PENNSTATE

Classification Analysis

Presentation extended from the slides of the textbook, Introduction to Data Mining by Tan et al. and supplementary material

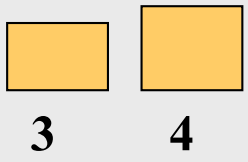
Overview

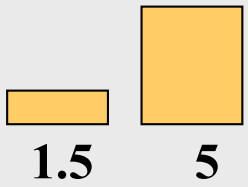
- Basics
- Logistic Regression
- Decision Trees
- Model Overfitting
- Model Evaluation
- Rule-Based Classifier
- Instance-Based Classifier
- Bayesian Classifiers
- Artificial Neural Networks
- Support Vector Machines

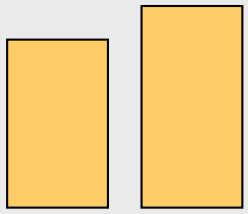
Materials on Logistic Regression prepared based on machine learning lectures by Andrew Ng.

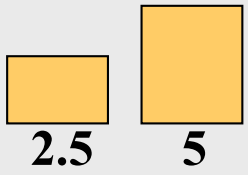
Toy Problem 1

Examples of class A

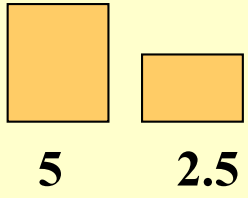


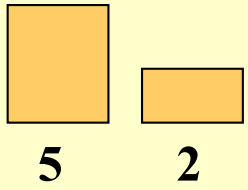


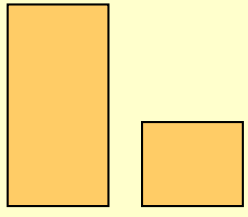


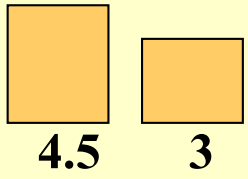


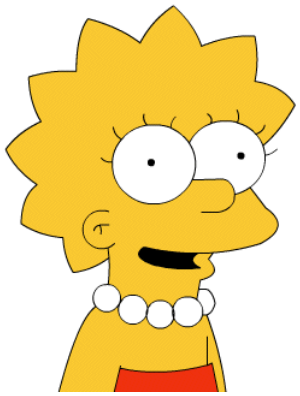
Examples of class B



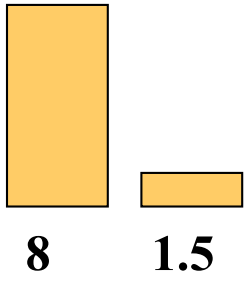




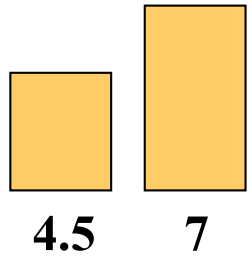




What class is this object?

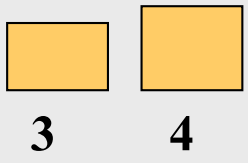


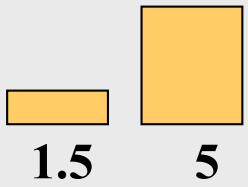
What about this one, A or B?

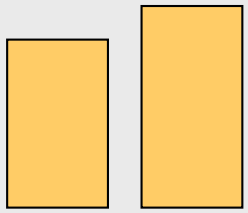


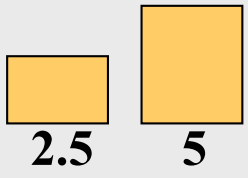
Toy Problem 1

Examples of class A

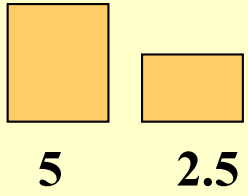


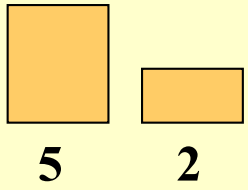


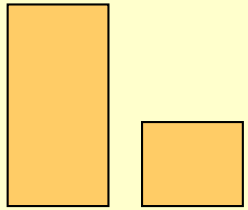


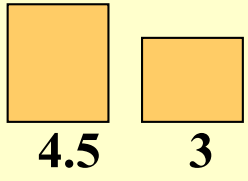


Examples of class B

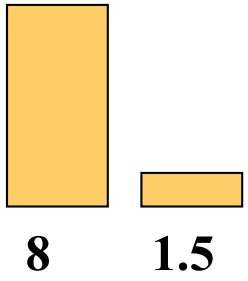








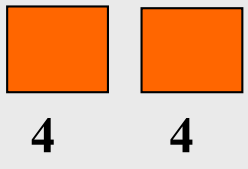
This is a B!

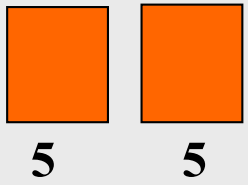


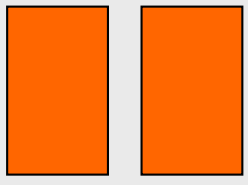
Here is the rule. If the left bar is smaller than the right bar, it is an A, otherwise it is a B.

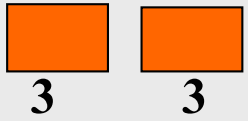
Toy Problem 2

Examples of class A

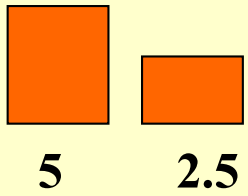


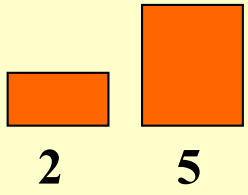


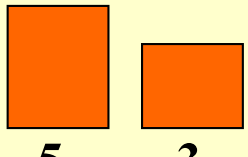


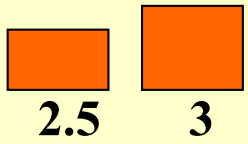


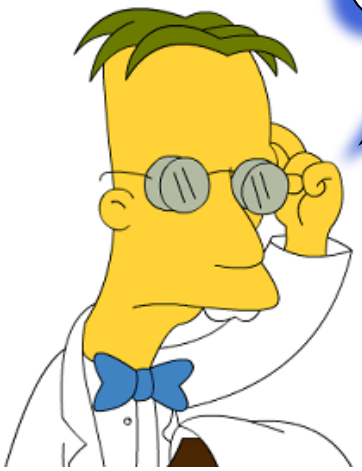
Examples of class B

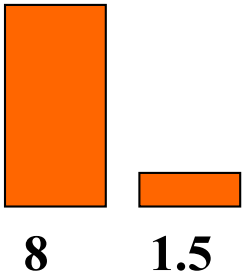


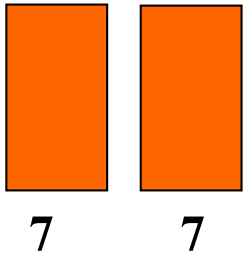






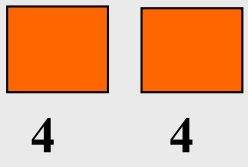


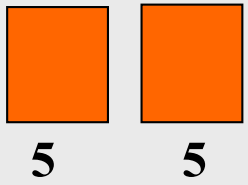


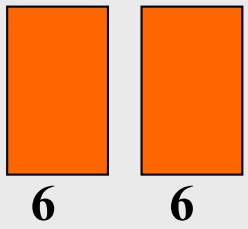


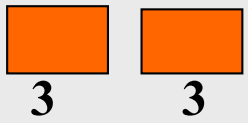
Toy Problem 2

Examples of class A

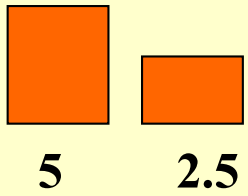


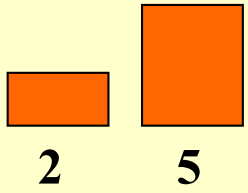


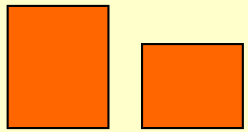


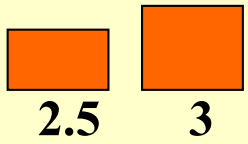


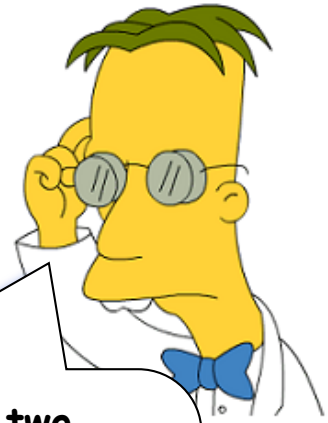
Examples of class B





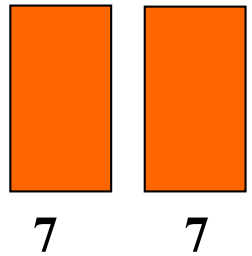






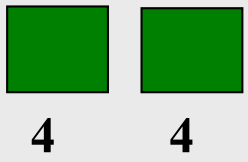
The rule is as follows, if the two bars are equal sizes, it is an A. Otherwise it is a B.

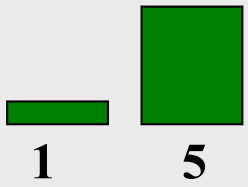
So this one is an A.

