Data Mining and Analysis

ML@scale

CSCE 676 :: Fall 2019
Texas A&M University
Department of Computer Science & Engineering
Prof. James Caverlee

Resources

MMDS Chapter 12 + slides

Introduction to Data Mining Chapter 3

https://www-users.cs.umn.edu/~kumar001/dmbook/slides/chap3_basic_classification.pdf

PLANET: Massively Parallel Learning of Tree Ensembles with MapReduce, VLDB 2009

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36296.pdf

The Plan

Brief backgrounder: ML@scale

Decision Trees (classic!)

ML

Parallel Learning of Decision Trees

@scale

Supervised Learning

Given some data ==> “learn” a function to map from the **input** to the **output**

Given:

Training examples (x_i, $y_i = f(x_i)$) for an unknown function **f**

Find:

A good approximation to f

Other ML Paradigms

Supervised

Given **labeled** data {x,y}, learn f(x)=y

Unsupervised

Given only **unlabeled** data {x}, learn f(x)

Semi-supervised

Given **some labeled** {x,y} and **some unlabeled** {x}, learn f(x)=y

Active learning

When we predict f(x)=y, we then receive true y*

Transfer learning

Learn f(x) so that it works well on new domain f(z)

Supervised Learning

Task: Given data (X,Y) build a model f() to predict Y' based on X'

Strategy: Estimate y=f(x) on (X,Y)

Hope that the same f(x) also works to predict unknown Y'

This hope is "generalization"

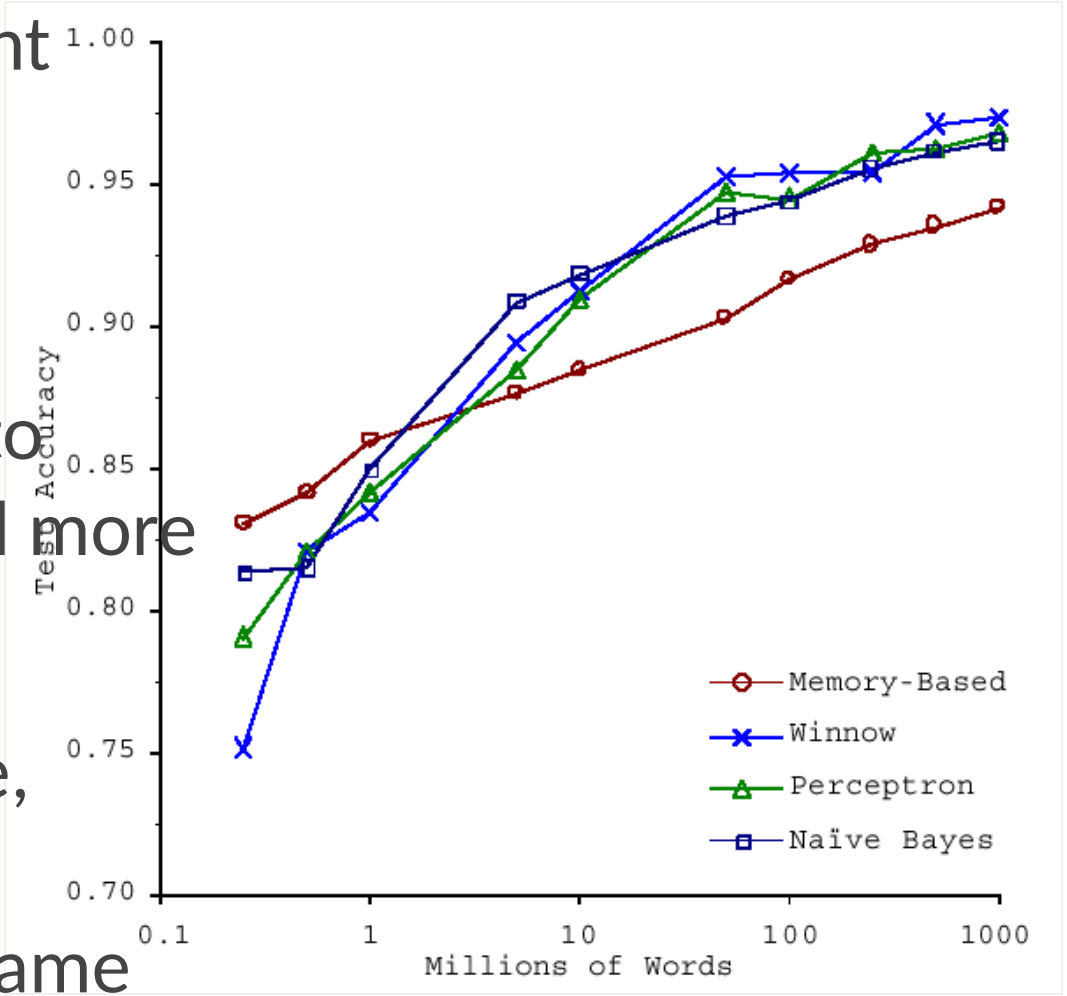
Overfitting: if f(x) predicts Y well but cannot predict Y'

Why Large-Scale ML?

In 2001, Microsoft researchers ran a test to evaluate 4 of different approaches to ML translation

Findings:

- Size of the dataset used to train the model mattered more than the model itself
- As the dataset grew large, performance difference between the models became small



Banko, M. and Brill, E. (2001) , "Scaling to Very Very Large Corpora for Natural Language Disambiguation"

Halevy, Norvig, and Pereira, IEEE Intelligent Systems 2009

The Unreasonable Effectiveness of Data

EXPERT OPINION

Contact Editor: **Brian Brannon,** bbrannon@computer.org

The Unreasonable Effectiveness of Data

Alon Halevy, Peter Norvig, and Fernando Pereira, *Google*

Eugene Wigner's article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences"[1] examines why so much of physics can be neatly explained with simple mathematical formulas

behavior. So, this corpus could serve as the basis of a complete model for certain tasks—if only we knew how to extract the model from the data.

Learning from Text at Web Scale

The biggest successes in natural-language-related

Revisiting the Unreasonable …

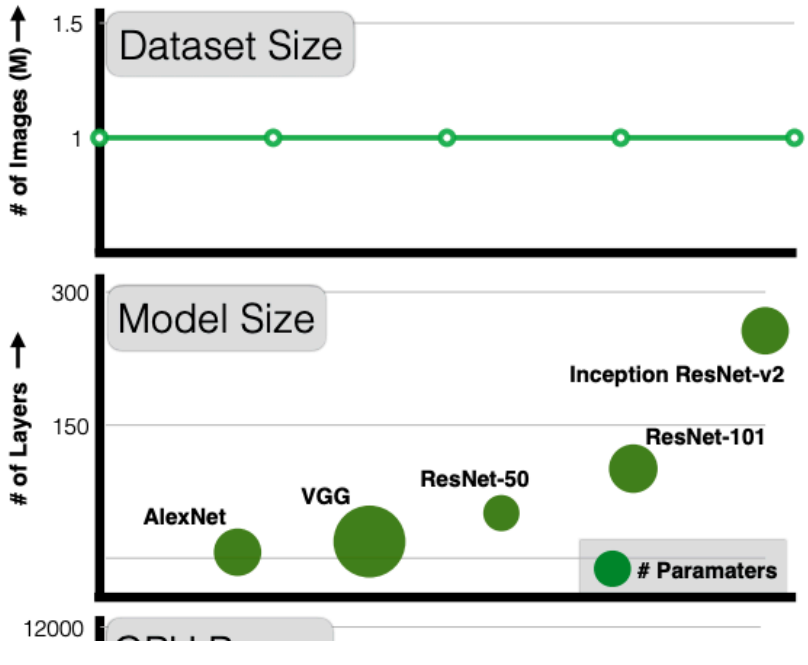
Revisiting Unreasonable Effectiveness of Data in Deep Learning Era

