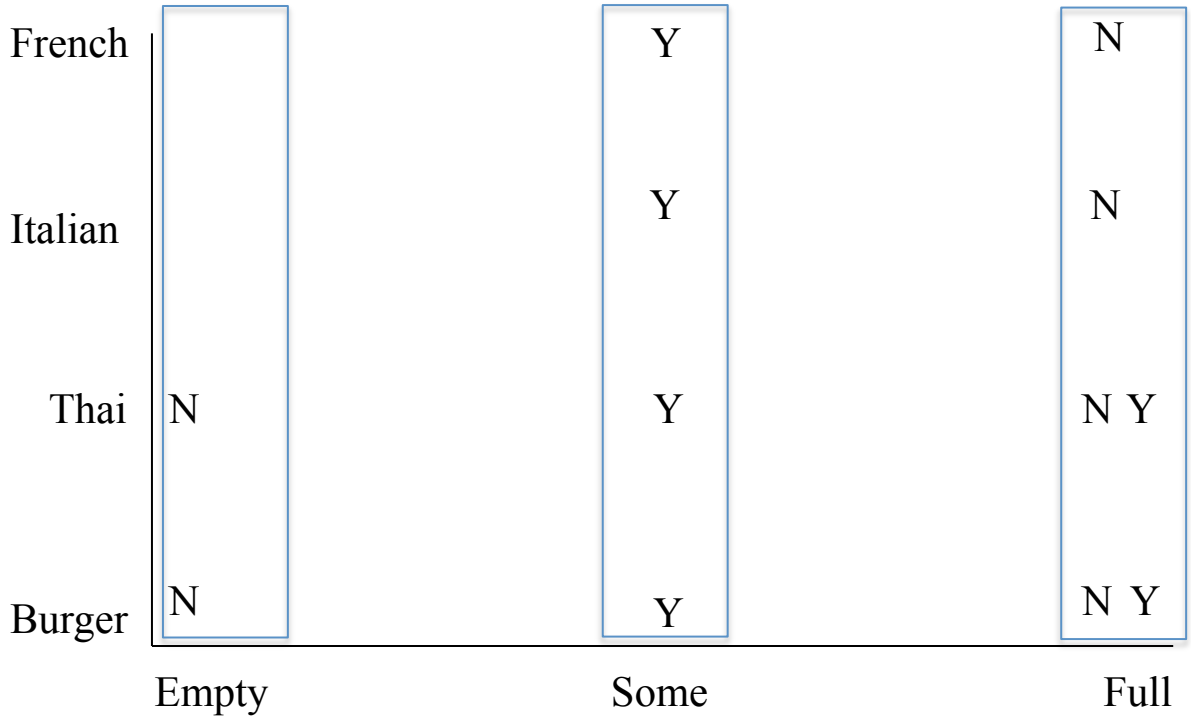
Gain (Pat, X) = ?
Gain (Type, X) = ?

Computing information gain

I(X) =
- (.5 log .5 + .5 log .5)
= .5 + .5 = 1

I (Pat, X) =
2/12 (0) + 4/12 (0) +
6/12 (- (4/6 log 4/6 +
2/6 log 2/6))
= 1/2 (2/3*.6 +
1/3*1.6)
= .47

I (Type, X) = ?



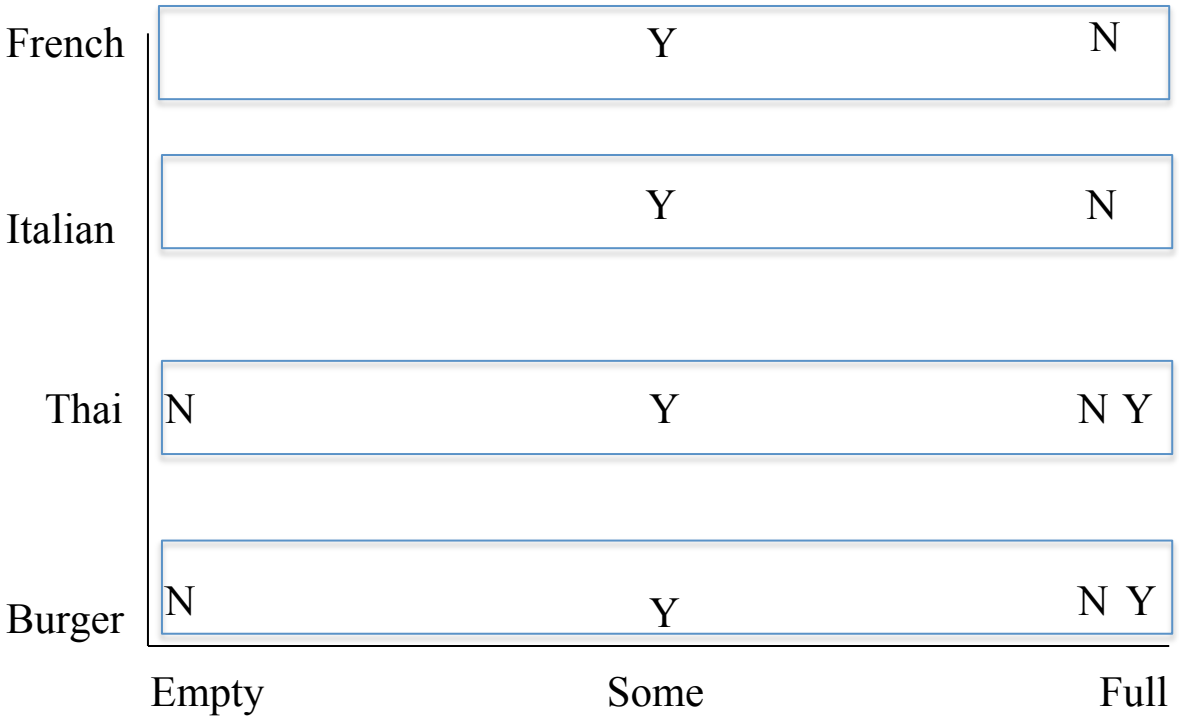
Gain (Pat, X) = ?
Gain (Type, X) = ?