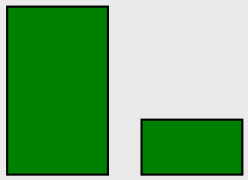


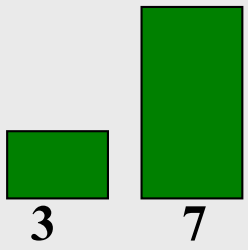
Toy Problem 3

Examples of class A

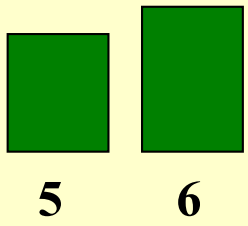


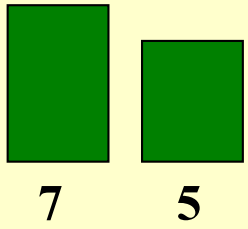


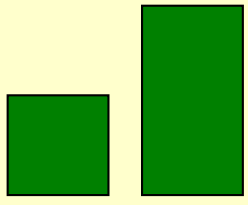


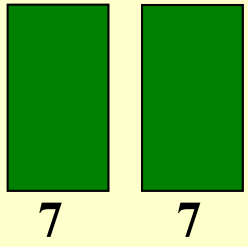


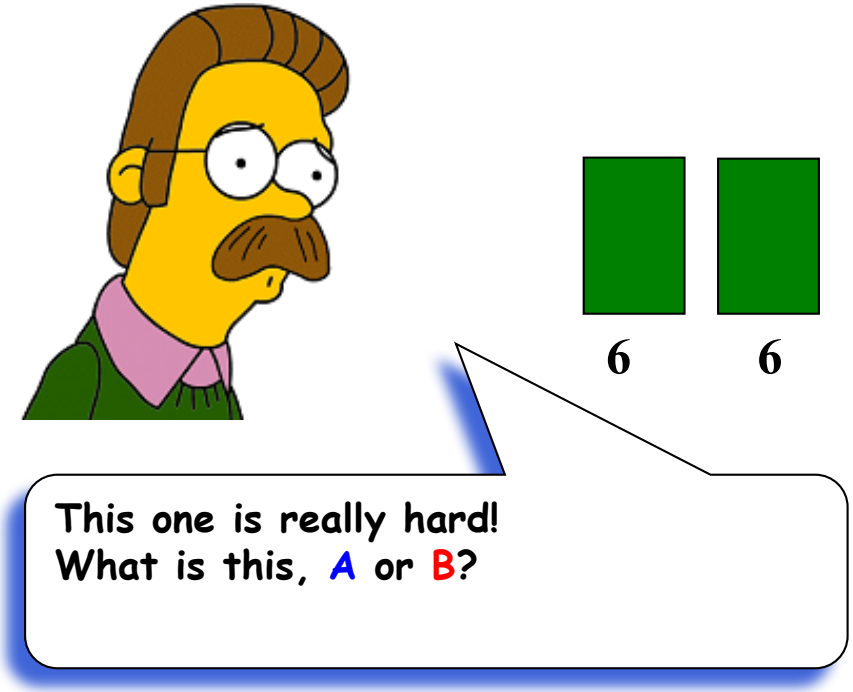
Examples of class B





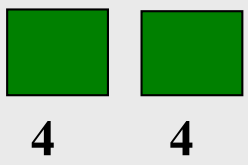


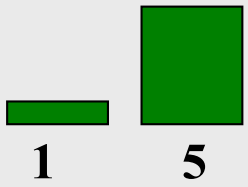


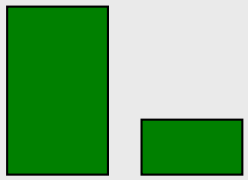


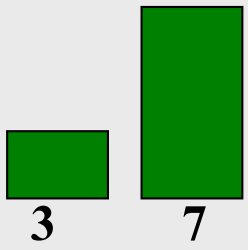
Toy Problem 3

Examples of class A

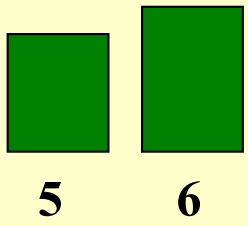


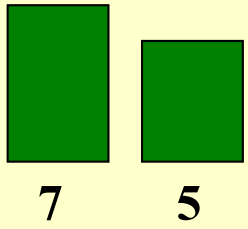


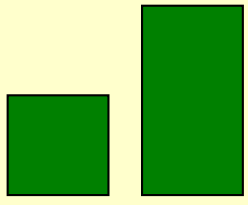


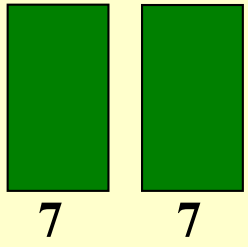


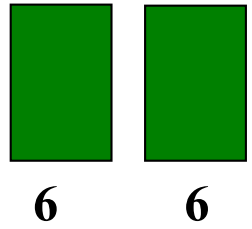
Examples of class B









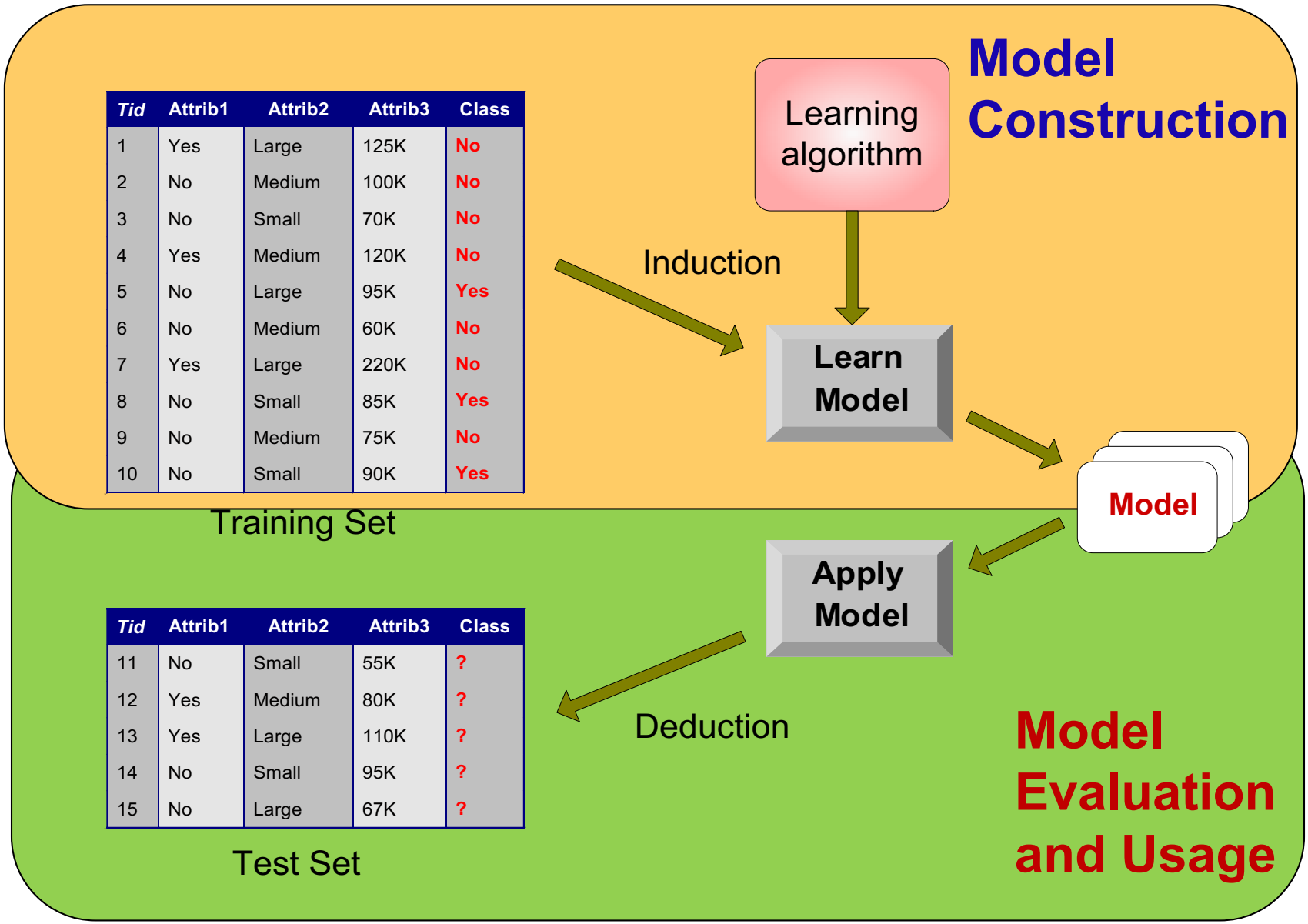


The rule is as follows, if the the sum of the two bars is less than or equal to 10, it is an A. Otherwise it is a B.

Classification: Definition

- **Given**: a collection of records/objects (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- **Find**: a *model* for class attribute as a function of the values of other attributes.
- **Goal**: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model.
  - Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.
- Most effective for datasets with binary or nominal categories, less effective on ordinal categories.

General Classification Process



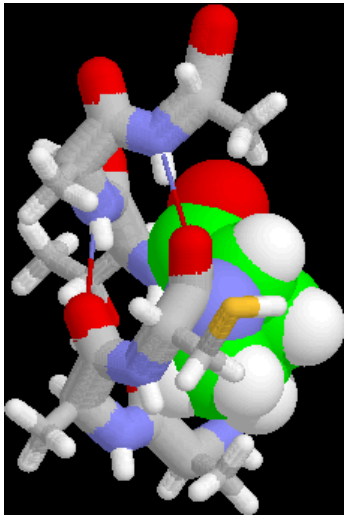
Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

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	?

Examples of Classification Task

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent

- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil



- Categorizing news stories as finance, weather, entertainment, sports, etc

Classification Techniques

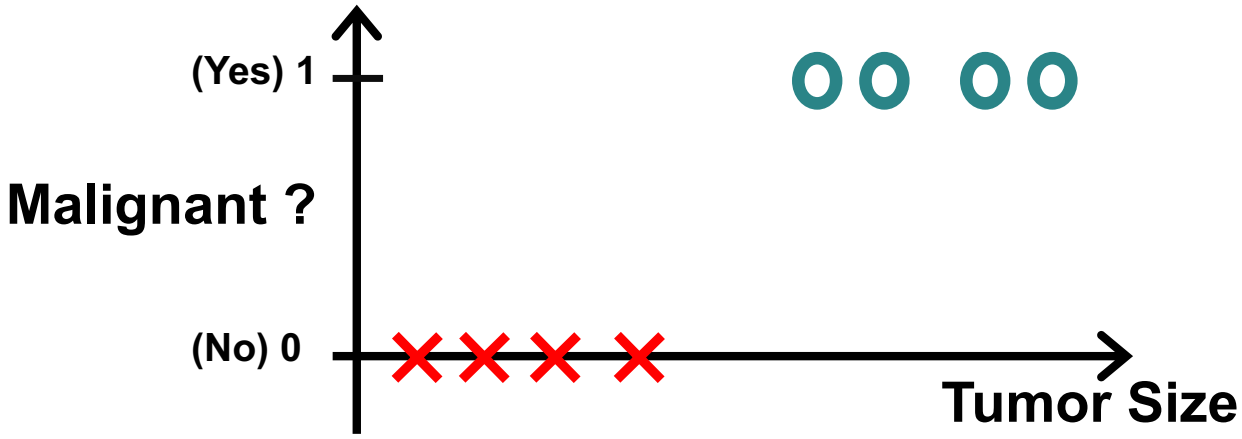
- Logistic Regression
- Decision Tree based Methods
- Rule-based Methods
- Instance-based Methods
  - Memory based reasoning
- Bayesian Classifiers
- Artificial Neural Networks
- Support Vector Machines

Binary Classification

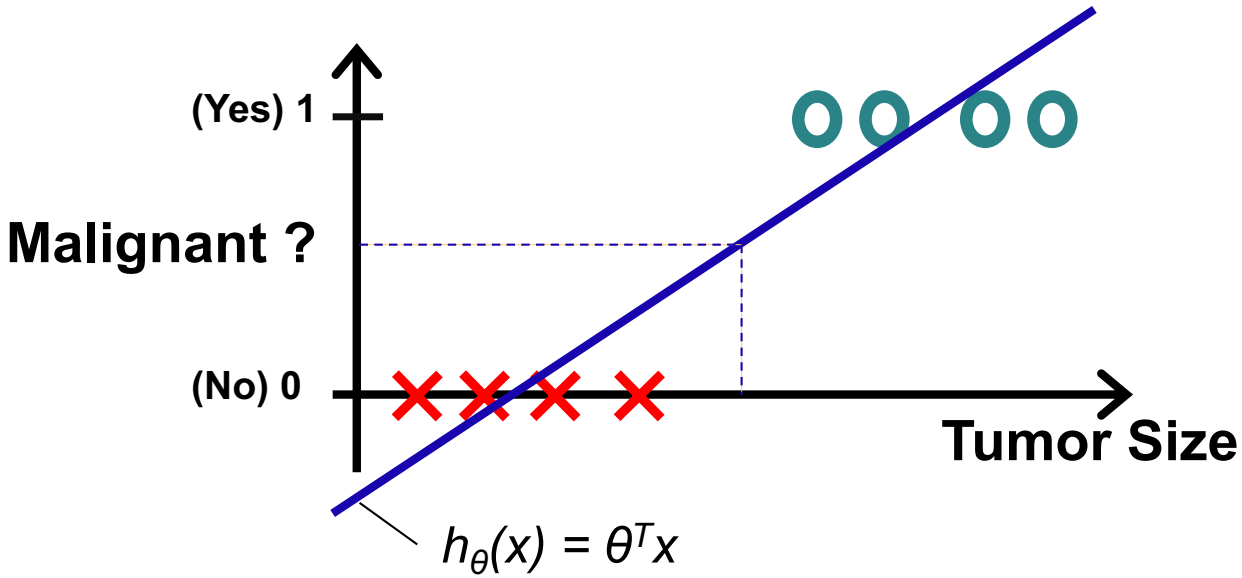
- Consider a binary classification problem of predicting whether a tumor is malignant based on the size of the tumor:

$y \in \{0, 1\}$

0: “Negative Class” (e.g., benign tumor)
1: “Positive Class” (e.g., malignant tumor)

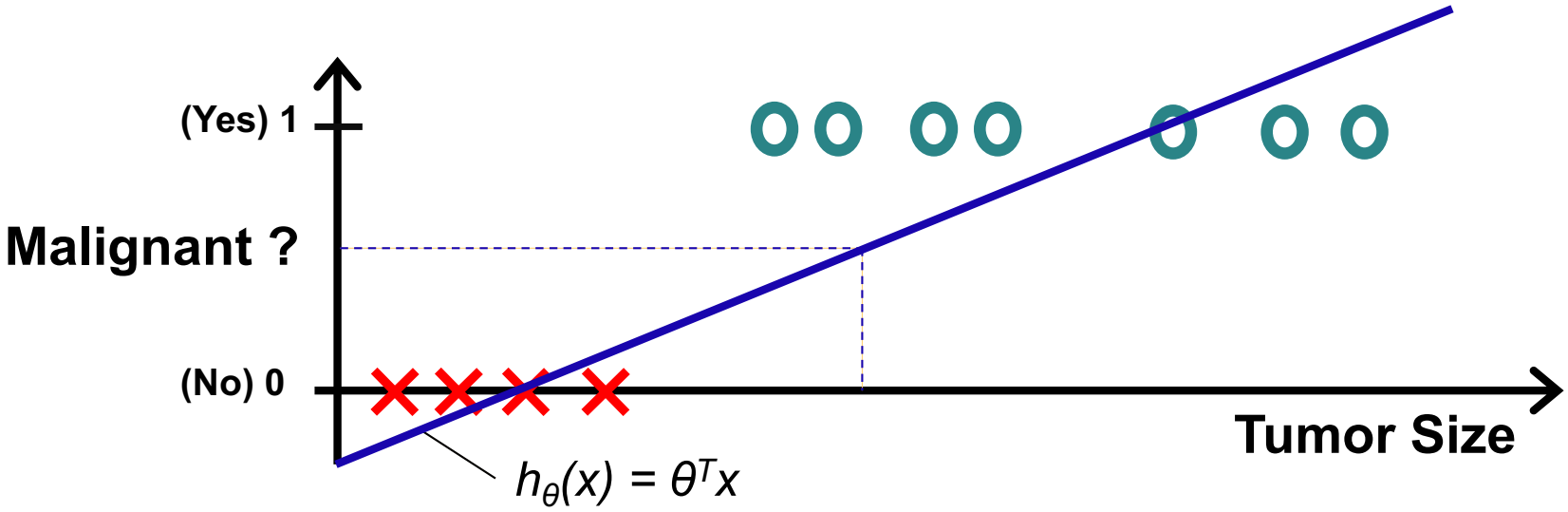


Linear Regression as a Solution



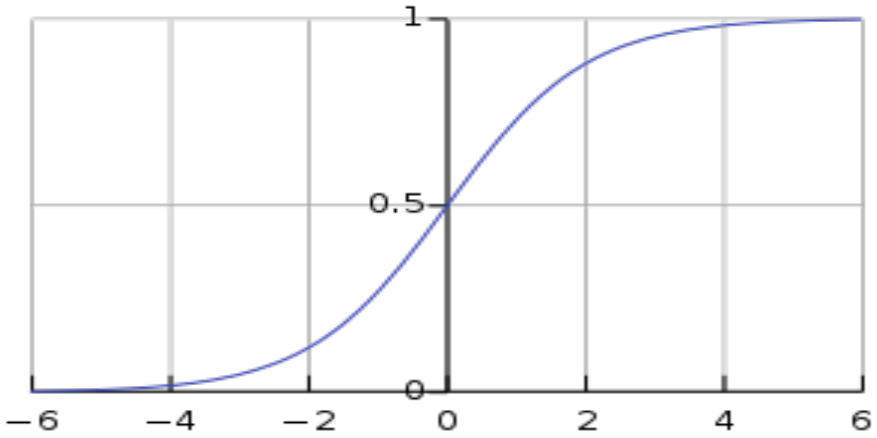
- May make classification by set a threshold on the predicted value $h_\theta(x)$ at 0.5
  - *$y = 1$ if $h_\theta(x) >= 0.5$*
  - *$y = 0$ if $h_\theta(x) < 0.5$*

Issues with Linear Regression



- Linear regression model may perform rather poorly
  - The shape of linear functions (i.e., line) does not fit very well with the class attributes (i.e., 0/1)
  - It does not make sense for $h_\theta(x)$ to generate values larger than 1 or smaller than 0
- We need a different function with a shape better fitting the training data while falling with [0,1]

Sigmoid Function



$$S(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

- Sigmoid with the desired properties serves well for binary classification!
  - Bounded within [0, 1] and fitting well with samples
- Logistic regression model applies sigmoid function to transform a parameterized function (e.g., linear function) into the desirable S-shape curve.