Chen Sun[1], Abhinav Shrivastava[1,2], Saurabh Singh[1], and Abhinav Gupta[1,2]

[1]Google Research
[2]Carnegie Mellon University

Abstract

The success of deep learning in vision can be attributed to: (a) models with high capacity; (b) increased computational power; and (c) availability of large-scale labeled data. Since 2012, there have been significant advances in representation capabilities of the models and computational capabilities of GPUs. But the size of the biggest dataset has surprisingly remained constant. What will happen if we increase the dataset size by 10× or 100×? This paper takes a step towards clearing the clouds of mystery surrounding the relationship between 'enormous data' and visual deep learning. By exploiting the JFT-300M dataset which has more than 375M noisy labels for 300M images, we inves-



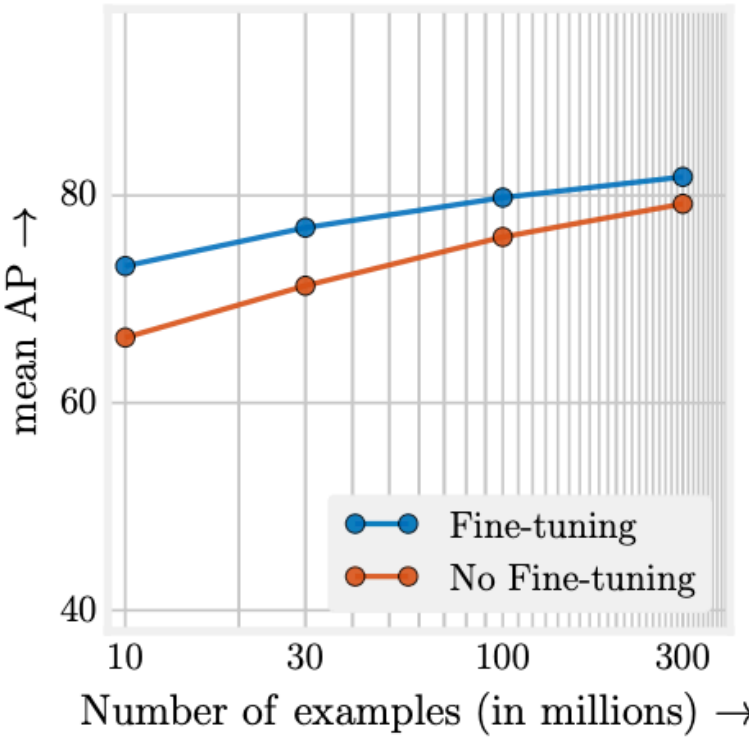
Revisiting the Unreasonable …

In 2017, Google revisited the same type of experiment with the latest Deep Learning models in computer vision

Findings:

- Performance increases logarithmically based on volume of training data
- Complexity of modern ML models (i.e., deep neural nets) allows for even further performance gains

Large datasets + large ML models => amazing results!!



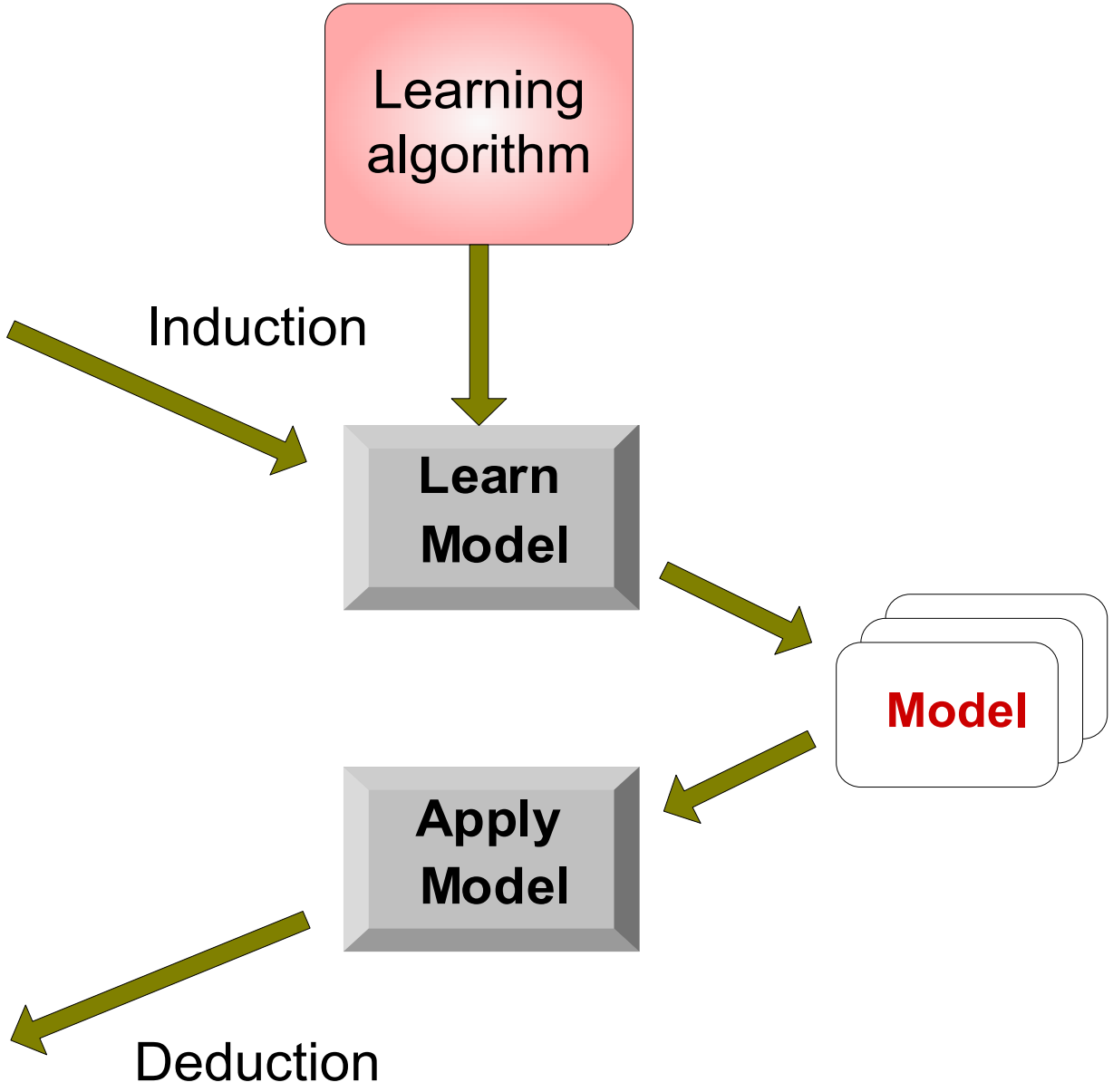
Decision Trees

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Classification Techniques

Decision trees ← today

Naive Bayes

Nearest Neighbors (kNN)