Computing information gain

I(X) =
- (.5 log .5 + .5 log .5)
= .5 + .5 = 1

I (Pat, X) =
2/12 (0) + 4/12 (0) +
6/12 (- (4/6 log 4/6 +
2/6 log 2/6))
= 1/2 (2/3*.6 +
1/3*1.6)
= .47

I (Type, X) =
2/12 (1) + 2/12 (1) +
4/12 (1) + 4/12 (1) = 1



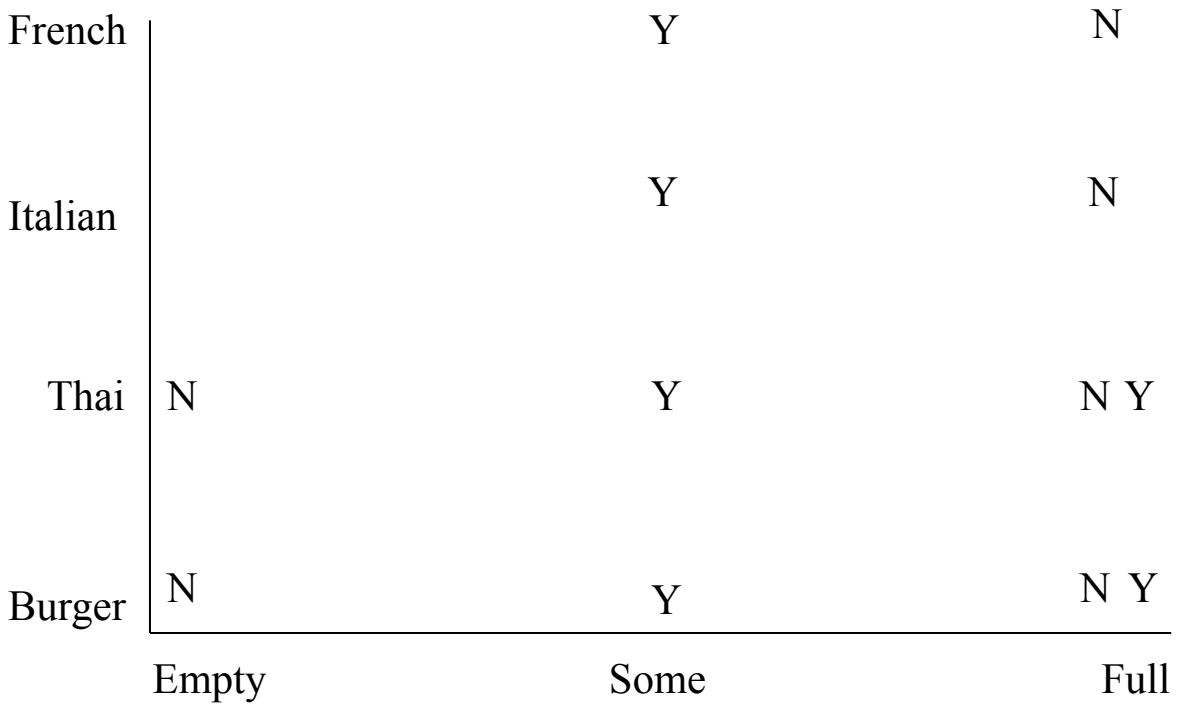
Gain (Pat, X) = ?
Gain (Type, X) = ?

Computing information gain

I(X) =
- (.5 log .5 + .5 log .5)
= .5 + .5 = 1

I (Pat, X) =
2/12 (0) + 4/12 (0) +
6/12 (- (4/6 log 4/6 +
2/6 log 2/6))
= 1/2 (2/3*.6 +
1/3*1.6)
= .47

I (Type, X) =
2/12 (1) + 2/12 (1) +
4/12 (1) + 4/12 (1) = 1



Gain (Pat, X) = 1 - .47 = .53

Gain (Type, X) = 1 – 1 = 0

Attributes with Many Values

- Problem
  - If attribute has many values, InfoGain() will select it
  - e.g., imagine using **`date = Jan_28_2011`** as an attribute

- Alternative approach: use GainRatio() instead

$$GainRatio(X, A) = \frac{InfoGain(X, A)}{SplitInformation(X, A)}$$

$$SplitInformation(X, A) = -\sum_{v \in values(A)} \frac{|X_v|}{|X|} \log_2 \frac{|X_v|}{|X|}$$

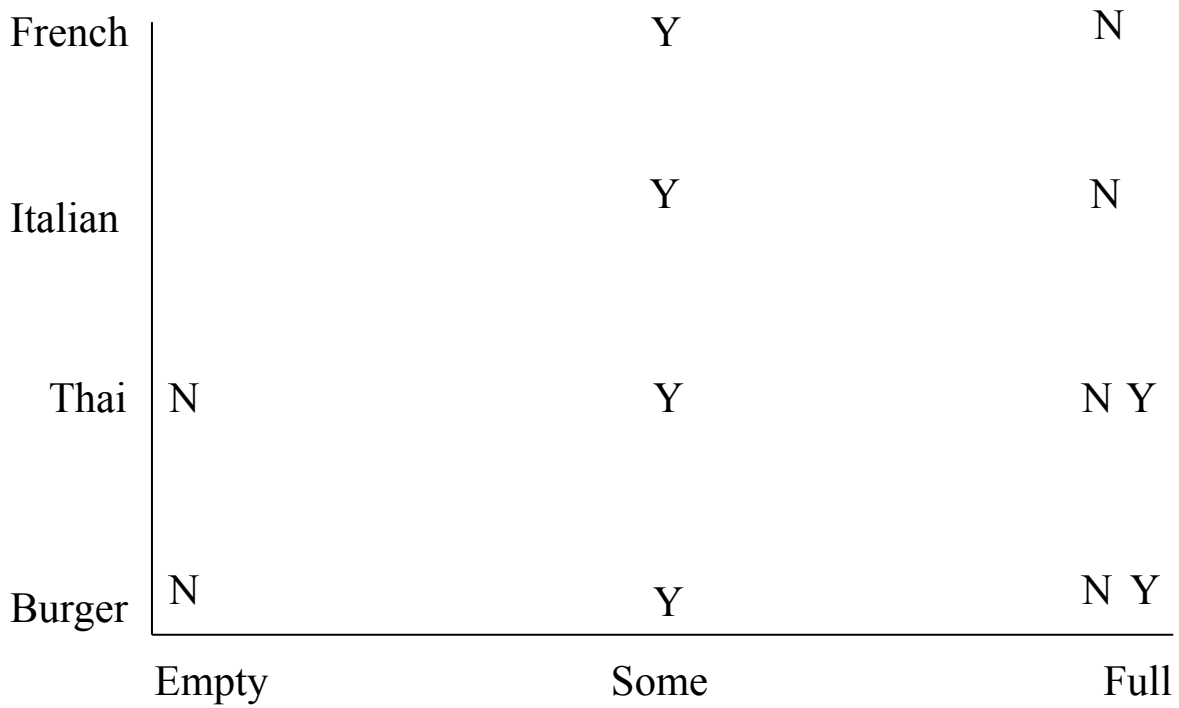
where X_v is a subset of X for which A has value v

Based on slide by Pedro Domingos

Computing Gain Ratio

Already computed:

- I(X) = 1
- I (Pat, X) = 0.47
- I (Type, X) = 1
- Gain (Pat, X) = 0.53
- Gain (Type, X) = 0



SplitInfo (Pat, X) = - (1/6 log 1/6 + 1/3 log 1/3 + 1/2 log 1/2)