Logistic Regression Model

- Sigmoid function takes a real number *z* as the input:

$$S(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

- Models weight on the dependent variables ***x*** *(x_1, …, x_n)* with corresponding parameters *θ* *(θ_0, θ_1,…, θ_n)*
  - Linear regression has a hypothesis function $h_\theta(x) = \theta^T x$
- Logistic Regression applies Sigmoid on $\theta^T x$ to serve as its hypothesis function.

$$h_\theta(x) = S(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

Interpretation of Hypothesis Output

- The hypothesis function of Logistic Regression model returns a value within [0,1]. This output can be interpreted as:
- The probability that y = 1, given x, parameterized by *θ.*

$$h_\theta(x) = P(y = 1|x; \theta)$$

  - For example, a tumor of 1cm is found in patient p. $h_\theta(1cm) = 0.8$ tells the patient that there is a 80% chance for the tumor to be malignant.
- The probability that y = 0, given x, parameterized by *θ:*

$$P(y = 0|x;\ \theta) = 1 - P(y = 1|x; \theta)$$

Logistic Regression Learning

- Optimization problem: Choose θ (θ_0, θ_1, … θ_n) so that h_θ is close to y in our training examples ($\boldsymbol{x}$, y)

Hypothesis:

$$h_\theta(x) = S(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

Parameters:

$$\theta = \theta_0, \theta_1, \ldots, \theta_n$$

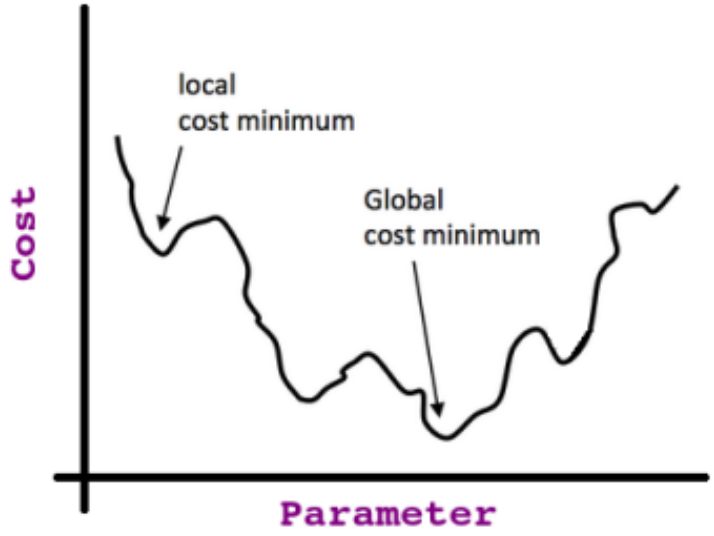
Objective (Cost) Function: (based on idea of SSE)

$$J(\theta) = J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal: $\underset{\theta}{\text{minimize}}\ J(\theta)$

Objective (Cost) Function

- In linear regression optimization, the objective is to minimize the average prediction errors based on the idea of *SSE*.
- With the adopted hypothesis function, however, $J(\theta)$ is no longer a convex function.
  - There is no guarantee for the gradient descent algorithm to reach the global minimum



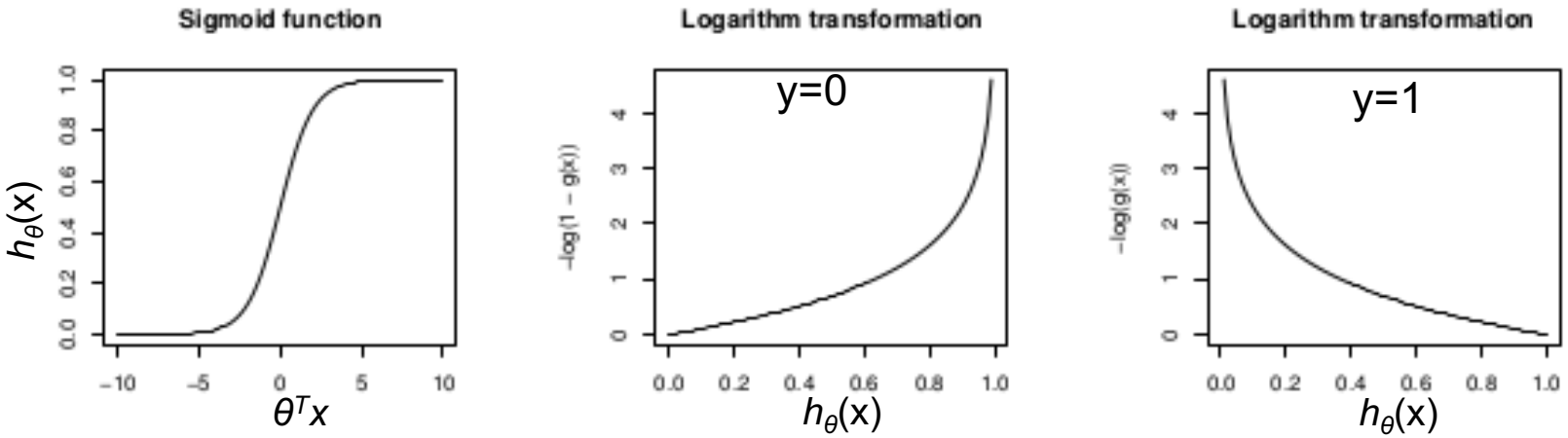
- To address this issue, an alternative error measure is desirable.

Reformulation of Prediction Error

- Error defined using the logarithm of sigmoid function

$$Err(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$Err(h_\theta(x), y) = (y - 1)\log(1 - (h_\theta(x)) - y\log(h_\theta(x))$$



- Intuition: For *y=0*, the Error is small when the prediction $h_\theta(x)$ is close to 0 (more accurate) and grow as $h_\theta(x)$ increases to 1 (inaccurate). For y=1, vice versa.

Logistic Regression Learning

- Optimization problem: Choose θ (θ_0, θ_1, … θ_n) so that h_θ is close to y in our training examples ($\boldsymbol{x}$, y)

Hypothesis:

$$h_\theta(x) = S(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

Parameters:

$$\theta = \theta_0, \theta_1, \ldots, \theta_n$$

Objective (Cost) Function: (based on log of Sigmoid)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Err(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Goal: $\underset{\theta}{\text{minimize}}\ J(\theta)$

Gradient Descent for Logistic Regression

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

- Derivatives are the same as linear regression!
- Repeat

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

Until *J(θ)* converges

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

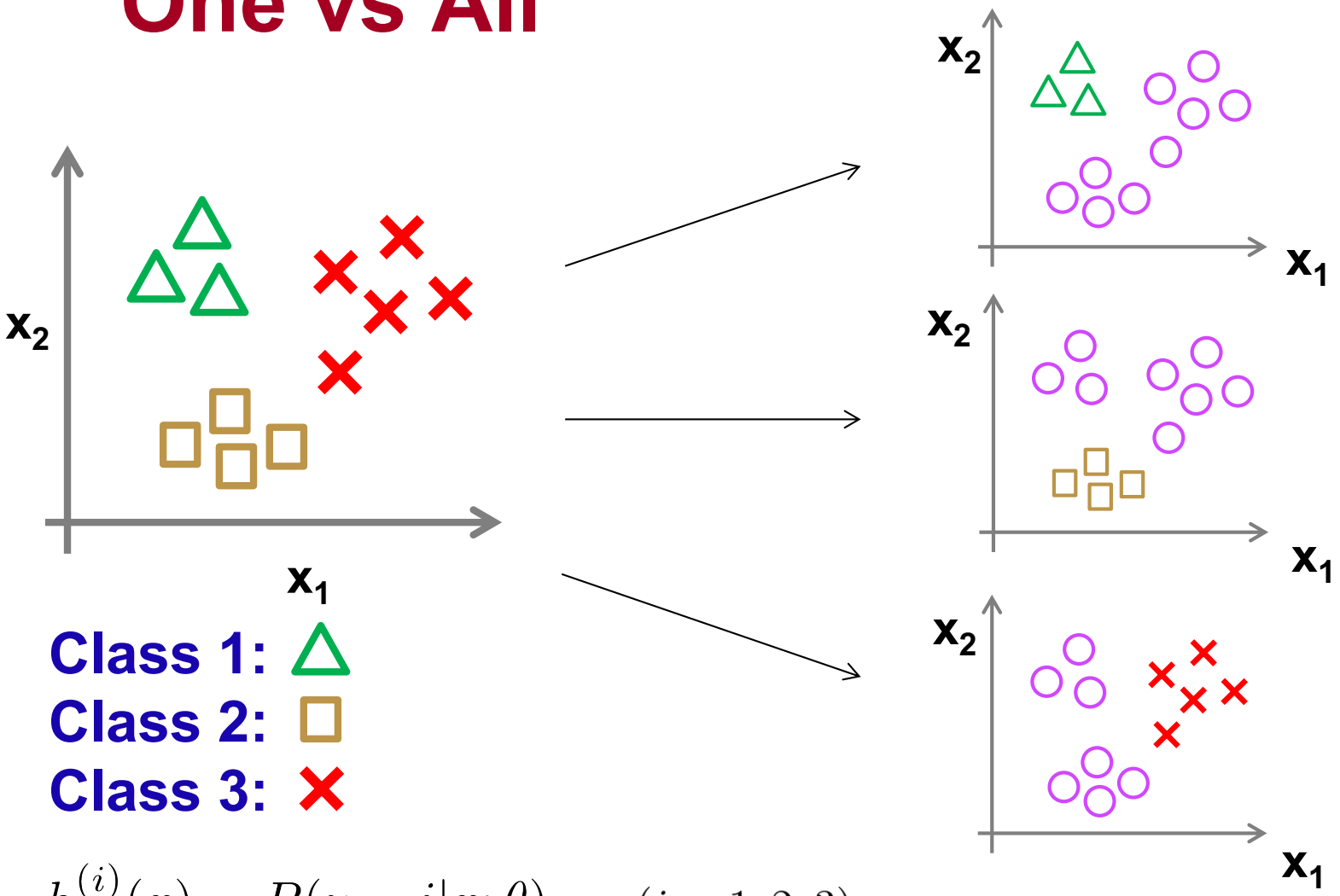
Multi-class Classification Problem

- There are many real-world applications where the data has more than two classes.
  - Weather: Sunny, Cloudy, Rain, Snow
  - Email tagging: Work, Friends, Family, Spam
  - Movies: Action, Adventure, Comedy, Drama, Horror
- Multi-class Classification:
  - Given: N different classes where each object in the training data belongs to only one class.
  - Goal: Construct a model (function) which correctly predict the class of a new data object.
- Multi-Label Classification
  - Each object can belong to multiple classes.

General Strategies for Multi-Class Classification

- Transformation to binary
  - One-vs-All/Rest: training a binary classifier per class, with that class vs all other classes. The base classifiers need to produce a confidence score for making decision.
  - One-vs-One: For each pair of classes, train a binary classifier. The decision is made by voting.
- Extension from binary:
  - Extend the ideas of existing binary classifiers to solve multi-class classification problems.
- Hierarchical classification:
  - Organize the classes into a hierarchy. Each parent node is divided into multiple child nodes. Follow the hierarchy to make decision.

One vs All



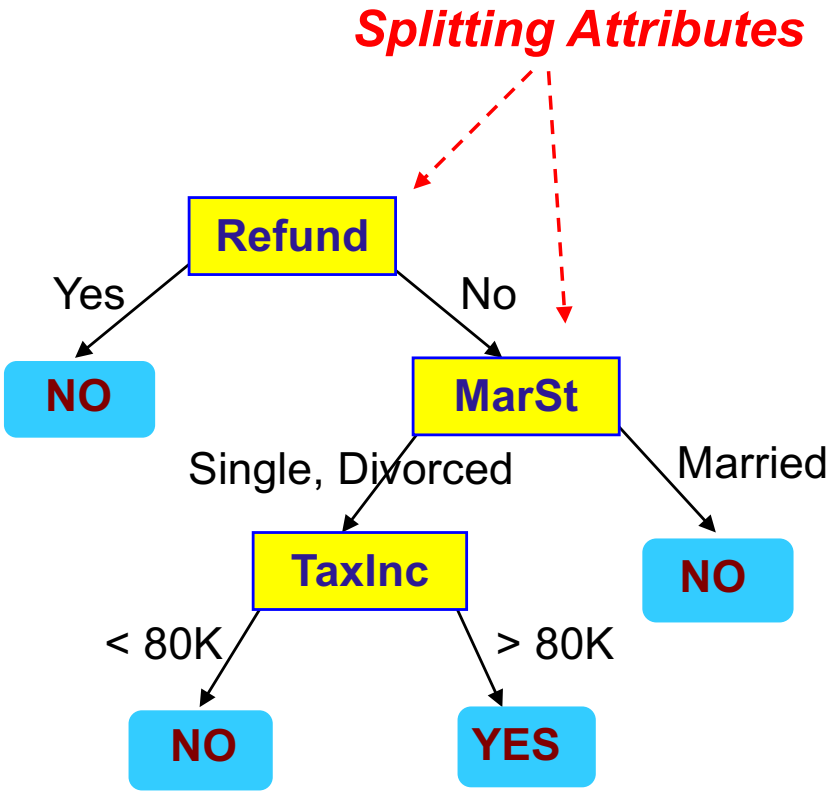
$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$