Support Vector Machines

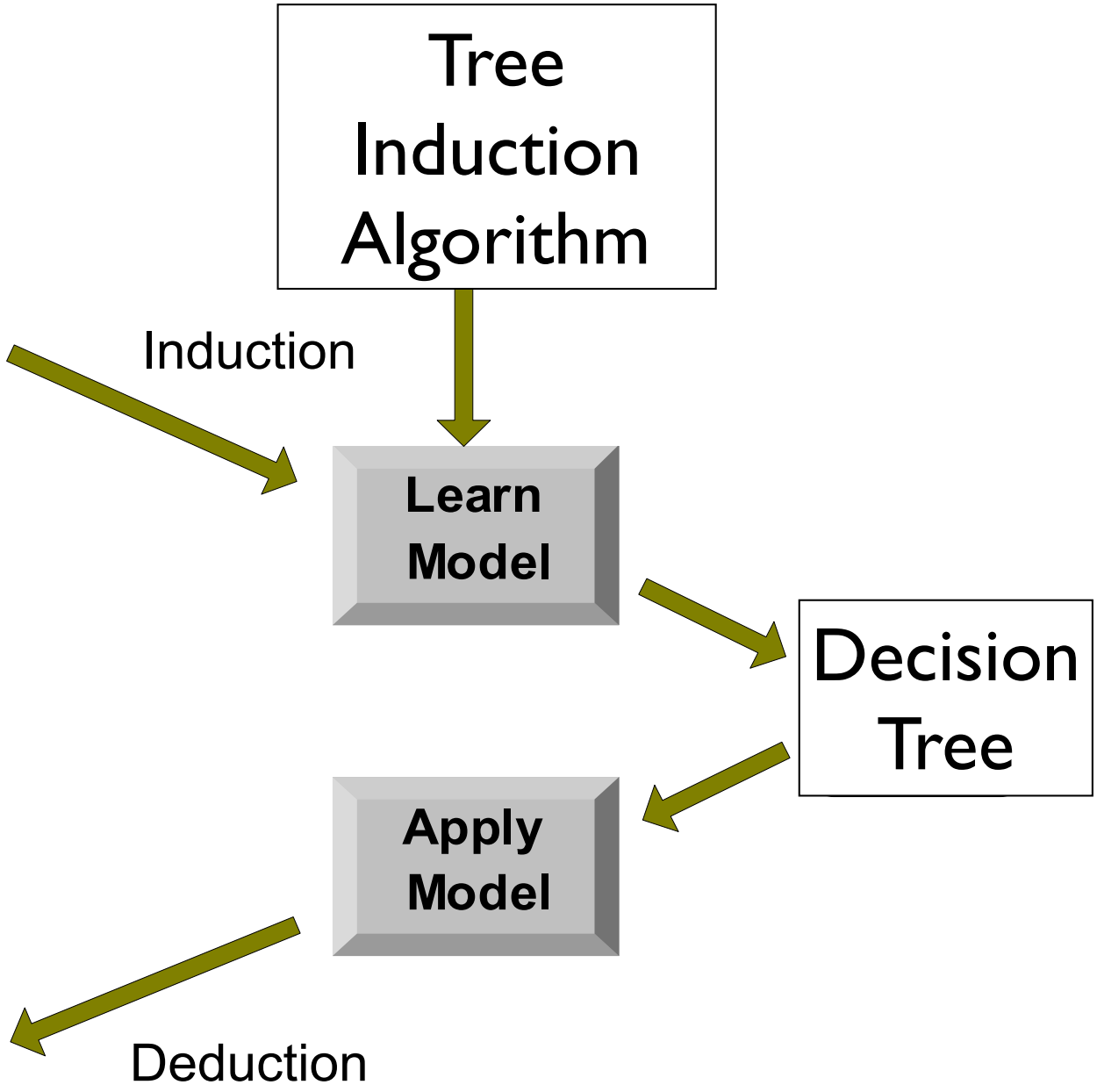
…

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



What is a Decision Tree?

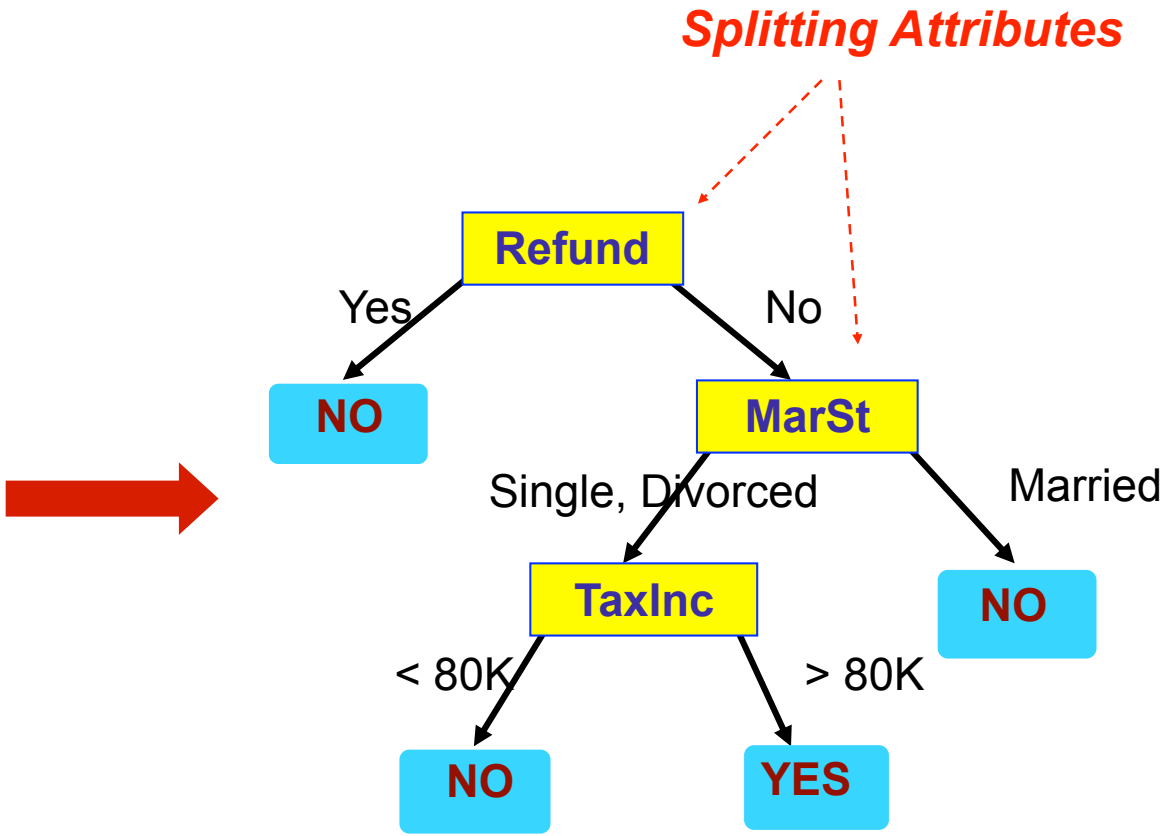
Hierarchical structure of **nodes** and **directed edges**