= 1/6*2.6 + 1/3*1.6 + 1/2*1 = 1.47

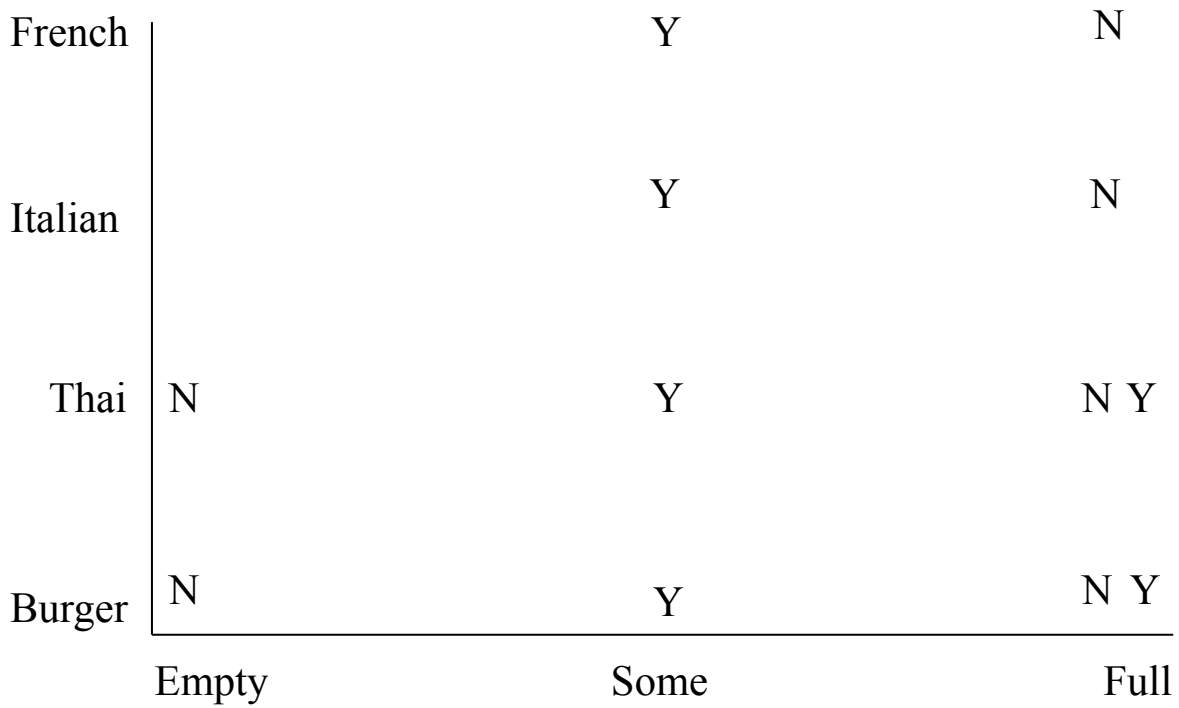
SplitInfo (Type, X) = 1/6 log 1/6 + 1/6 log 1/6 + 1/3 log 1/3 + 1/3 log 1/3

= 1/6*2.6 + 1/6*2.6 + 1/3*1.6 + 1/3*1.6 = 1.93

Computing Gain Ratio

Already computed:

- I(X) = 1
- I (Pat, X) = 0.47
- I (Type, X) = 1
- Gain (Pat, X) = 0.53
- Gain (Type, X) = 0



SplitInfo (Pat, X) = - (1/6 log 1/6 + 1/3 log 1/3 + 1/2 log 1/2)

= 1/6*2.6 + 1/3*1.6 + 1/2*1 = 1.47

SplitInfo (Type, X) = 1/6 log 1/6 + 1/6 log 1/6 + 1/3 log 1/3 + 1/3 log 1/3

= 1/6*2.6 + 1/6*2.6 + 1/3*1.6 + 1/3*1.6 = 1.93

GainRatio (Pat, X) = Gain (Pat, X) / SplitInfo(Pat, X) = 0.53 / 1.47 = 0.36

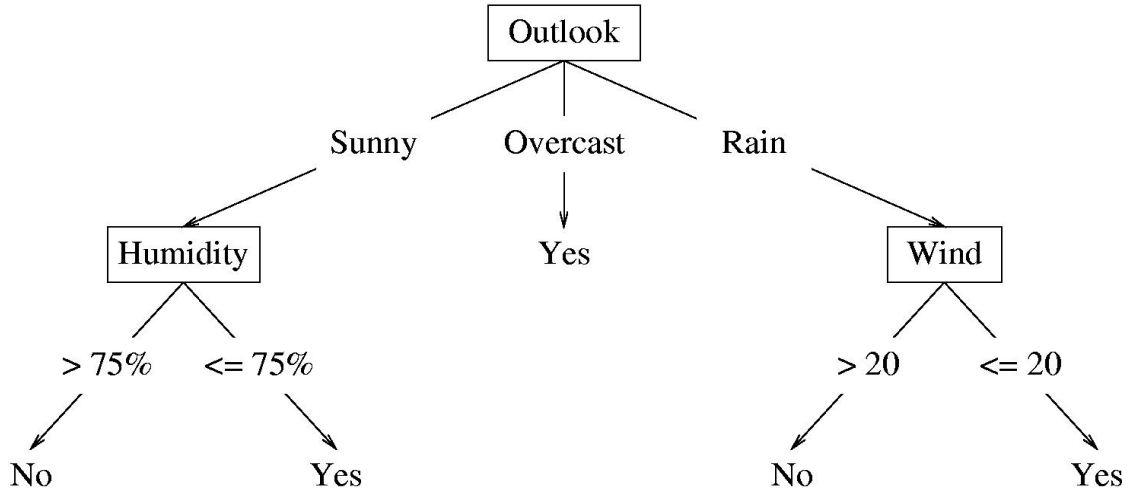
GainRatio (Type, X) = Gain (Type, X) / SplitInfo (Type, X) = 0 / 1.93 = 0

Extensions of ID3

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

Based on Slide from M. desJardins & T. Finin

Real-Valued Features



- Change to binary splits by choosing a threshold
- One method:
  - Sort instances by value, identify adjacencies with different classes

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

candidate splits

  - Choose among splits by InfoGain()

Unknown Attribute Values

What if some examples are missing values of *A*?