- For *x*, pick the class *i* that maximizes $\max_i h_\theta^{(i)}(x)$

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

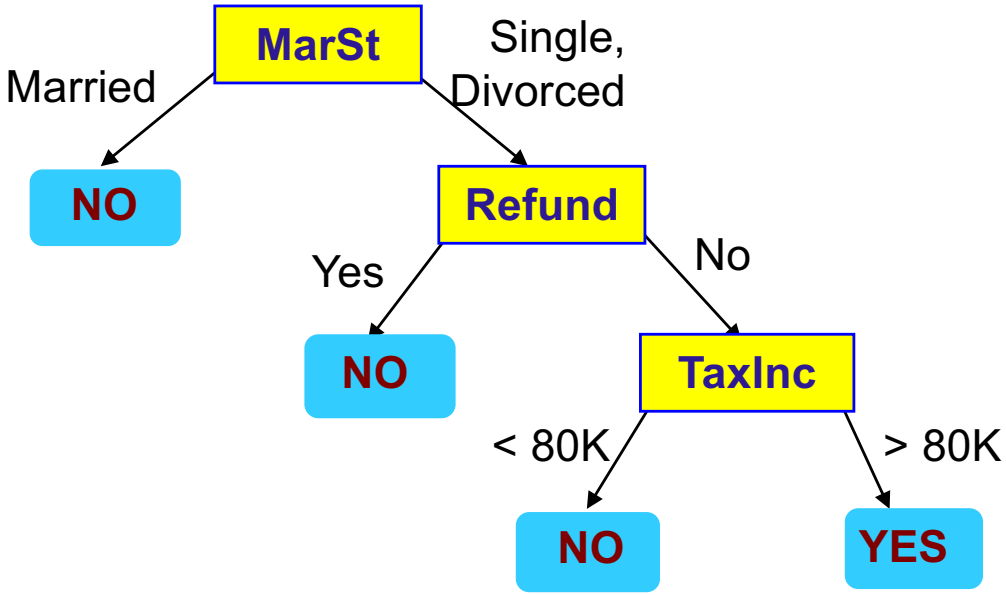
Training Data



Model: Decision Tree

Another Example of Decision Tree

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



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

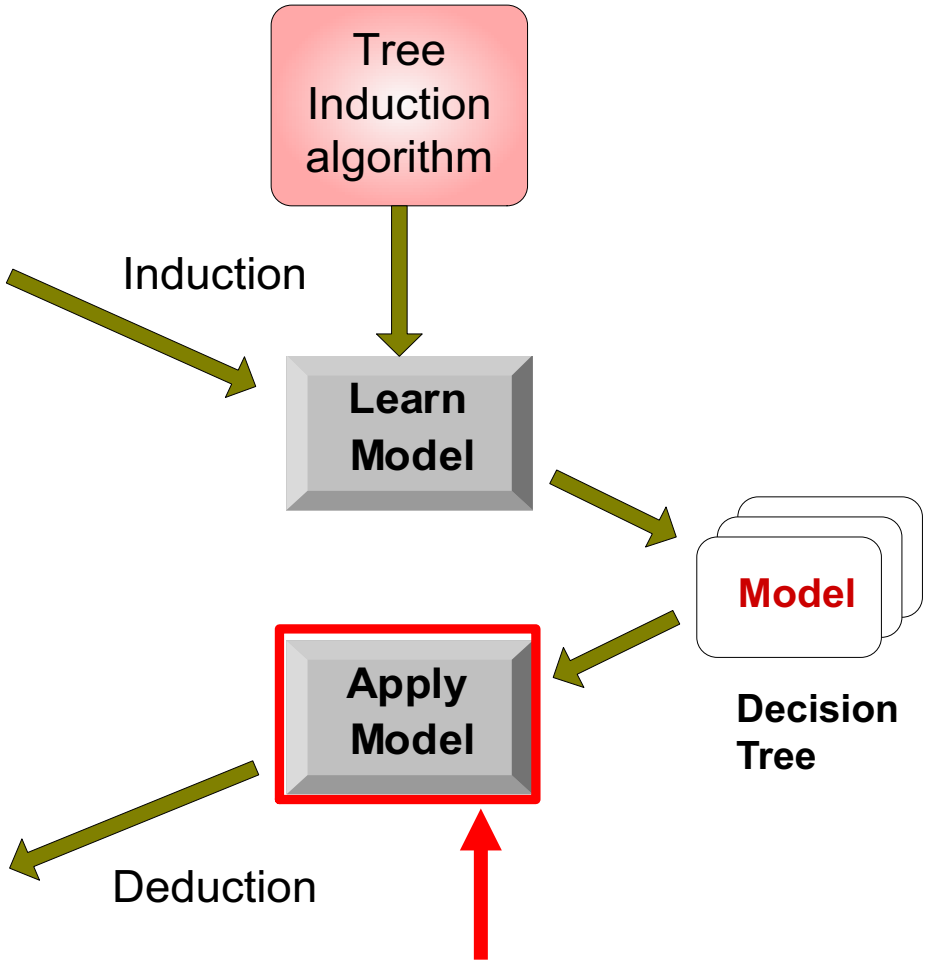
Decision Tree Classification Task

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

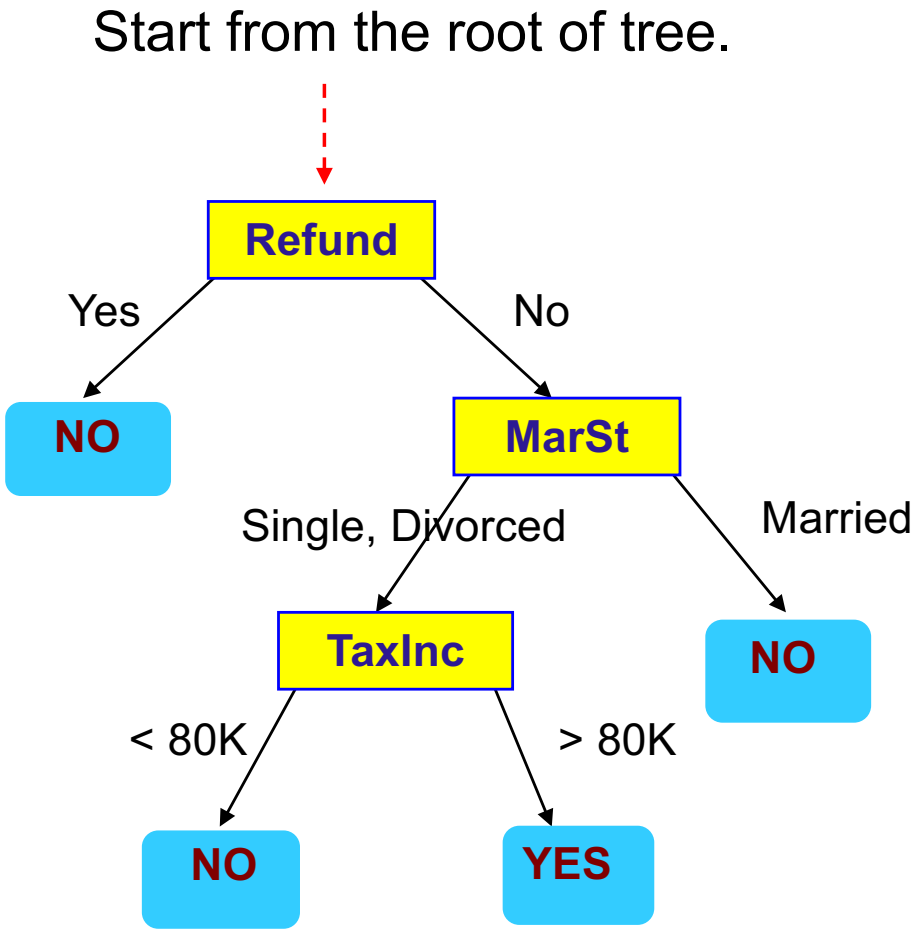
Training Set

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



Apply Model to Test Data



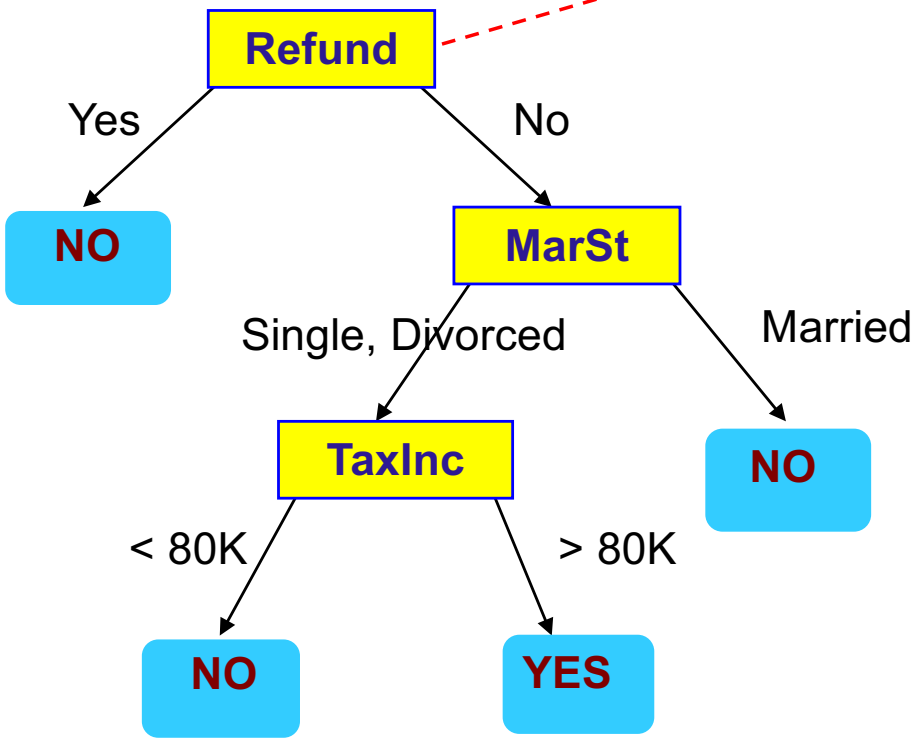
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

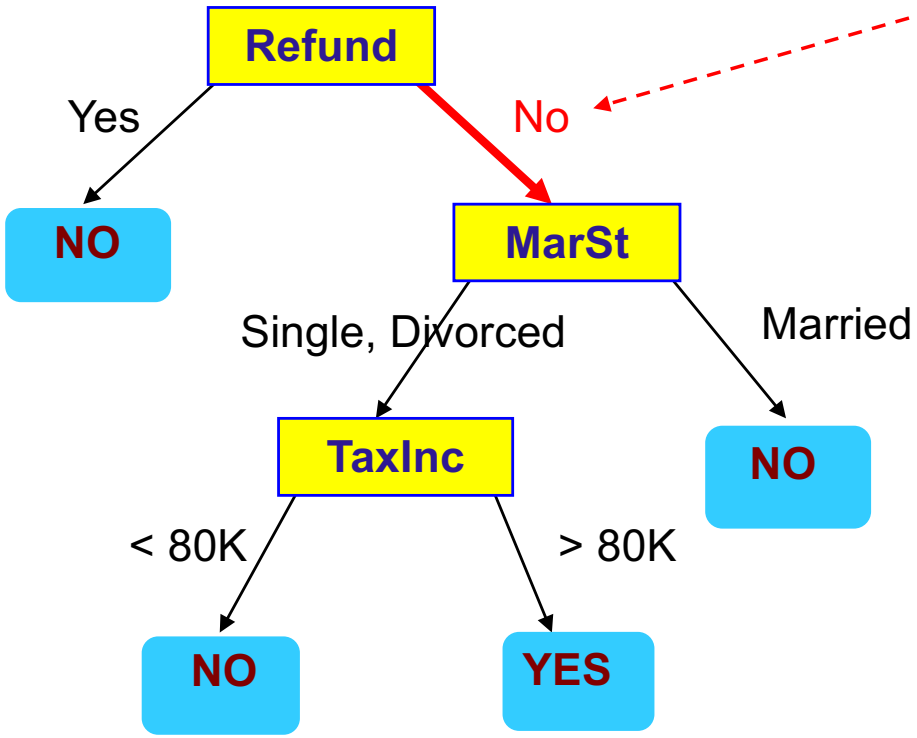
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

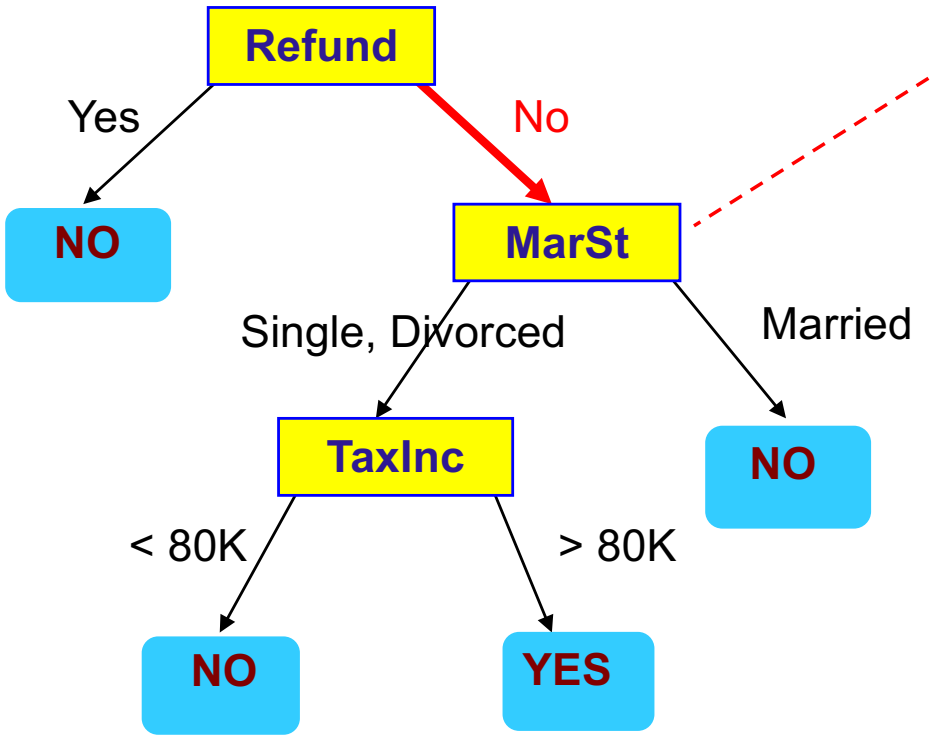
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

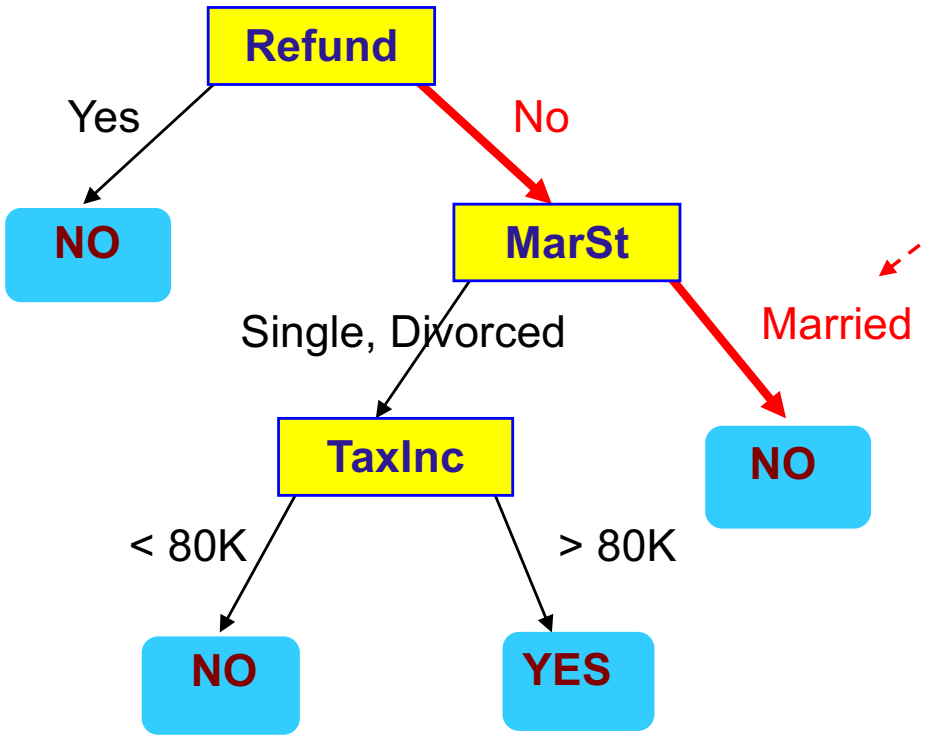
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