- **Root node:** no incoming edges; zero or more outgoing edges
- **Internal node:** one incoming edge; two or more outgoing edges
- **Leaf node:** one incoming edge; no outgoing edges; **Labeled with a class**

Example of a Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

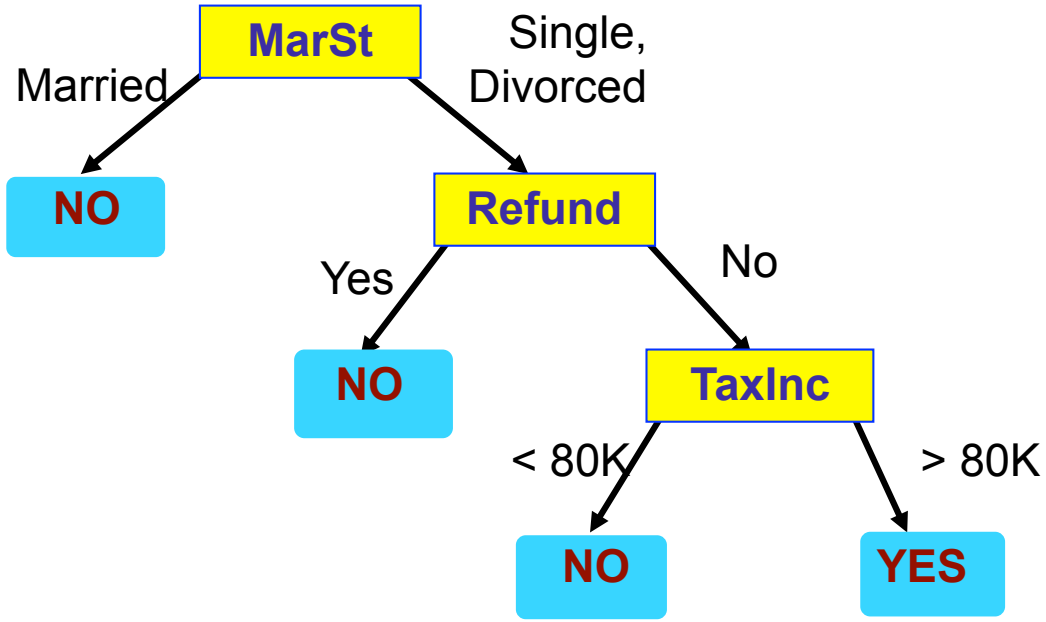
Training Data



Model: Decision Tree

Another Example of Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



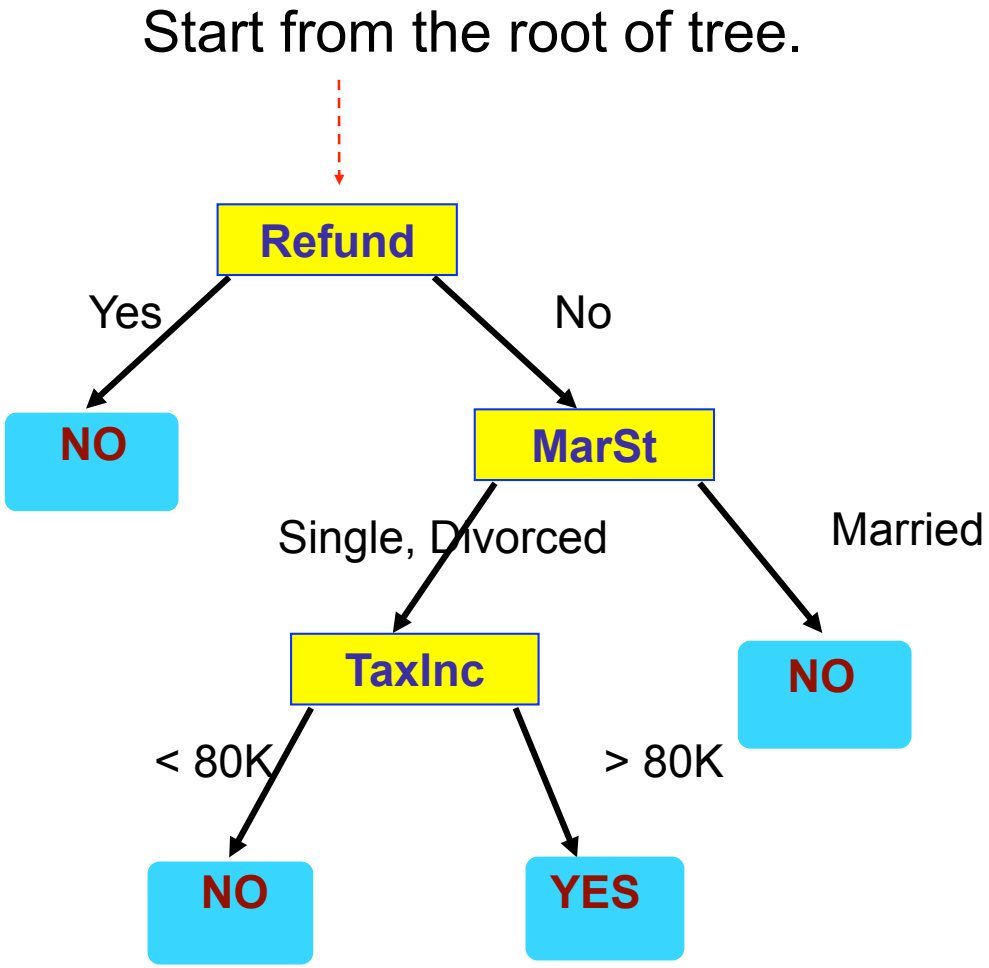
There could be more than one tree that fits the same data!

The Hope

The decision tree (or whatever classifier we use) **generalizes** to new data!!