Use training example anyway, sort through tree:

- If node *n* tests *A*, assign most common value of *A* among other examples sorted to node *n*
- Assign most common value of *A* among other examples with same class label
- Assign probability p_i to each possible value v_i of *A.*

  Assign fraction p_i of example to each descendent of tree

Classify new examples in same fashion

Slide by Pedro Domingos

Noisy Data

- Many kinds of "noise" can occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
  - The instance was labeled incorrectly (+ instead of -)

- Also, some attributes are irrelevant to the decision-making process
  - e.g., color of a die is irrelevant to its outcome

Based on Slide from M. desJardins & T. Finin

Overfitting

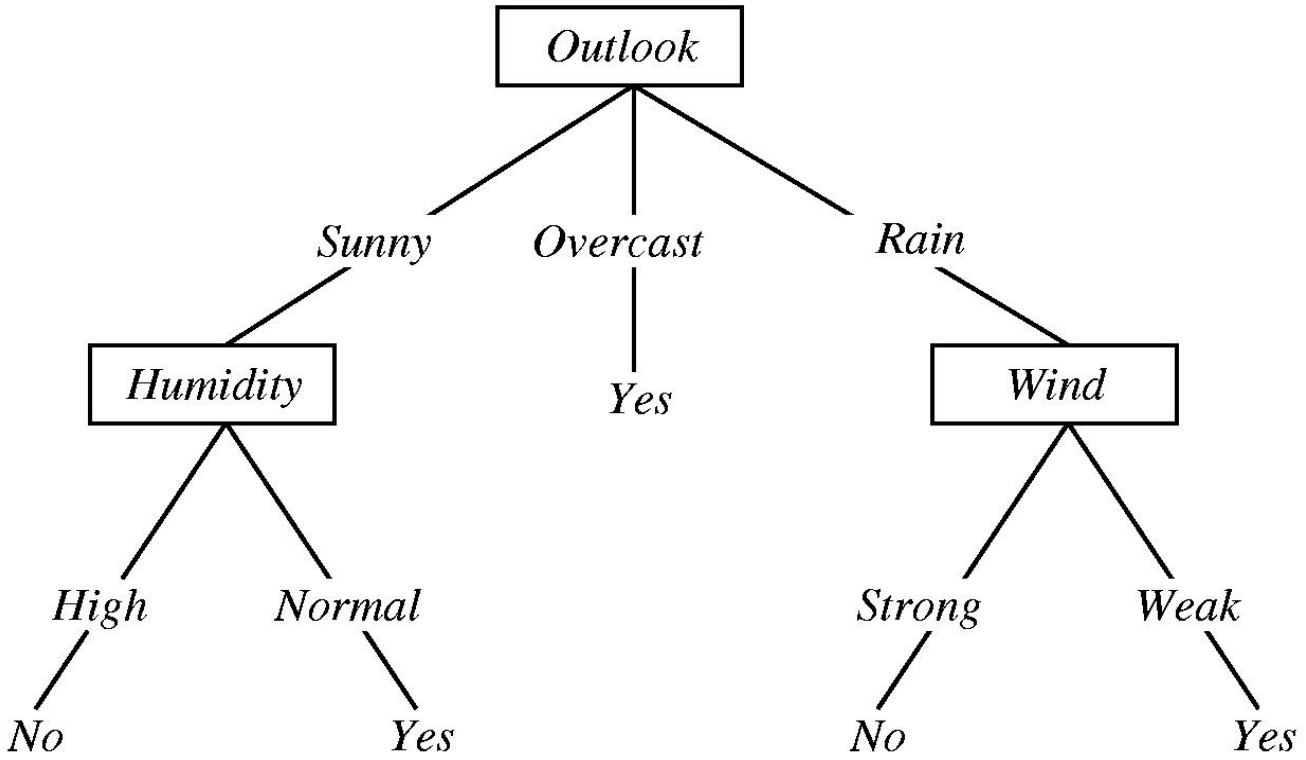
- Irrelevant attributes can result in *overfitting* the training example data
  - If hypothesis space has many dimensions (large number of attributes), we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features

- If we have too little training data, even a reasonable hypothesis space will ‘overfit’

Based on Slide from M. desJardins & T. Finin

Overfitting in Decision Trees

Consider adding a noisy training example to the following tree:



What would be the effect of adding:

<outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No> ?

Based on Slide by Pedro Domingos

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

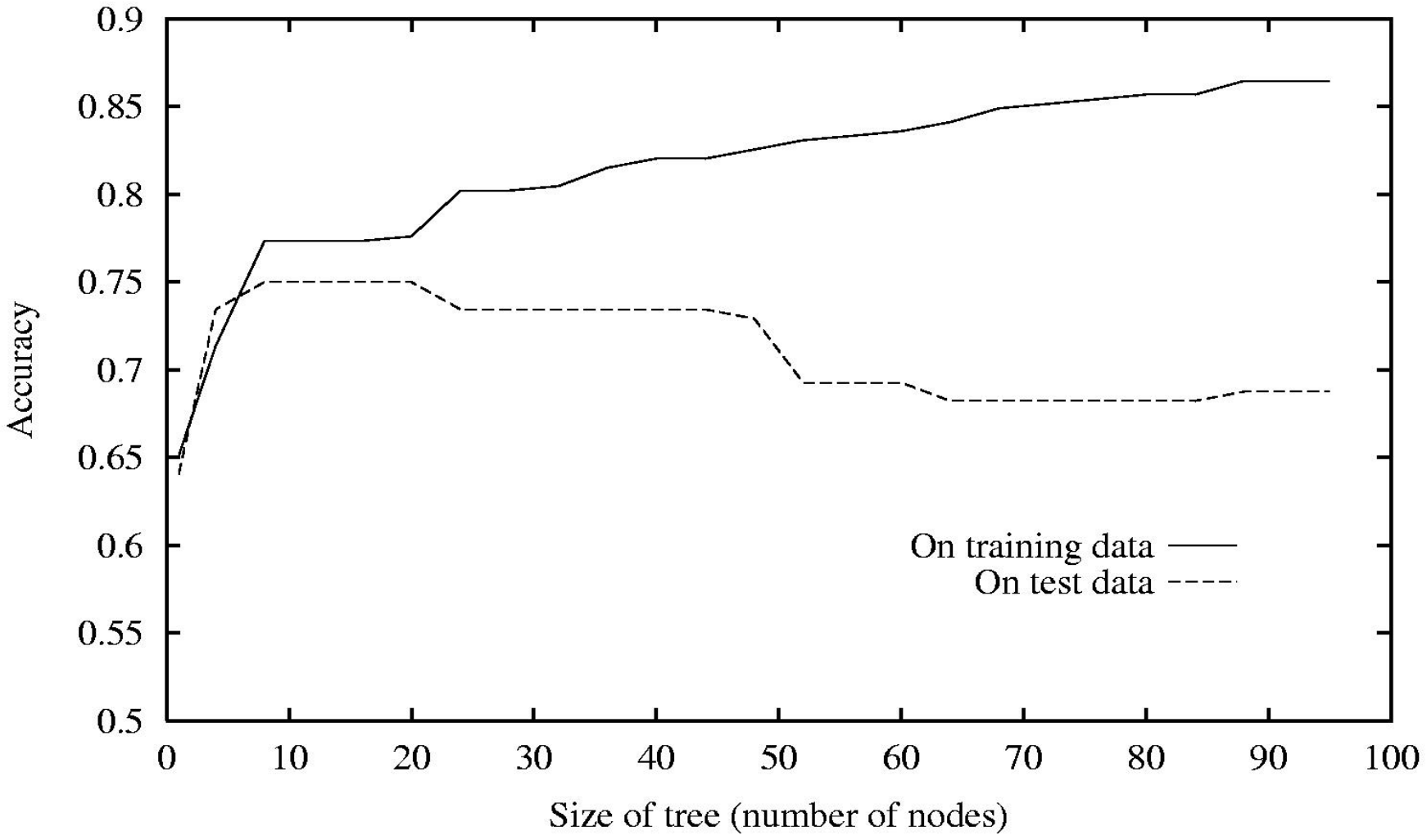
$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Slide by Pedro Domingos