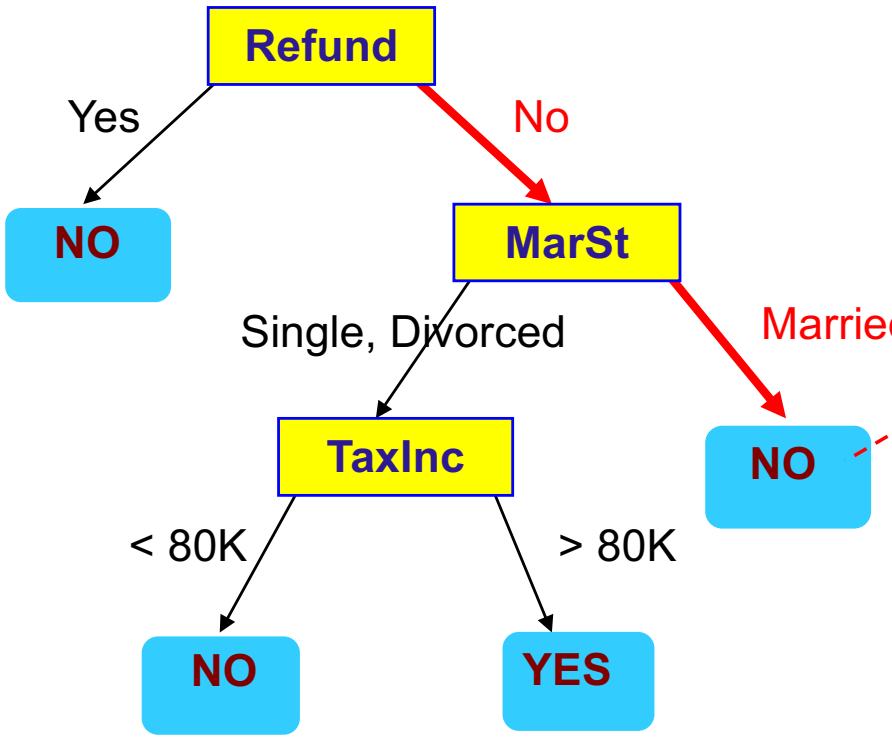
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

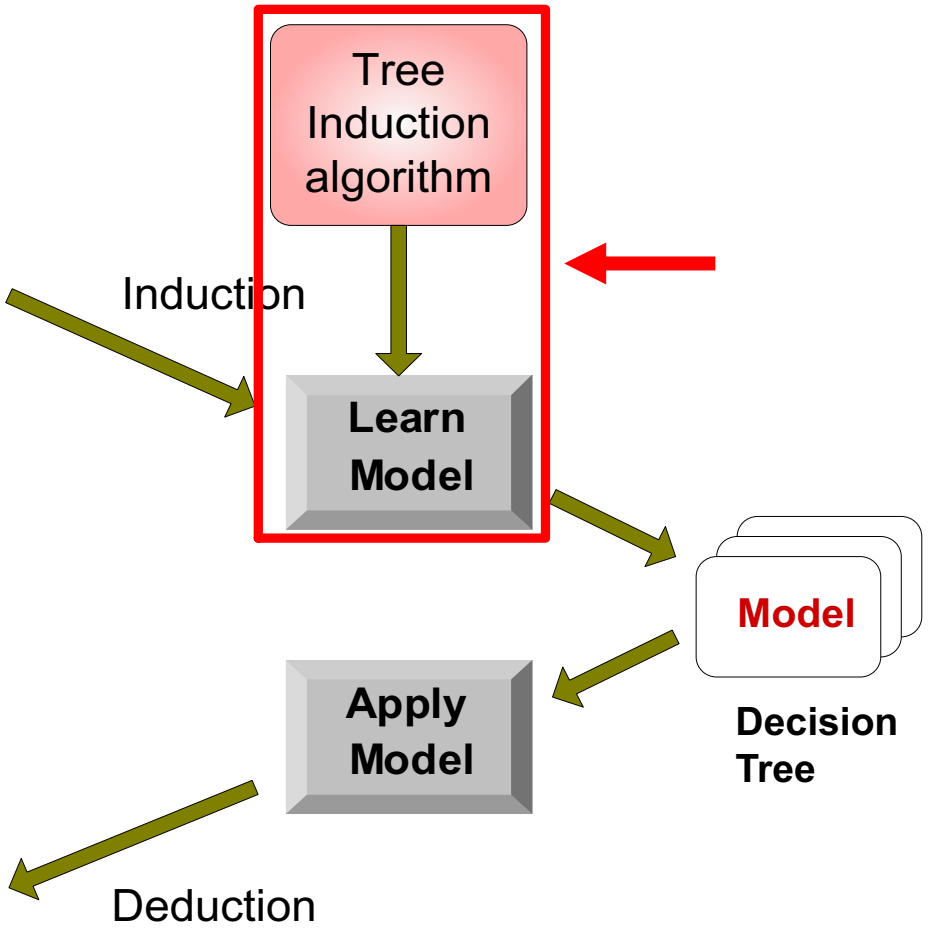
Decision Tree Induction Task

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



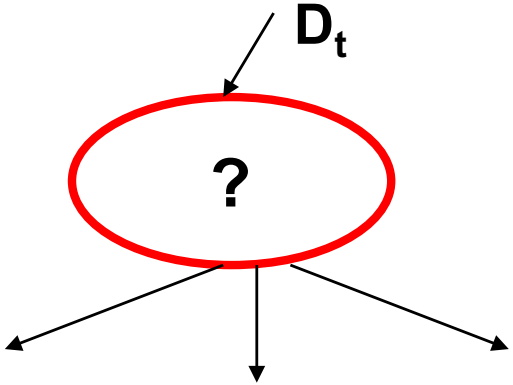
Decision Tree Induction

- Idea of Decision Tree:
  - Post a series of questions about attributes of a test record in order to figure out (i.e., classify) its class.
- Key Issue:
  - How to build a decision tree (with corresponding questions) based on attributes of the training records
  - Exponentially many possible decision trees
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

Hunt's Algorithm

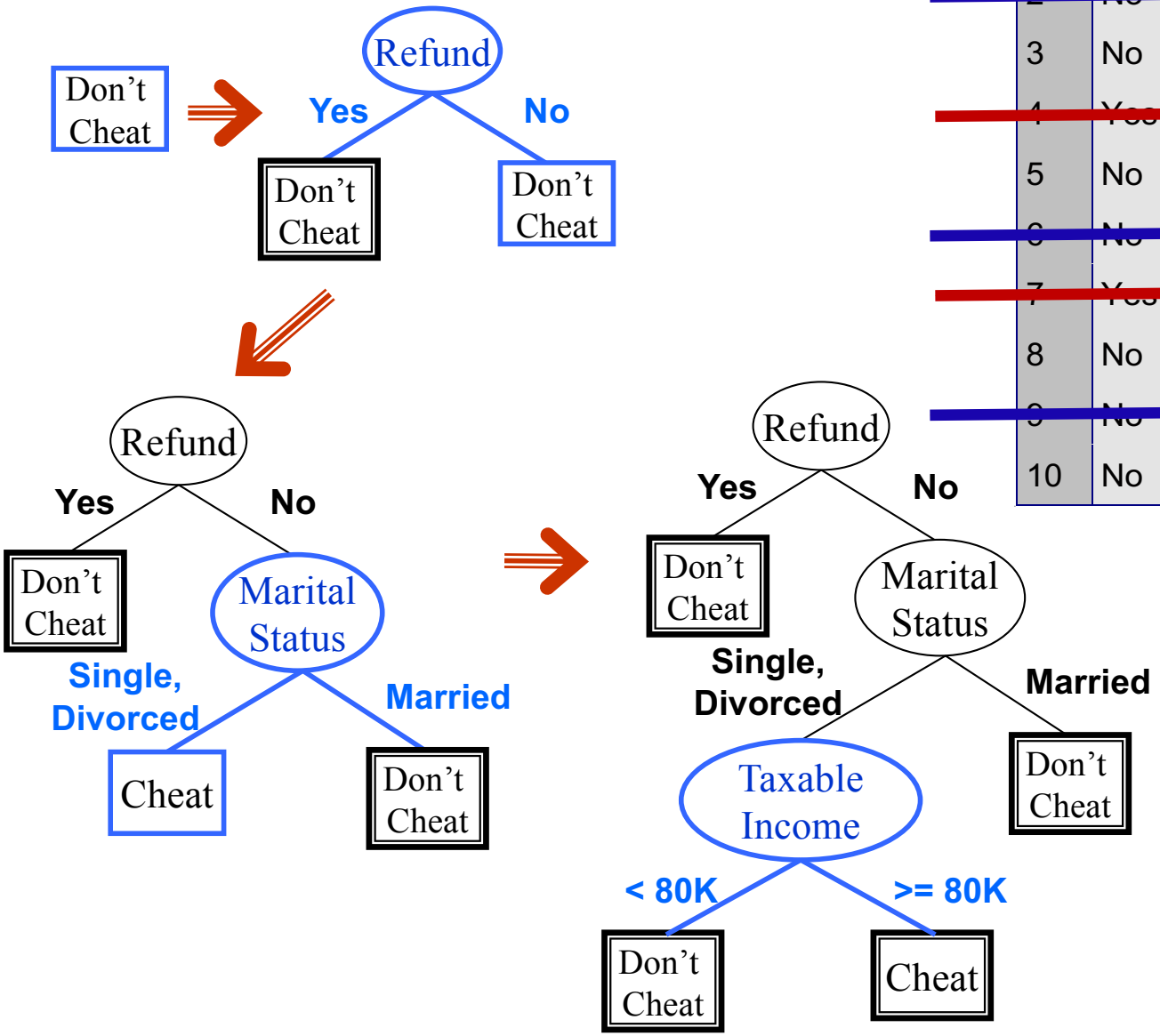
- Let D_t be the set of training records that reach a node t. *Recursively* apply the following:
  - If D_t contains records that all belong to class y_t, then t is a leaf node labeled as y_t
  - If D_t is empty, t is a leaf node labeled by default class, y_d, (majority class of parent node)
  - If D_t cannot be split, then t is labeled by the majority class.
  - If D_t belongs to more than one class, use an *attribute test condition* to split the data into smaller subsets.

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



Hunt's Algorithm (Illustration)

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

- There are too many ways to build decision trees
  - Exponential partitioning of the attribute space
- Greedy strategy.
  - Split the records based on an *attribute test condition* that optimizes certain criterion.
- Issues
  - Determine *how to split the records*
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine *when to stop splitting*

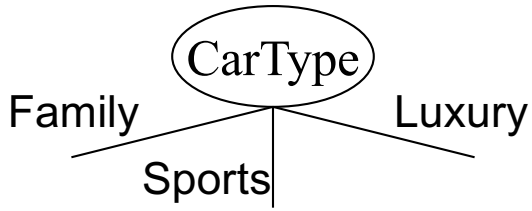
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

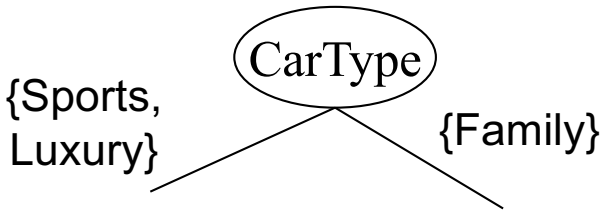
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

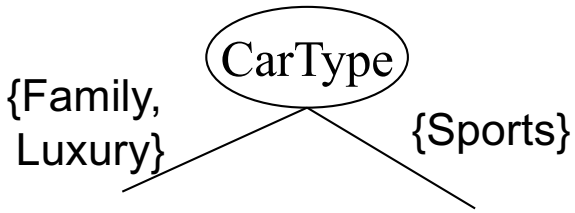
- Multi-way split: Use as many partitions as distinct values.



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

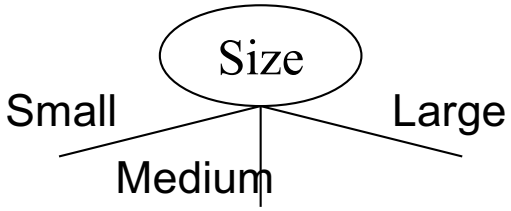


OR

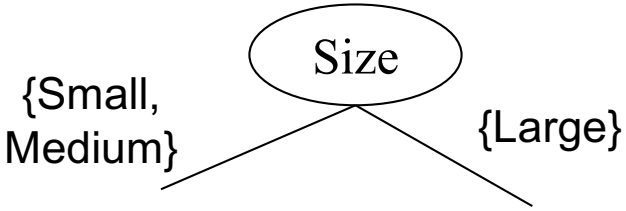


Splitting Based on Ordinal Attributes

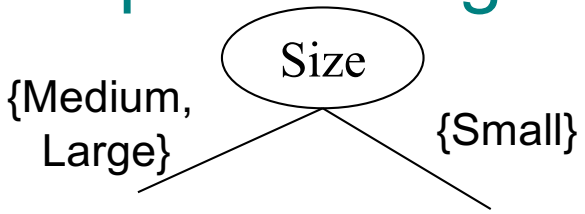
- Multi-way split: Use as many partitions as distinct values.



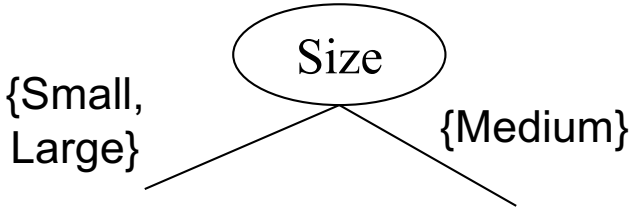
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



OR



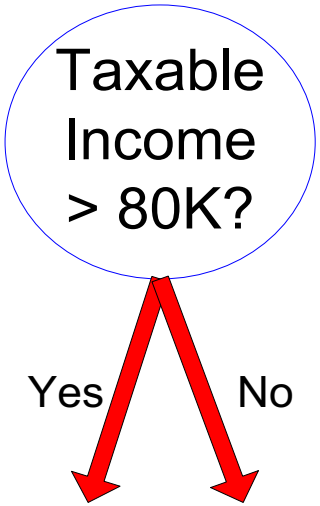
- What about this split?