So we can have confidence in it

Apply Model to Test Data



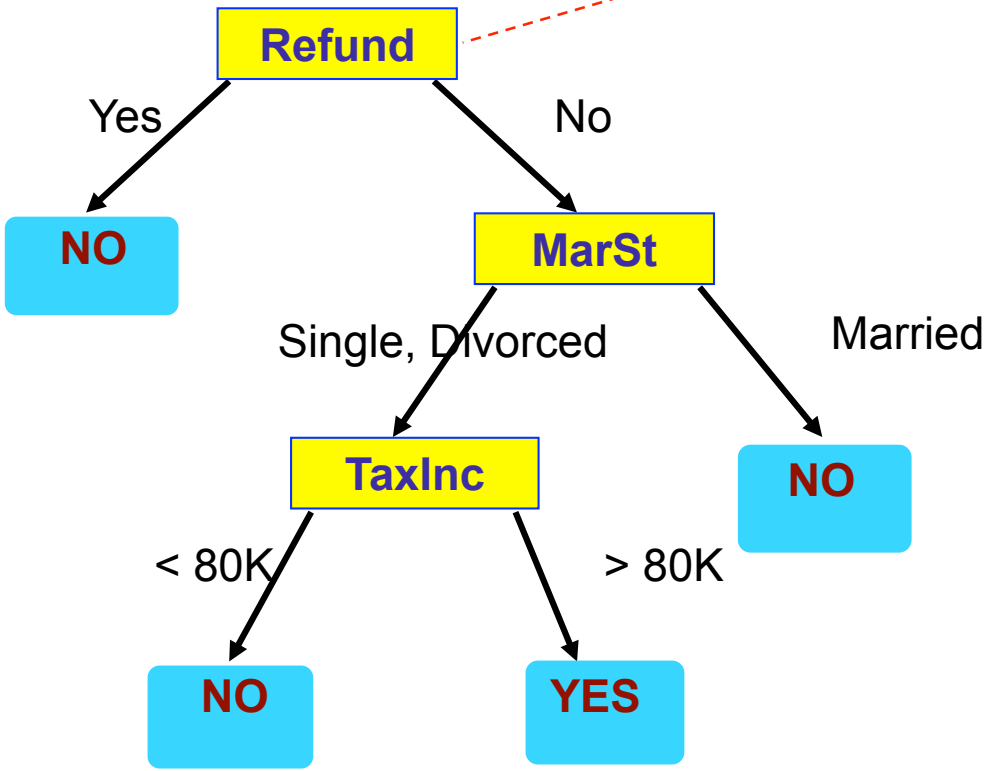
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

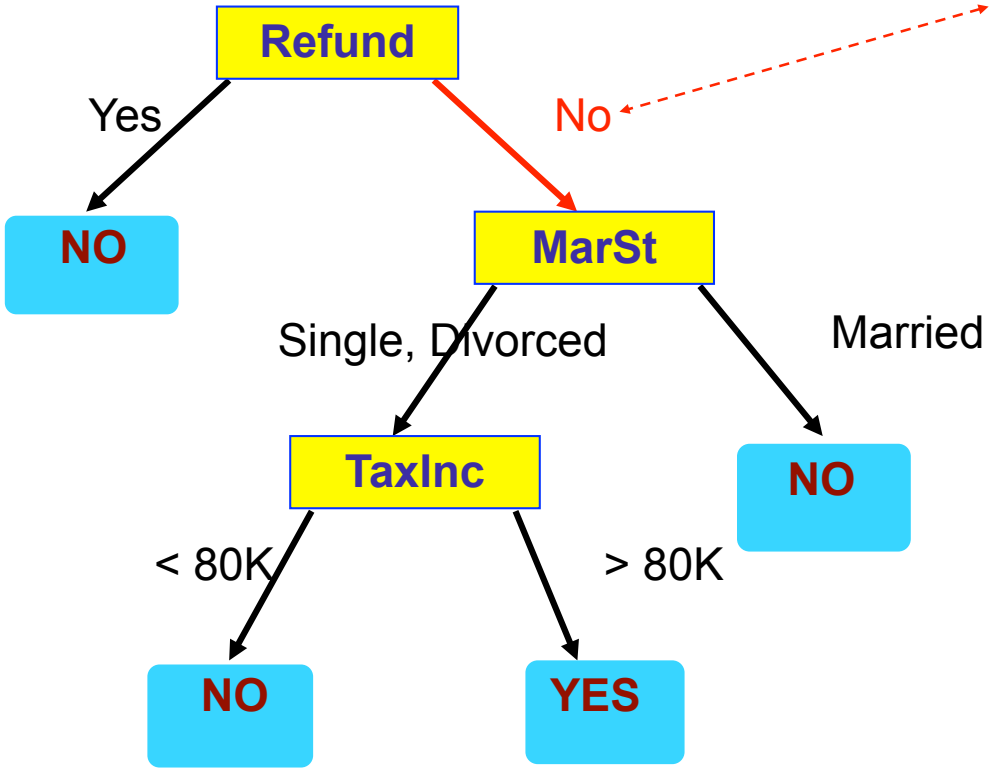
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

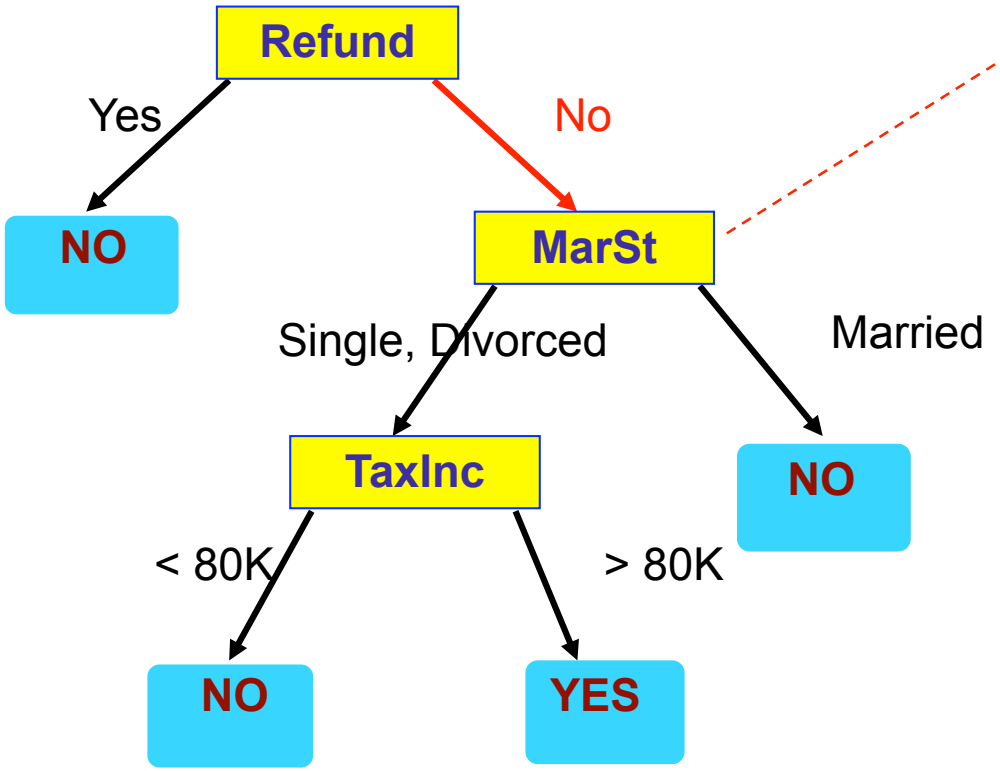
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