Overfitting in Decision Tree Learning



Slide by Pedro Domingos

Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)
- Grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

Based on Slide by Pedro Domingos

Reduced-Error Pruning

Split data into *training* and *validation* sets

Grow tree based on *training set*

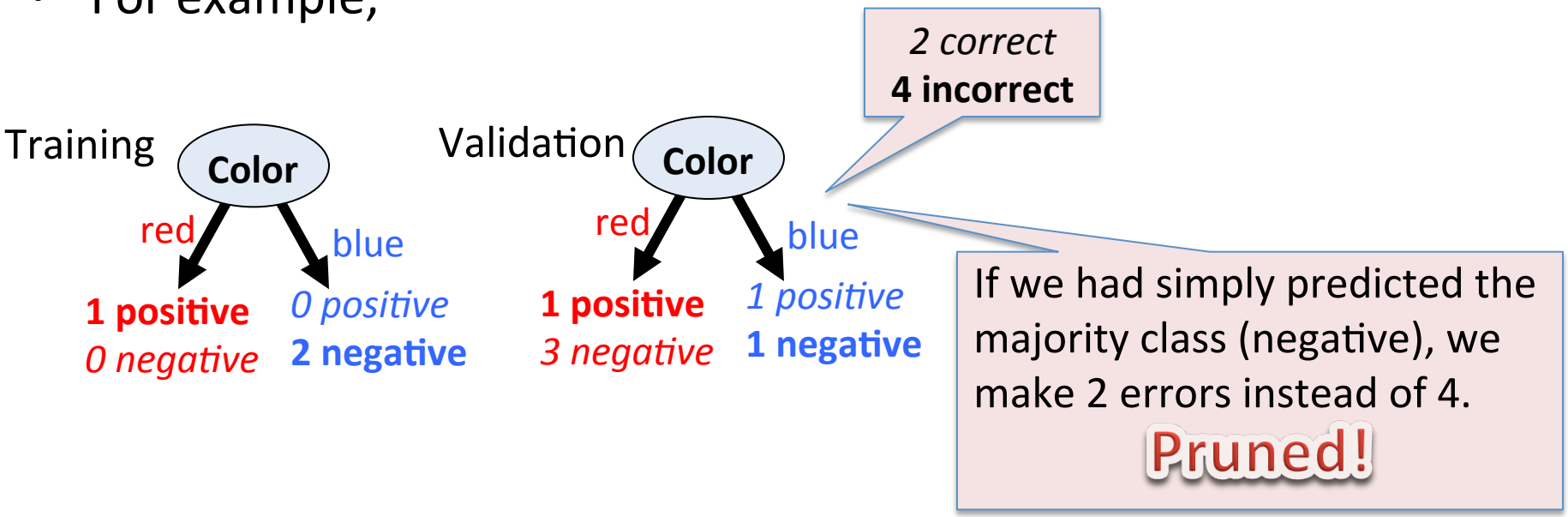
Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the node that most improves *validation set* accuracy