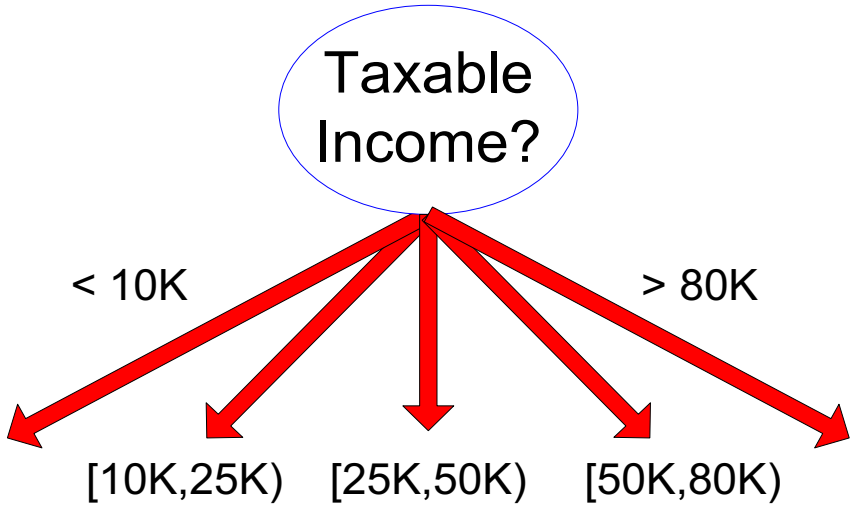
Splitting Based on Continuous Attributes

- 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 at *v*
    - Can be more computationally intensive

Splitting Based on Continuous Attributes



(i) Binary split

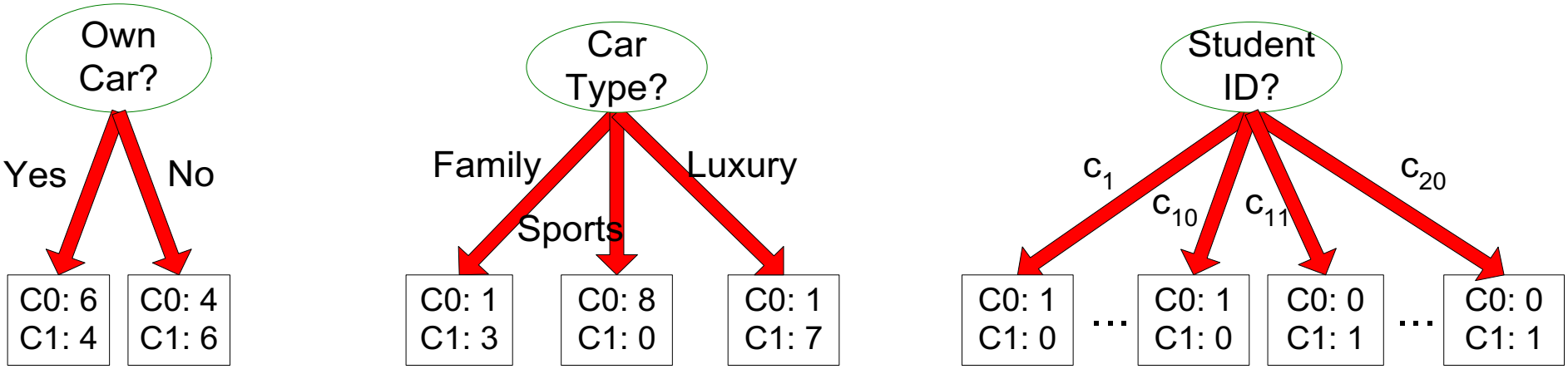


(ii) Multi-way split

While we can generate so many different ways of split, the problem is *how to find the best split*.

How to determine the Best Split?

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



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of *node impurity*:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	**Homogeneous, Low degree of impurity**

Measures of Node Impurity

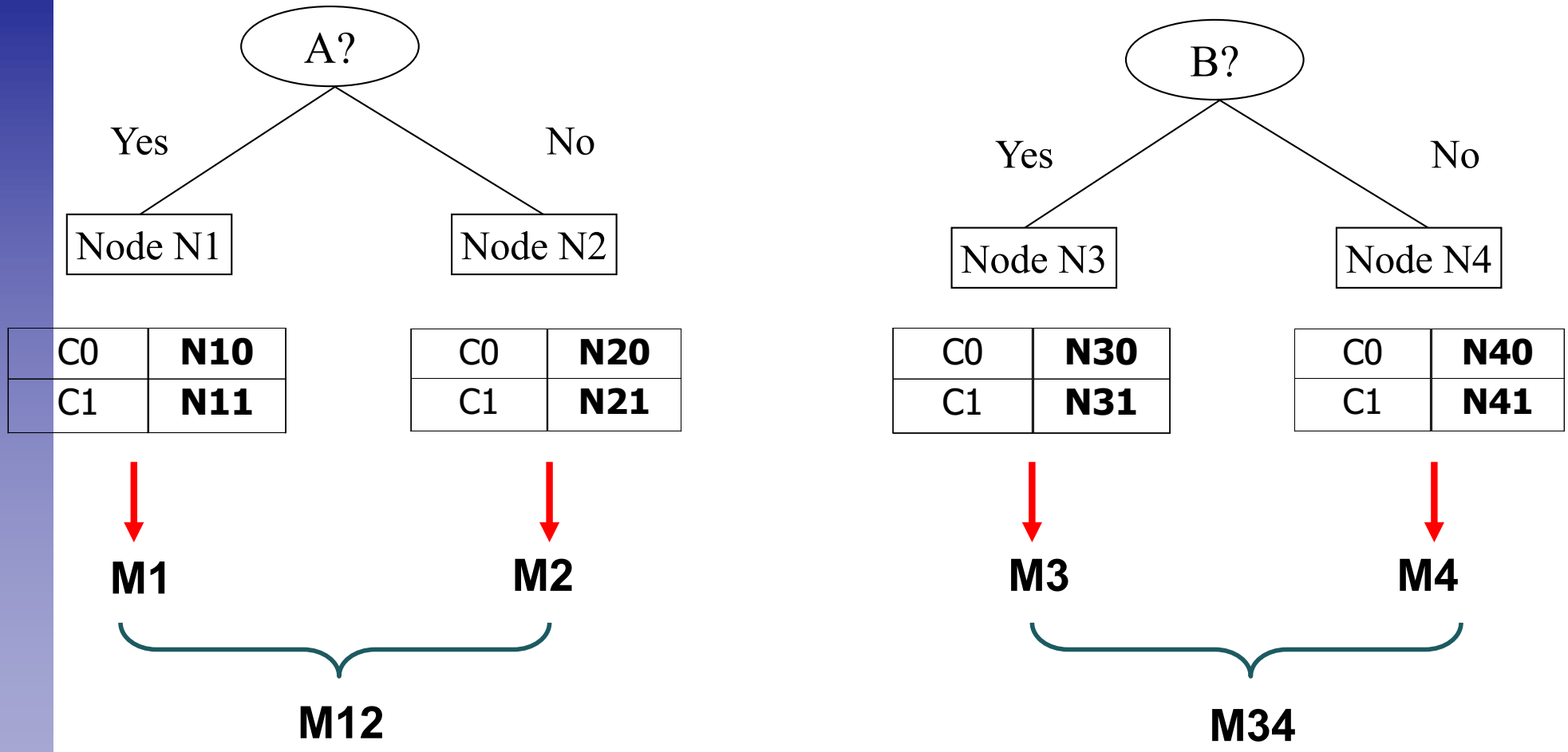
- Gini Index
- Entropy
- Misclassification error

How to Find the Best Split

Before Splitting:

C0	**N00**
C1	**N01**

→ **M0 (Impurity measure)**



Gain = M0 – M12 vs M0 – M34

Measure of Impurity: GINI

- Gini Index for a given node *t* :

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

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

- Maximum is $1 - 1/n_c$ where n_c is no. of classes, when *records are equally distributed* among all classes, implying least interesting information
- Minimum is 0 when *all records belong to one class*, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Computing GINI for Node t: Examples

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

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 *t* is split into *k* partitions (children), the *quality of split* is computed as

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

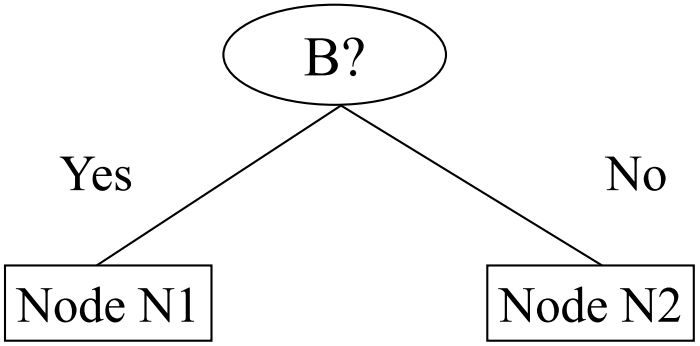
where, n_i = number of records at child i,

n = number of records at node t.

- This is the weighted sum of GINI for individual nodes.

GINI Index: 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	

Gini(N1)
= 1 – (5/7)² – (2/7)²
= 0.408

Gini(N2)
= 1 – (1/5)² – (4/5)²
= 0.32

	N1	N2
C1	5	1
C2	2	4
Gini=0.371		

Gini(Children)
= 7/12 * 0.408 +
5/12 * 0.32
= 0.371

Gini Index: Categorical 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)

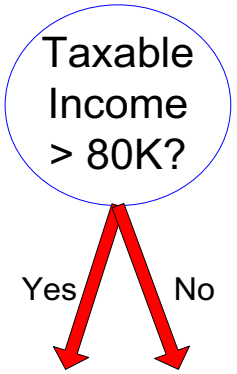
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

Gini Index: Continuous Attributes

- Making Binary Decisions to split based on one value
  - Many possible splitting values
  - # of possible splitting values ~= # of distinct values
- Each splitting value v has a count matrix with class counts in each partition, A < v; A ≥ v
- Simple method to choose best v
  - For each v, scan database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Scanning DB is repetitive.

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



Gini Index: Continuous 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

Alternative Splitting Criteria based on Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

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

- Measures *homogeneity* of a node.
  - Maximum = log n_c when records are equally distributed among all classes implying least information
  - Minimum = 0 when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Computing Entropy for Node t: Examples

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	0
C2	6

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

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	1
C2	5

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

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	2
C2	4

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

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on Information Gain (Entropy)

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

p split into *k* partitions; n_i :no. of records in partition *i*

  - Maximizes GAIN: Reduction in Entropy (impurity) is achieved due to the split. Choose split with most reduction
  - Used in ID3 and C4.5
  - Disadvantage: Tends to prefer splits that result in *large number of partitions, each being small but pure*.

Splitting Criteria based on Classification Error

- *Classification error* at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures *misclassification error made by a node*.
  - Maximum = 1 - $1/n_c$ when records are equally distributed among all classes, implying least interesting information
  - Minimum = 0 when all records belong to one class, implying most interesting information

Classification Error: Examples

$$Error(t) = 1 - \max_i P(i \mid t)$$

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

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

Error = 1 – max (0, 1) = 1 – 1 = 0

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

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

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

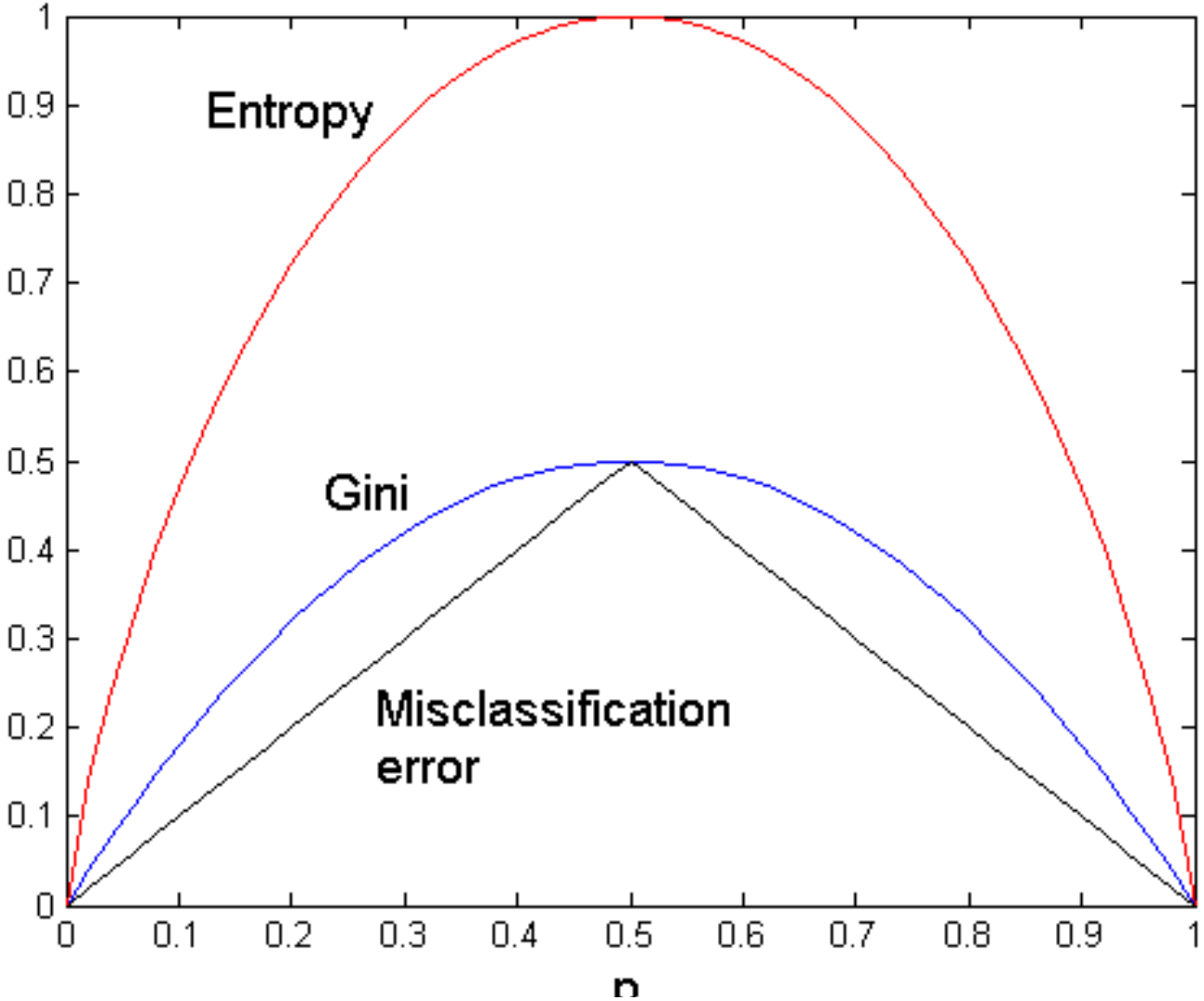
C1	**2**
C2	**4**

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

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Stopping Criteria for Tree Induction

- 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 (i.e., difficult to split)*
- Early termination based on some *threshold*, e.g., on number of records in a node.