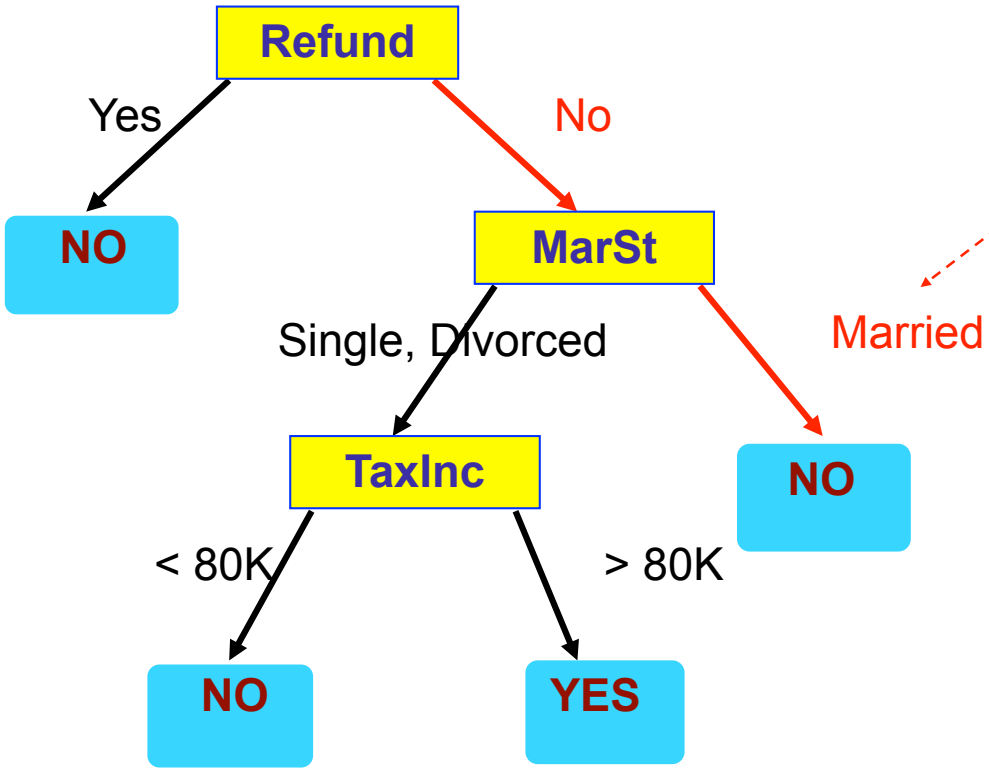
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

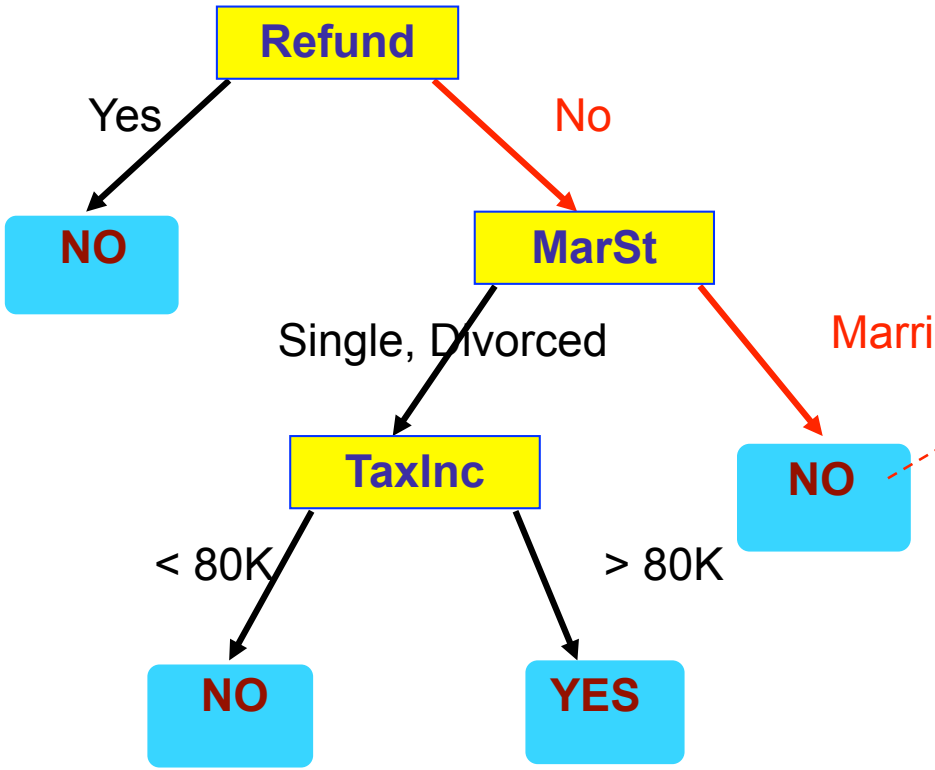
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

Why Decision Trees?

Popular!

Relatively inexpensive to build

Fast to classify new data

Easy to interpret

But first, we must
"Learn the Model"
(i.e., build the right decision tree)

Lots of approaches

Hunt's algorithm

CART

ID3, C4.5

SLIQ, SPRINT

Main ideas:

tree induction + tree pruning

General Structure of Hunt's Algorithm

[Recursively apply] Let D_t be the set of training records and y be the set of class labels $\{y_1, y_2, y_c\}$

If D_t contains records that belong the same class y_k, then its decision tree consists of a leaf node labeled as y_k

If D_t is an empty set, then its decision tree is a leaf node whose class label is determined from other information such as the majority class of the records

If D_t is contains records that belong to several classes, then a **test condition** based on one of the attributes of D_t is applied to split the data into more homogenous subsets

Example

Attributes:

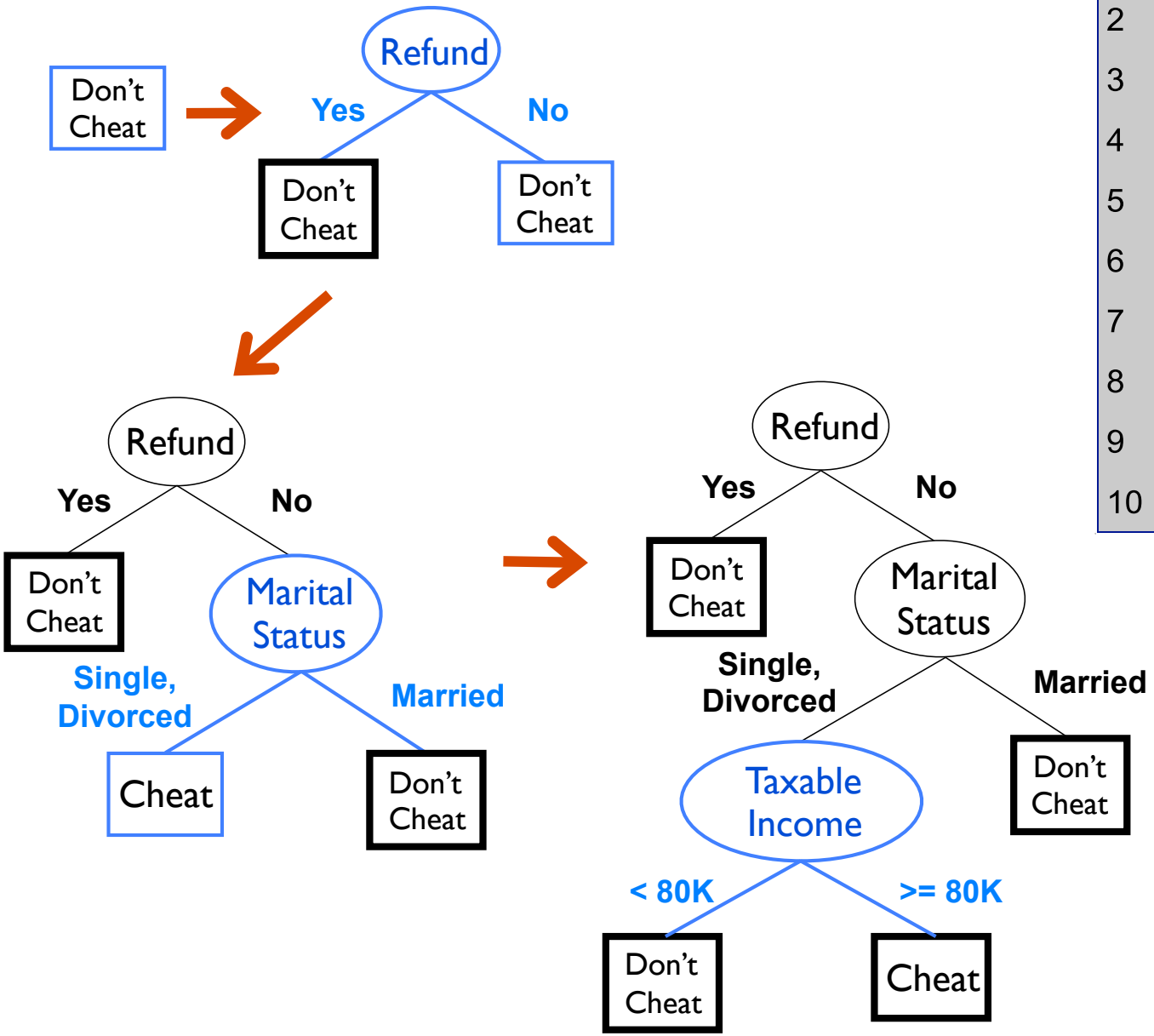
- Refund (Yes, No)
- Marital Status (Single, Divorced, Married)
- Taxable Income (quantitative)