Slide by Pedro Domingos

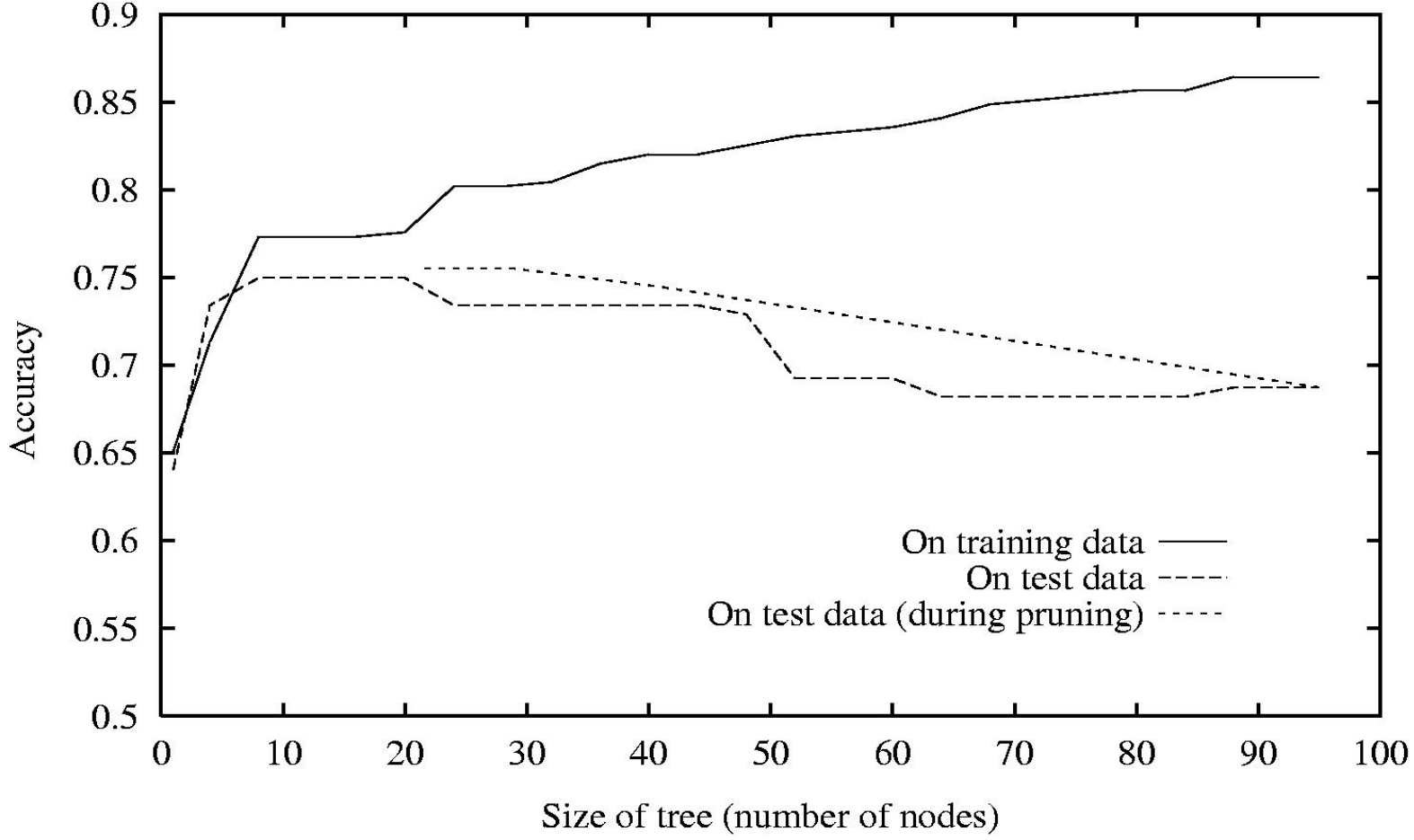
Pruning Decision Trees

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.
- For example,



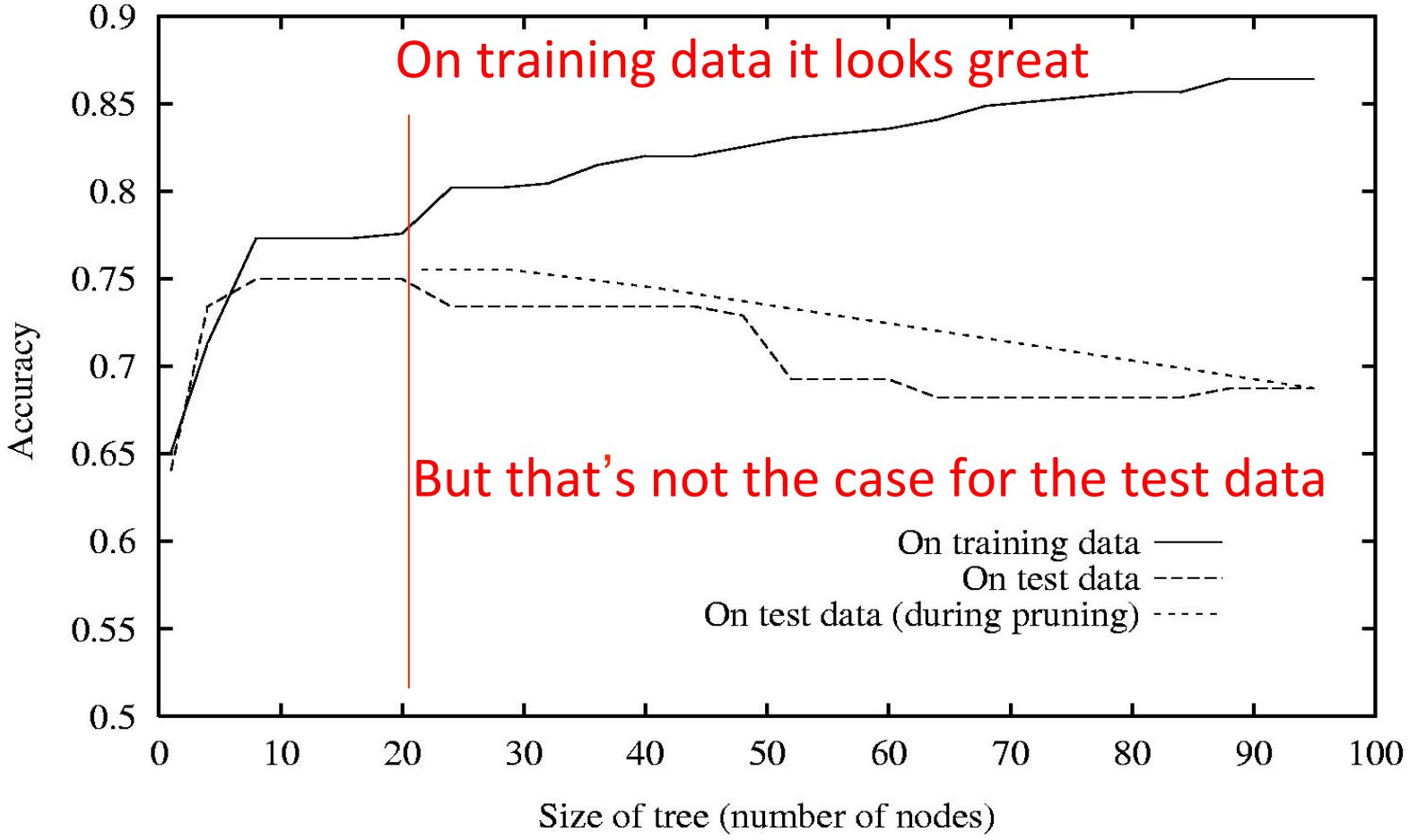
Based on Example from M. desJardins & T. Finin

Effect of Reduced-Error Pruning



Based on Slide by Pedro Domingos

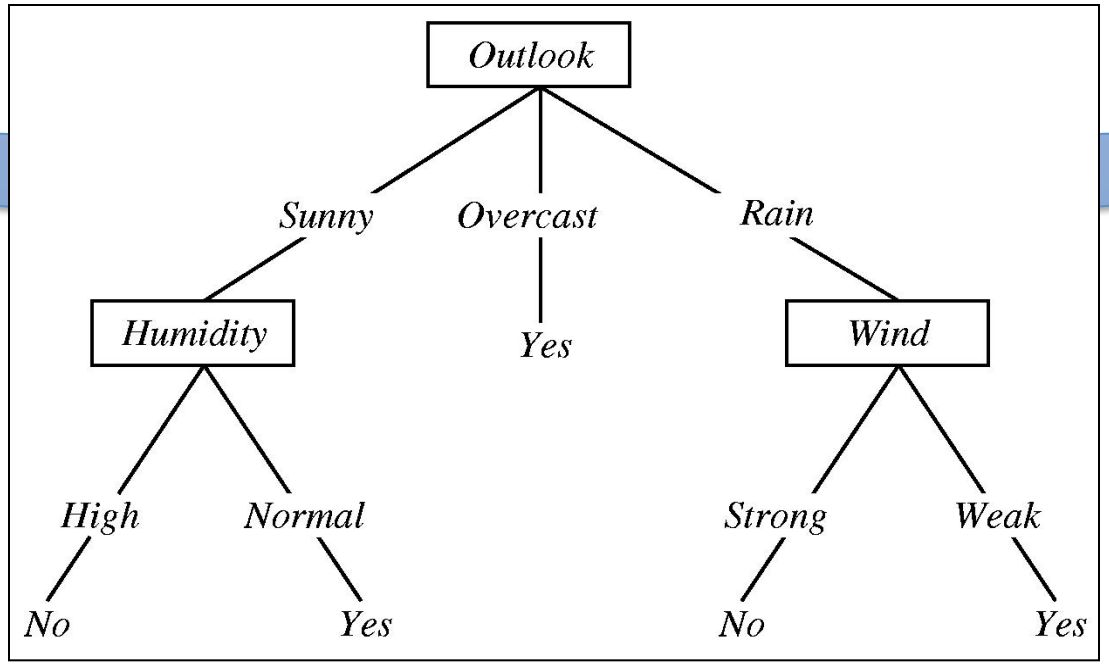
Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

Based on Slide by Pedro Domingos

Converting a Tree to Rules



IF (*Outlook* = *Sunny*) AND (*Humidity* = *High*)
THEN *PlayTennis* = *No*

IF (*Outlook* = *Sunny*) AND (*Humidity* = *Normal*)
THEN *PlayTennis* = *Yes*

. . .