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - *Easy to interpret* for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

Example: C4.5

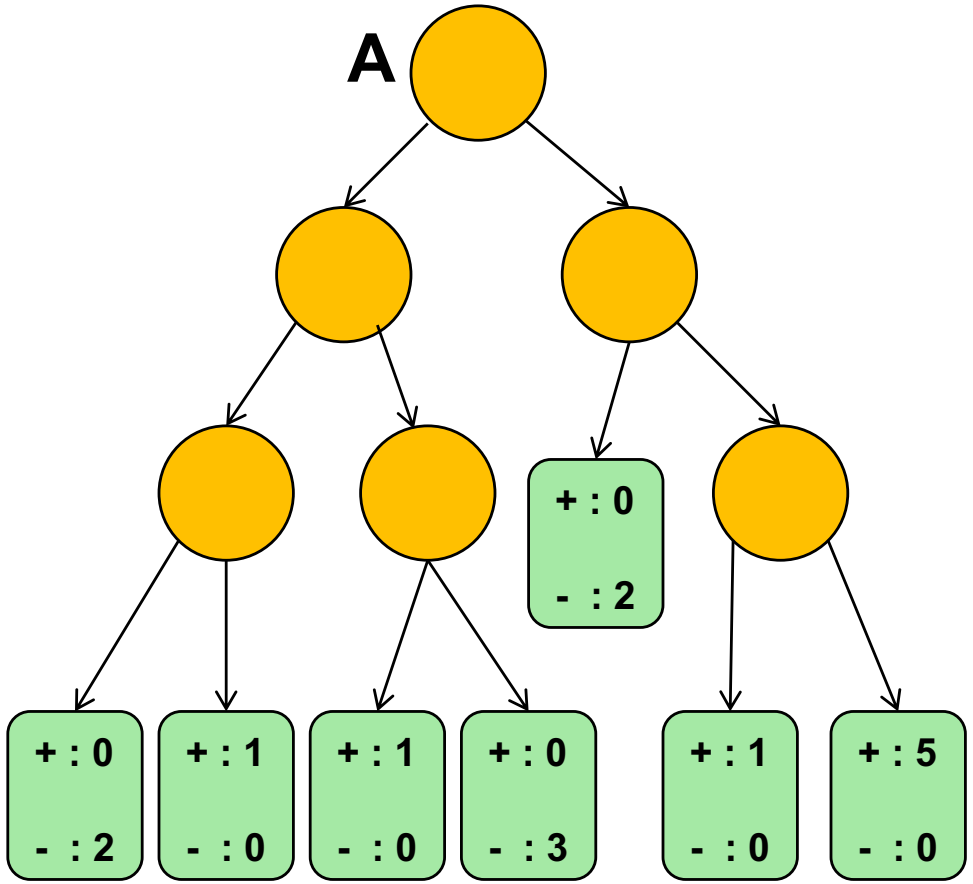
- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

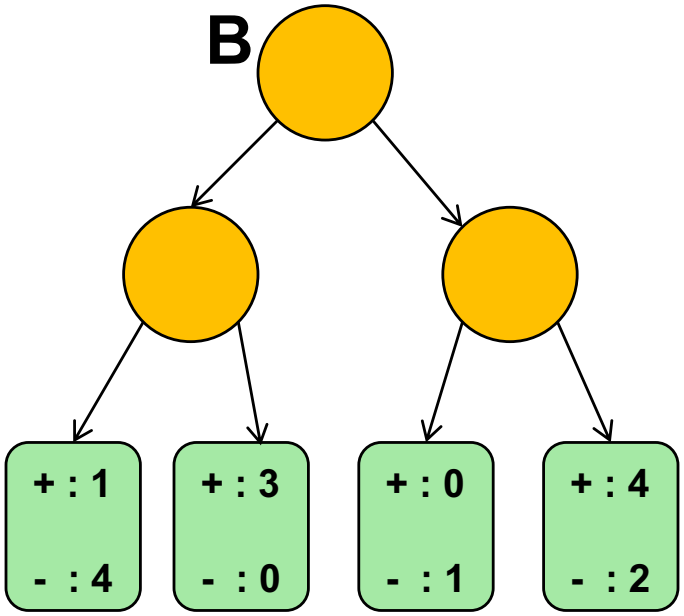
- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Practical Issues of Classification

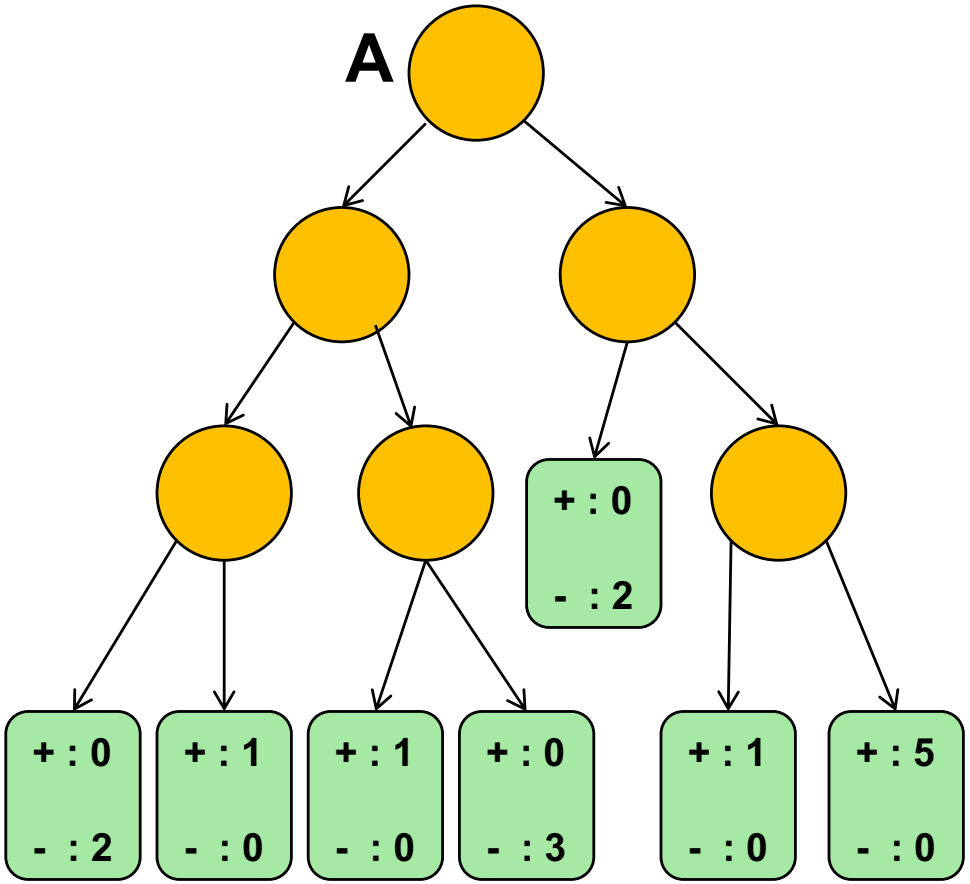
- Underfitting and Overfitting
- Performance Evaluation
  - Costs of Classification

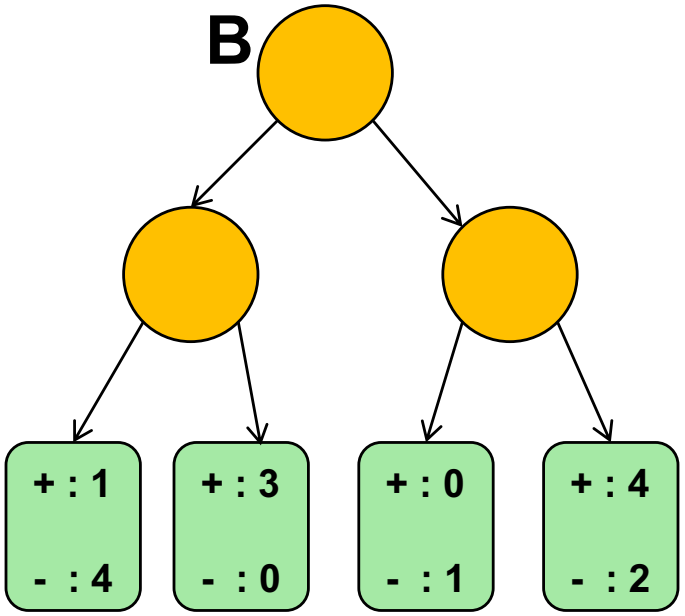
Which one is better?





Training Errors



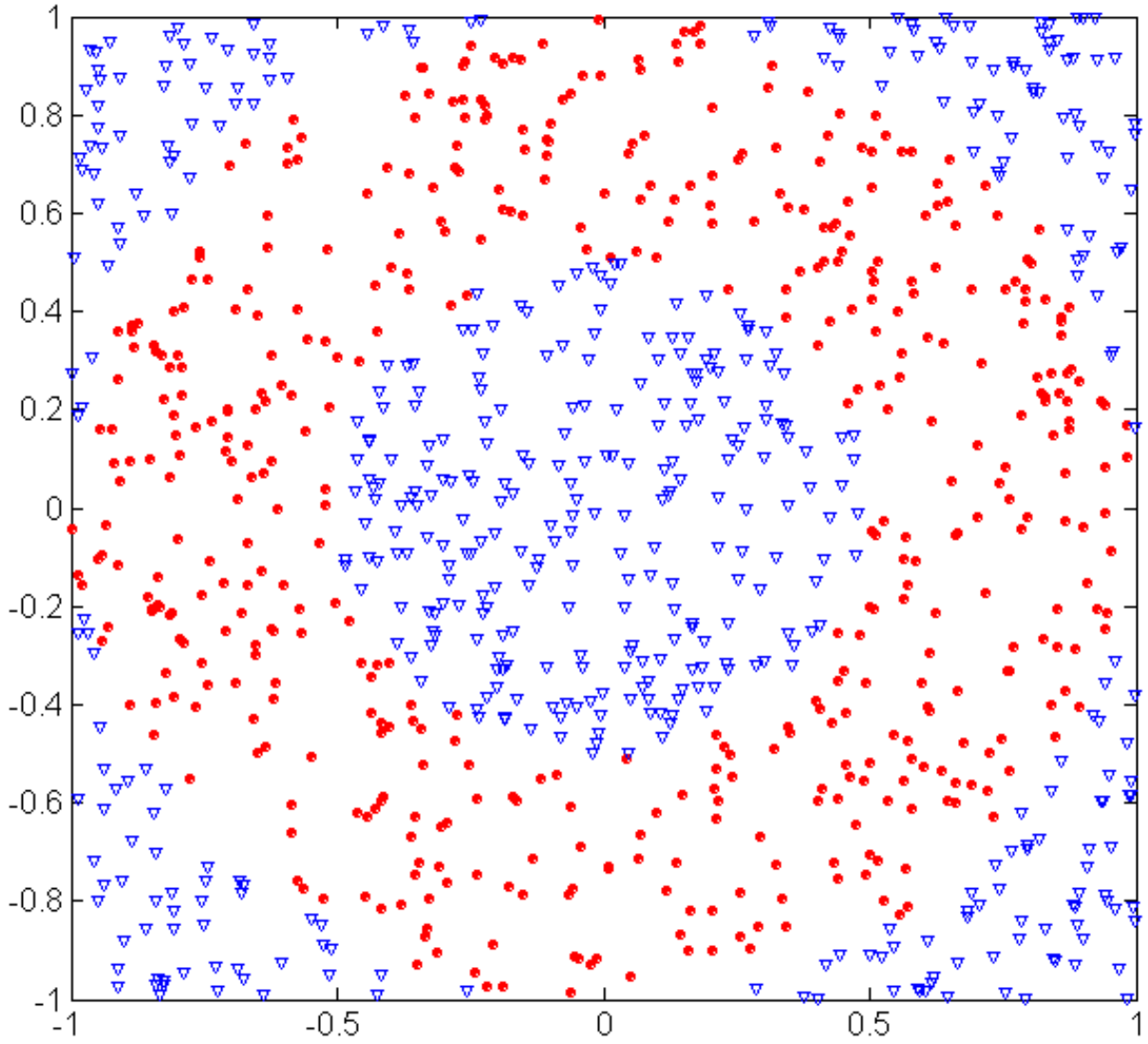


- Training error rates:
  - A = 0/15 = 0
  - B = 3/15 = 0.2

Model Overfitting

- Training errors
  - The number of misclassifications committed on training records
- Generalization errors
  - The expected errors on previously unseen records
- A good model makes *accurate prediction*!
  - must have low training and generalization errors
- A model may fit the training data very well but result in a poor result (high generalization errors)
  - This is known as *model overfitting*

Underfitting and Overfitting (Experiment)



500 circular and 500 triangular data points.

Circular points:

$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

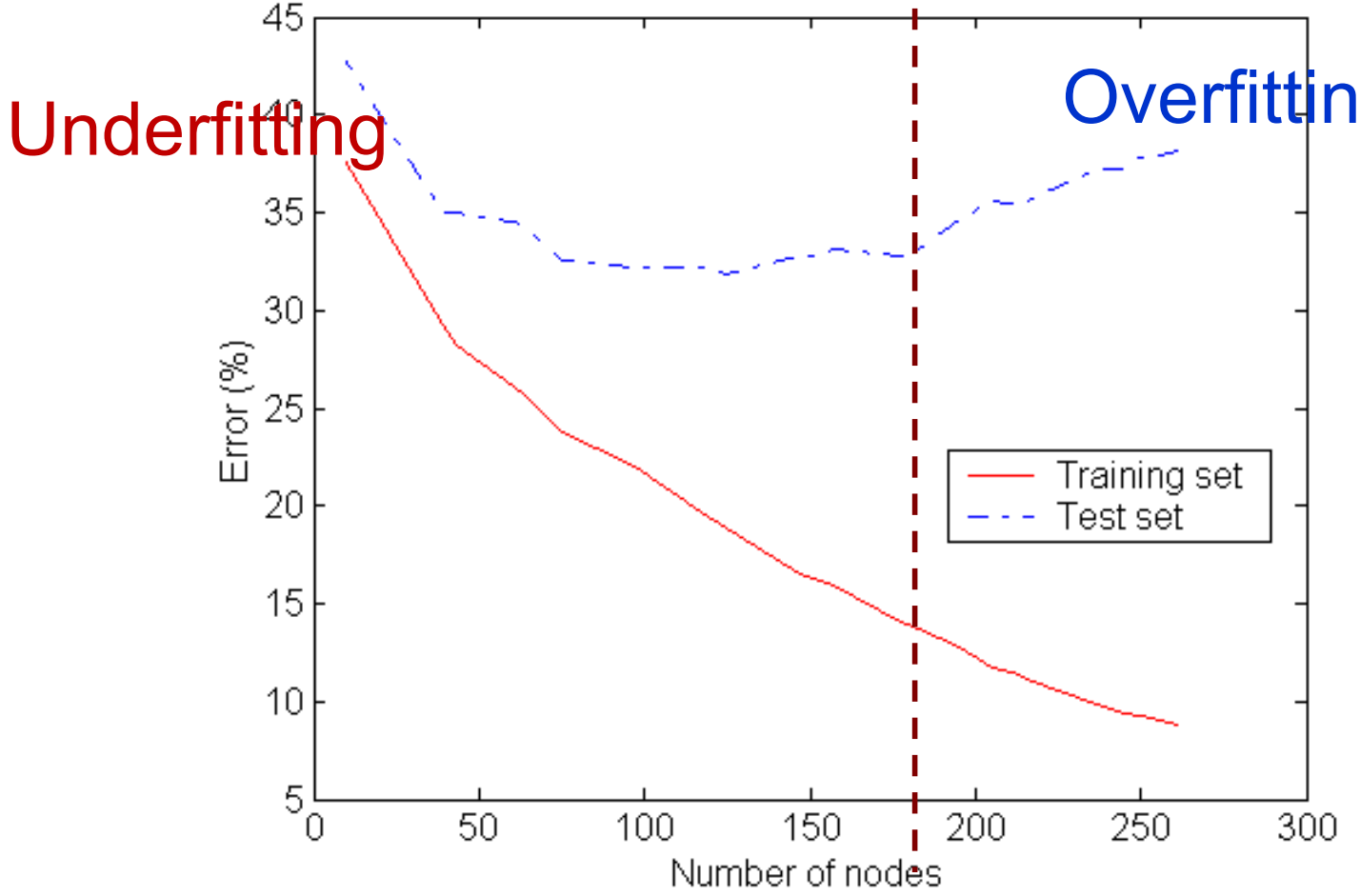
Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$