Class:

- Cheat, Don’t Cheat

Hunt's Algorithm

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Tree Induction

- Determine how to split the records
  - Use greedy heuristics to make a series of locally optimum decision about which attribute to use for partitioning the data
  - At each step of the greedy algorithm, a test condition is applied to split the data in to subsets with a more homogenous class distribution
    - How to specify test condition for each attribute
    - How to determine the best split
- Determine when to stop splitting
  - A stopping condition is needed to terminate tree growing process. Stop expanding a node
    - if all the instances belong to the same class
    - if all the instances have similar attribute values

Splitting?

Depends on what type of attribute

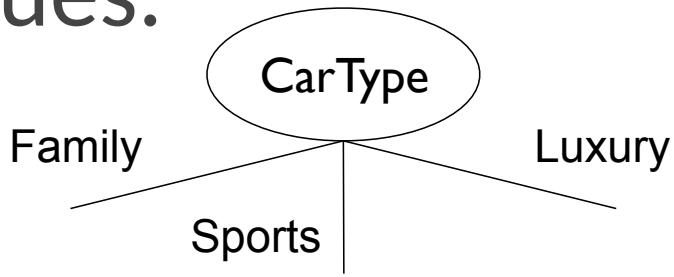
- Nominal
- Ordinal
- Quantitative

Depends on number of ways to split

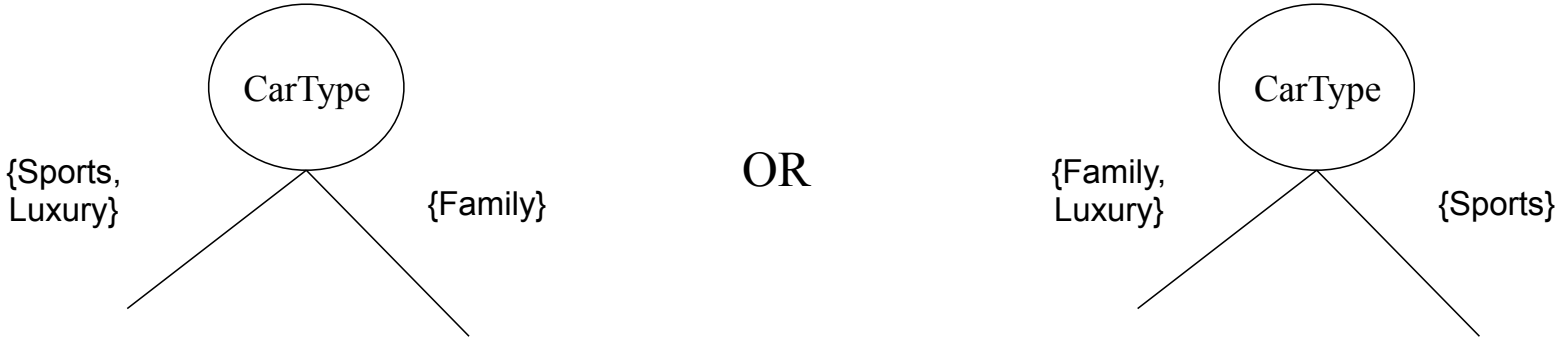
- 2-way
- Multi-way

For Nominal Attributes

Multi-way split: Use as many partitions as distinct values.

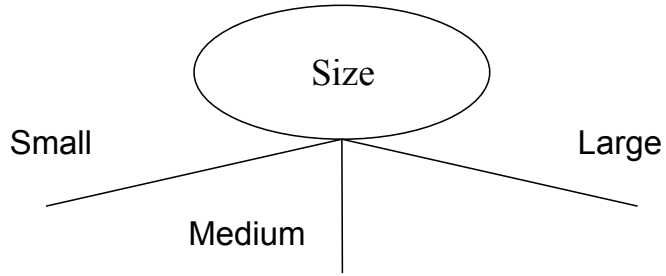


Binary split: Divides values into two subsets. Need to find optimal partitioning.

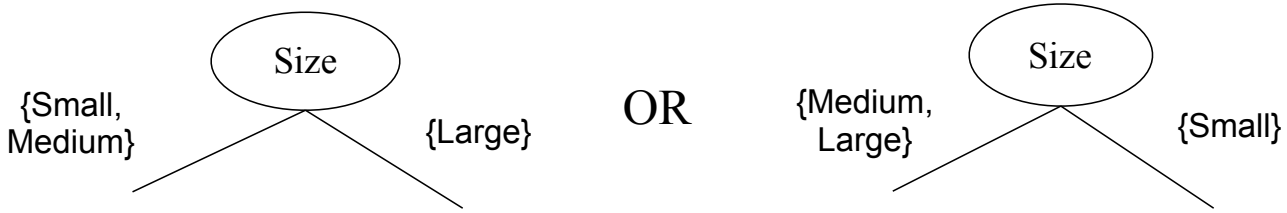


For Ordinal Attributes

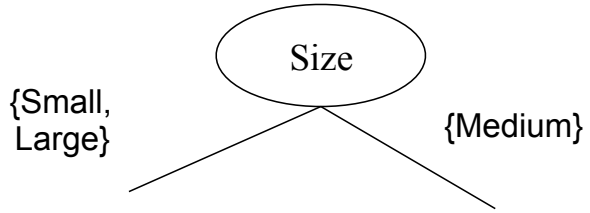
Multi-way split: Use as many partitions as distinct values.