Converting Decision Trees to Rules

- It is easy to derive rules from a decision tree: write a rule for each path from the root to a leaf

  (*Outlook* = *Sunny*) AND (*Humidity* = *High*) → *PlayTennis* = *No*

- To simplify the resulting rule set:
  - Let LHS be the left-hand side of a rule
  - LHS' obtained from LHS by eliminating some conditions
  - Replace LHS by LHS' in this rule if the subsets of the training set satisfying LHS and LHS' are equal
  - A rule may be eliminated by using meta-conditions such as "if no other rule applies"

Based on Slide from M. desJardins & T. Finin

Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

Scaling Up

- ID3, C4.5, etc.: assumes that data fits in memory
  (OK for up to hundreds of thousands of examples)

- SPRINT, SLIQ: multiple sequential scans of data
  (OK for up to millions of examples)

- VFDT: at most one sequential scan
  (OK for up to billions of examples)

Slide by Pedro Domingos

Comparison of Learning Methods

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

[Table 10.3 from Hastie, et al. Elements of Statistical Learning, 2nd Edition]

Summary: Decision Tree Learning

- Representation: decision trees
- Bias: prefer small decision trees
- Search algorithm: greedy
- Heuristic function: information gain or information content or others
- Overfitting / pruning

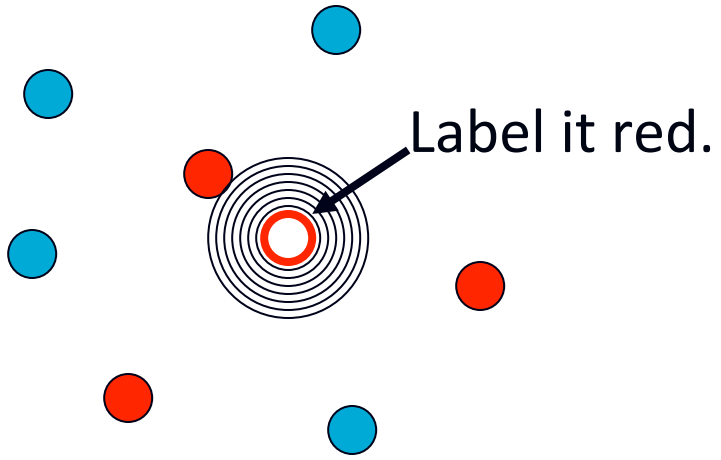
Slide by Pedro Domingos

Summary: Decision Tree Learning

- Widely used in practice
- Strengths include
  - Fast and simple to implement
  - Can convert to rules
  - Handles noisy data
- Weaknesses include
  - Univariate splits/partitioning using only one attribute at a time --- limits types of possible trees
  - Large decision trees may be hard to understand
  - Requires fixed-length feature vectors
  - Non-incremental (i.e., batch method)
  - Sacrifices predictive power

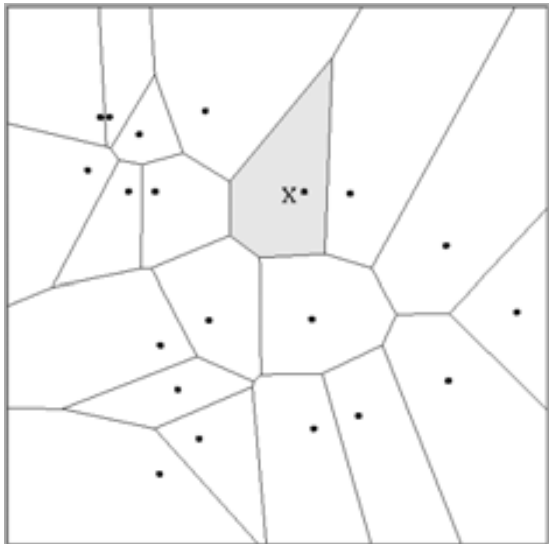
1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- Simple idea: label a new point the same as the closest known point



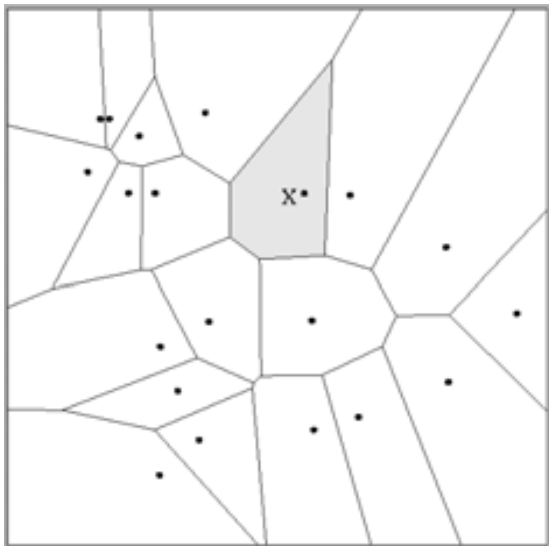
1-Nearest Neighbor

- A type of instance-based learning
  - Also known as "memory-based" learning
- Forms a Voronoi tessellation of the instance space

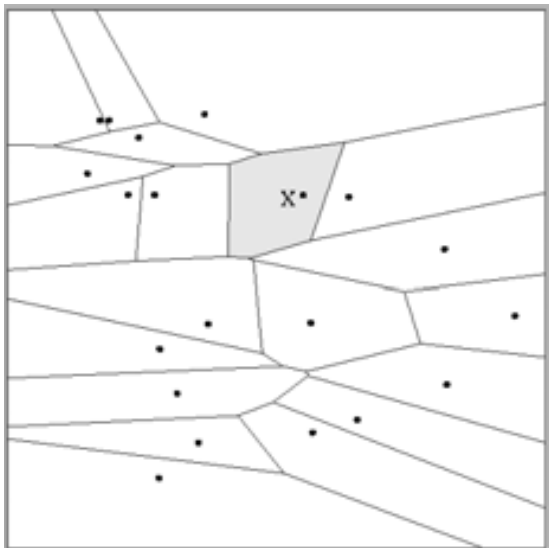


Distance Metrics

- Different metrics can change the decision surface



Dist(**a,b**) =$(a_1 - b_1)^2 + (a_2 - b_2)^2$



Dist(**a,b**) =$(a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

- Standard Euclidean distance metric:
  - Two-dimensional: Dist(a,b) = sqrt($(a_1 - b_1)^2 + (a_2 - b_2)^2$)
  - Multivariate: Dist(a,b) = sqrt($\sum (a_i - b_i)^2$)

Adapted from "Instance-Based Learning" lecture slides by Andrew Moore, CMU.

Four Aspects of an Instance-Based Learner:

1. A distance metric
2. How many nearby neighbors to look at?
3. A weighting function (optional)
4. How to fit with the local points?

Adapted from "Instance-Based Learning" lecture slides by Andrew Moore, CMU.