Binary split: Divides values into two subsets.
Need to find optimal partitioning.



What about this split?



For Quantitative

Different ways of handling

Discretization to form an ordinal categorical attribute

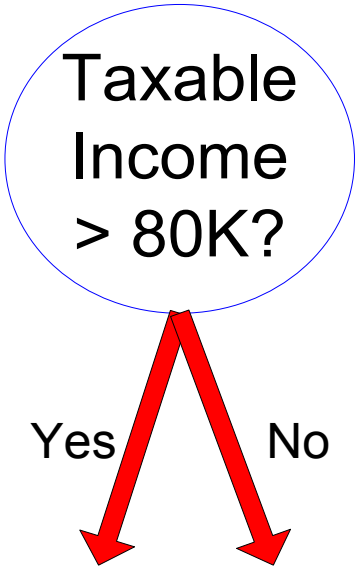
Static – discretize once at the beginning

Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

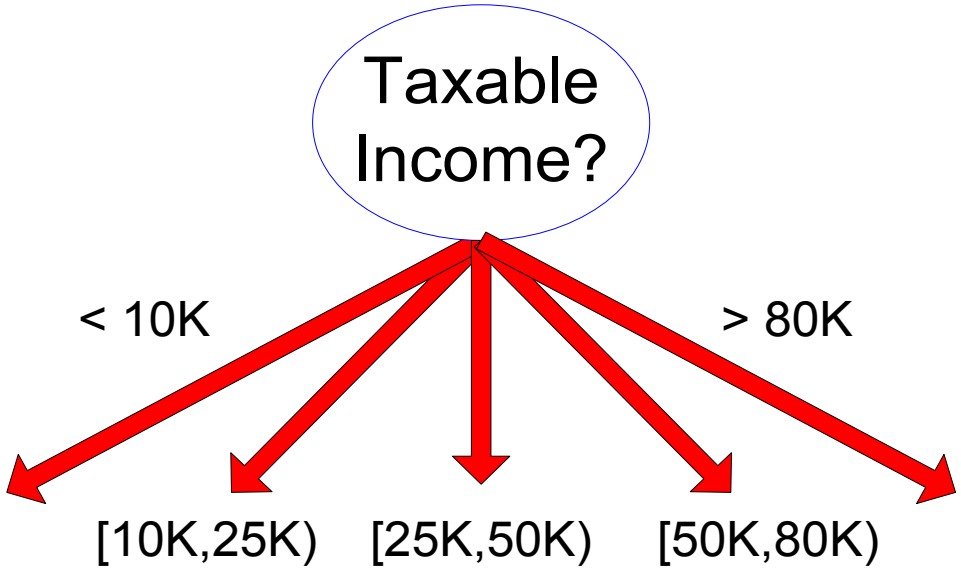
Binary Decision: $(A < v)$ or $(A \geq v)$

consider all possible splits and finds the best cut

can be more compute intensive



(i) Binary split



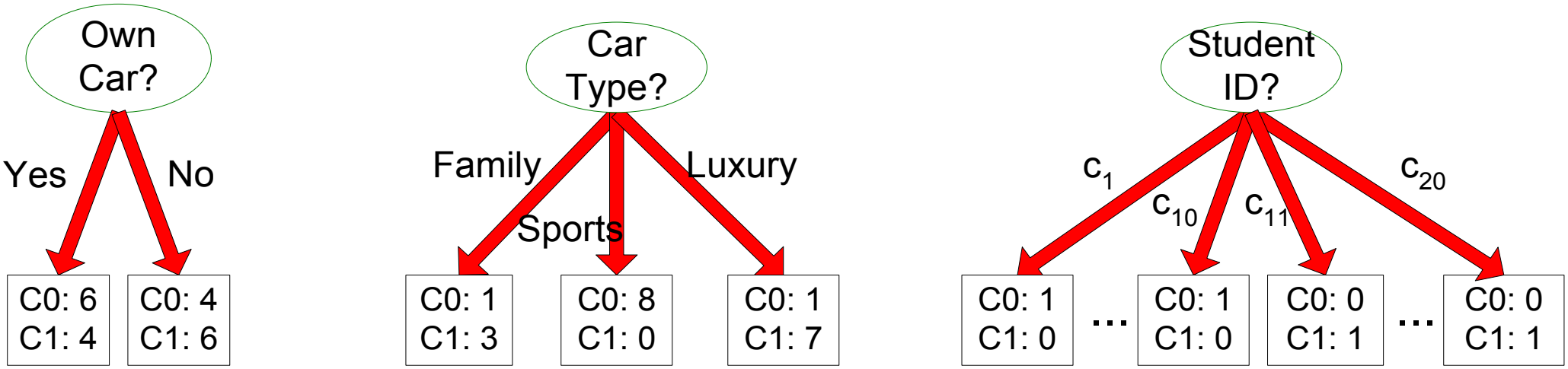
(ii) Multi-way split

But How do we Split?

Splitting Criterion

Given the data associated with a particular node in the tree, what **test condition** do we apply?

Before Splitting: 10 records of class 0, 10 records of class 1



Splitting Criterion

Ideas?

Intuition: Prefer nodes with *homogeneous* class distribution

Typical methods

- Gini Index
- Entropy / information gain
- Classification error

Splitting Criterion: GINI

Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

Measures the impurity of a node

Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information

Minimum (0.0) when all records belong to one class, implying most interesting information

GINI Example

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

Splitting Based on GINI

Used in CART, SLIQ, SPRINT.

When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

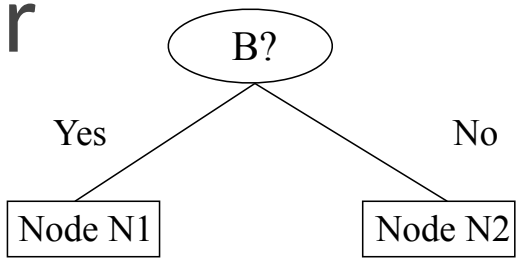
where, ni = number of records at child i, n = number of records at node p.

GINI for Binary Attributes

Splits into two partitions

Effect of weighing partitions

Larger and purer partitions are sought for



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

	N1	N2
C1	**0**	**6**
C2	**6**	**0**
Gini=0.000		

	N1	N2
C1	**5**	**1**
C2	**1**	**5**
Gini=0.278		

	N1	N2
C1	**4**	**2**
C2	**3**	**3**
Gini=0.486		

	N1	N2
C1	**3**	**3**
C2	**3**	**3**
Gini=0.500		

GINI for Nominal Attributes

For each distinct value, gather counts for each class in the dataset

Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

GINI for Quantitative Attributes

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values
    = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Gini for Quantitative Attributes

For efficient computation: for each attribute,

- Sort the attribute on values
- Linearly scan these values, each time updating the count matrix and computing gini index
- Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230

	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Other splitting criteria

Information Gain

Classification Error

(see readings)

When do we stop splitting?

Stop expanding a node when all the records belong to the same class

Stop expanding a node when all the records have similar attribute values

Early termination

Building Decision Trees with MapReduce