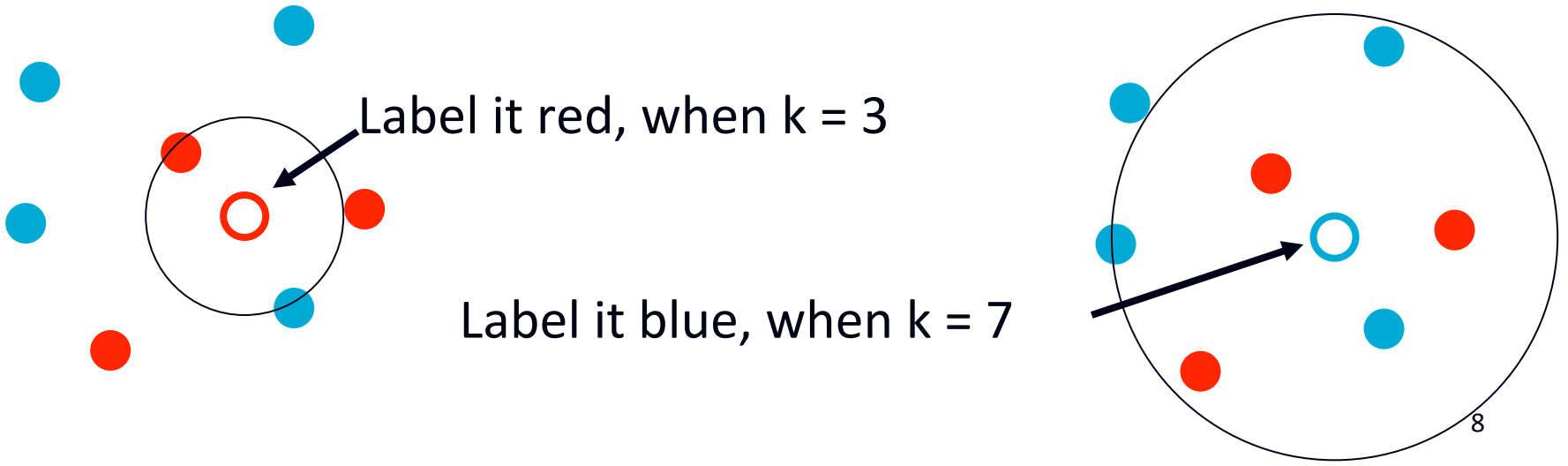
1-NN's Four Aspects as an Instance-Based Learner:

1. A distance metric
   - *Euclidian*
2. How many nearby neighbors to look at?
   - *One*
3. A weighting function (optional)
   - *Unused*
4. How to fit with the local points?
   - *Just predict the same output as the nearest neighbor.*

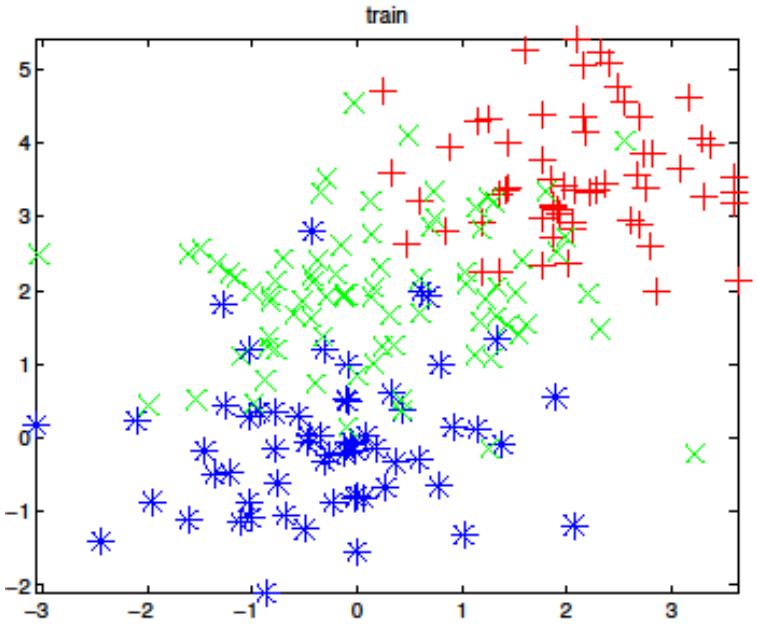
Adapted from "Instance-Based Learning" lecture slides by Andrew Moore, CMU.

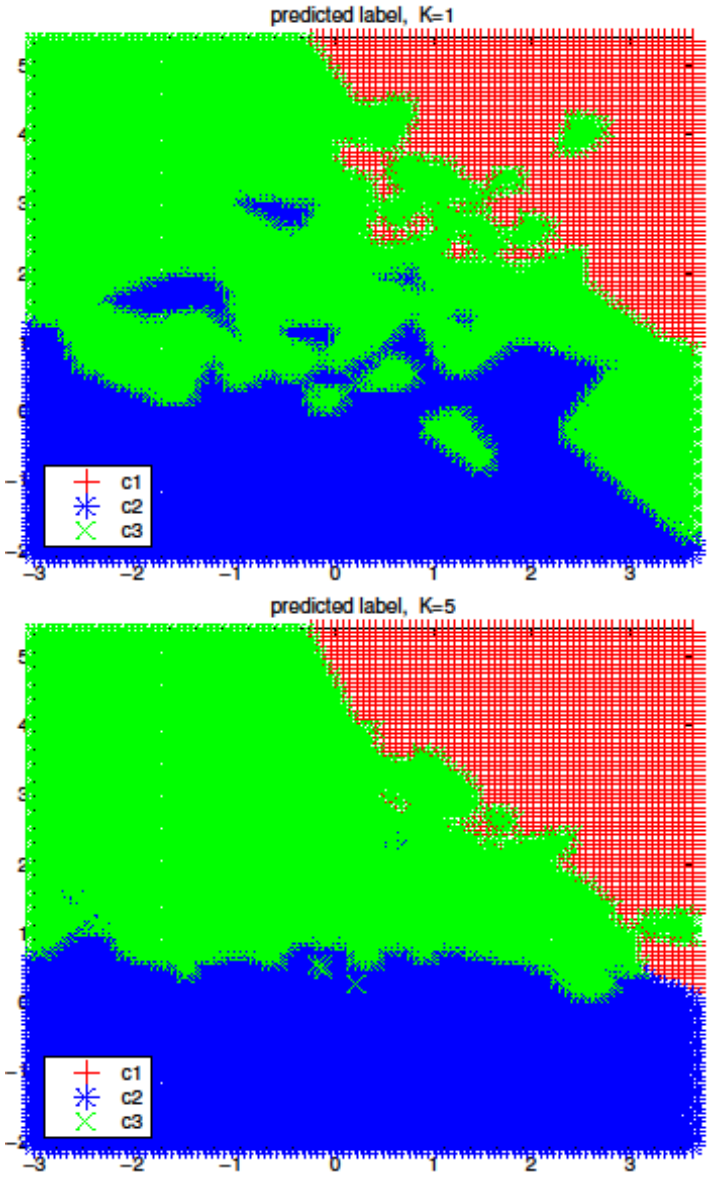
k – Nearest Neighbor

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its *k* nearest neighbors



KNN and Overfitting





- K increases → smoother Predictions
- K = N → majority label of the dataset

Today

- Decision Trees
- k-Nearest Neighbor