or

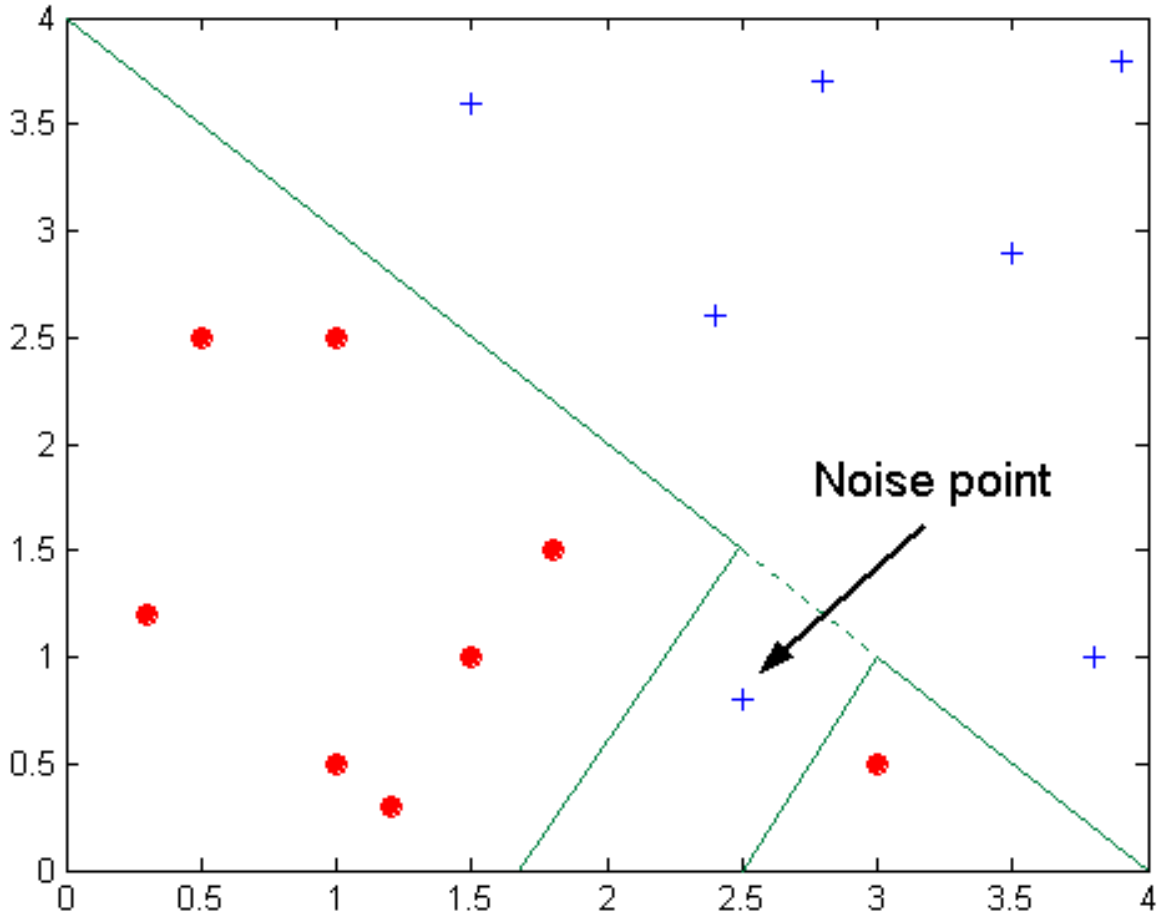
$\text{sqrt}(x_1^2+x_2^2) < 1$

Overfitting and Underfitting



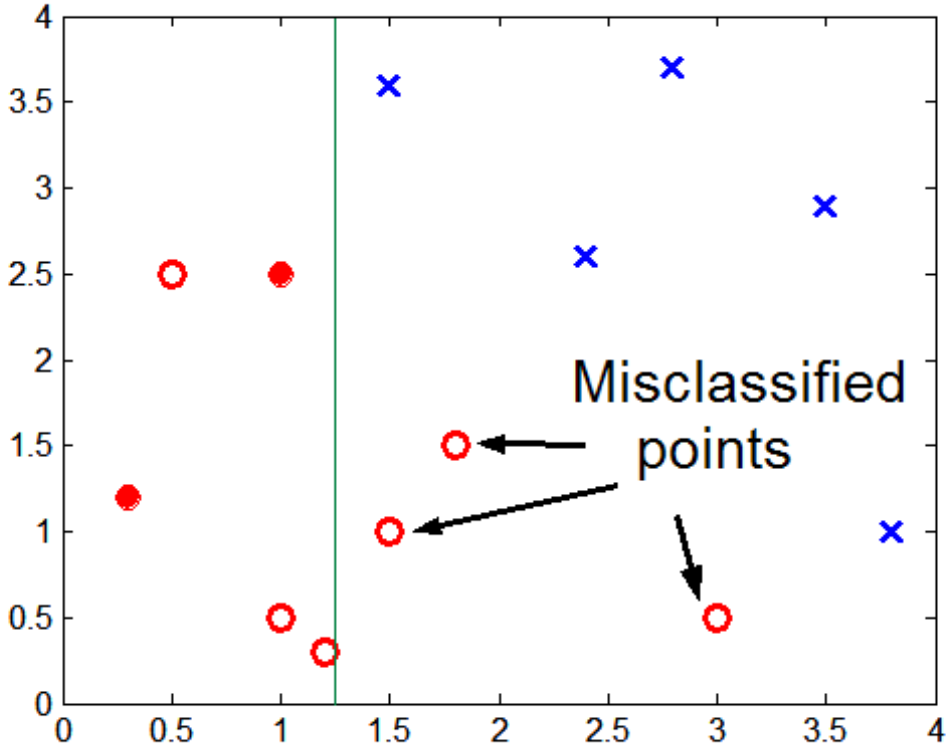
Underfitting: when the (decision tree) model is too simple (not capturing the data very well), both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



- Blue x and red circles are training data and white circles are testing data
- Lack of data points in the lower half of the diagram makes the prediction difficult
- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
  - Need new ways for estimating errors
- Is there a good way to estimate generalization errors?
  - Use training errors as the basis and consider the model (e.g., decision tree) complexity in the estimate

Occam's Razor

- *Given two models of similar generalization errors, one should prefer the simpler model over the more complex model*

- For complex models, there is a greater chance that it was fitted accidentally by errors in data

- Therefore, one should include model complexity when evaluating a model

Estimating Generalization Errors

- Training errors *e(T)*: error on training a Dec. Tree *T*
- Generalization errors *e'(T)*: error on testing *T*
- Methods for estimating generalization errors:
  - Optimistic approach: Use training error: *e'(T) = e(T)*
  - Pessimistic approach:
    - For each leaf *t*, add a penalty Ω: *e'(t) = (e(t)+Ω (t))*
    - Total errors: e'(T) = e(T) + N × 0.5
      (N: number of leaf nodes, *Ω(t)) =0.5*)
    - For a tree with 30 leaf nodes & 10 errors on training (out of 1000 instances):
      Training error = 10/1000 = 1%
      Generalization error = (10 + 30×0.5)/1000 = 2.5%
  - Empirical approach
    - uses a validation data set, e.g., 2/3 of the original data set, to estimate generalization error..

How to Address Overfitting

- *Pre-Pruning (Early Stopping Rule)*
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions (also consider model/tree complexity):
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting…

- *Post-pruning*
  - Grow decision tree to its entirety, then trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error (e.g., measured permissive approach) improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree

Example of Post-Pruning

Training Error (Before splitting) = 10/30

Pessimistic error = (10 + 0.5)/30 = 10.5/30

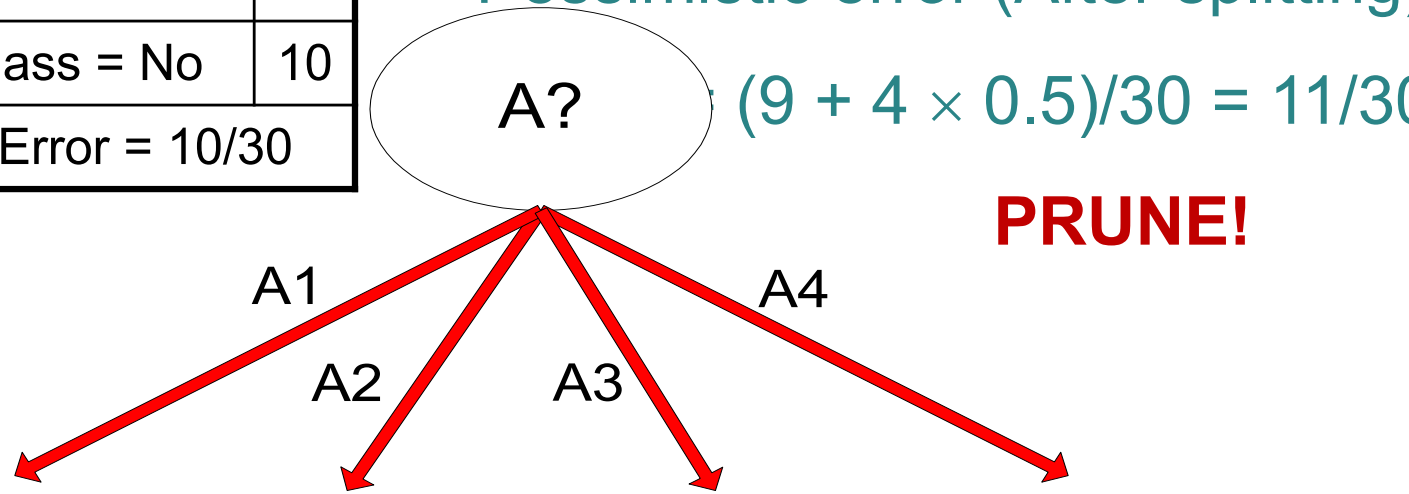
Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!

Class = Yes	20
Class = No	10
Error = 10/30	



Class = Yes	8
Class = No	4

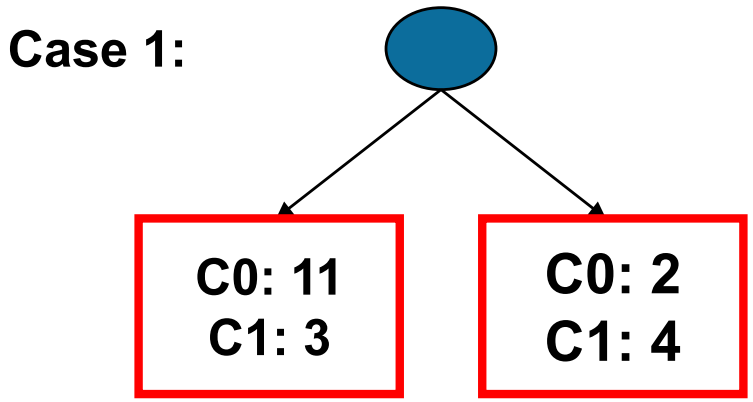
Class = Yes	3
Class = No	4

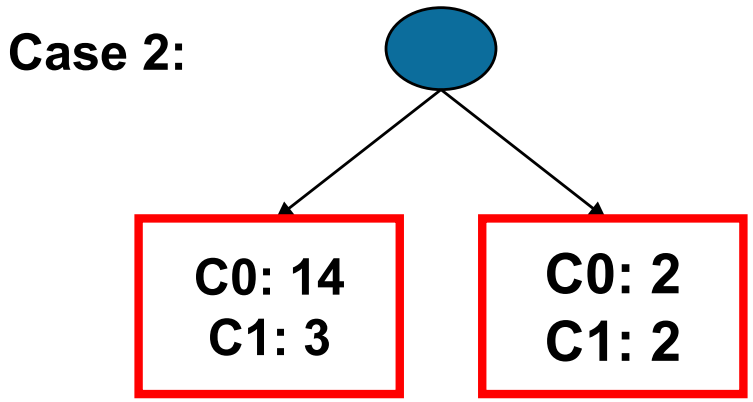
Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Examples of Post-pruning

- Optimistic error?
  - Don't prune for both cases
- Pessimistic error?
  - Don't prune case 1, prune case 2
- Reduced error pruning?
  - Depends on validation set





Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?

Metrics for Performance Eval.

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive) b: FN (false negative)

c: FP (false positive) d: TN (true negative)

Metrics for Performance Eval.

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Limitation of Accuracy

- Treat every class equally important – may not be suitable for imbalanced data set
- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example
- The performance metric (accuracy) by definition do not taking account all cases in confusion matrix.
  - Can give them weights/costs.

Cost Matrix

		PREDICTED CLASS	
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Cost assigned to penalize errors and reward success!

This allows performance measure to be customized.

Cost Measure of Classification

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

Cost vs. Accuracy

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$\text{Cost} = p\,(a + d) + q\,(b + c)$

$= p\,(a + d) + q\,(N - a - d)$

$= q\,N - (q - p)(a + d)$

$= N\,[q - (q\text{-}p) \times \text{Accuracy}]$

Alternative Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

Precision is biased towards C(Yes|Yes) & C(Yes|No)

Recall is biased towards C(Yes|Yes) & C(No|Yes)

F-measure, a harmonic mean of precision and recall, is biased towards all except C(No|No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Methods of Performance Evaluation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random Subsampling
  - Repeated holdout for several times for average
- Cross Validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=N (entire dataset)
- Stratified Sampling
  - Ensures at least one observation is picked in each group
- Bootstrap
  - Sampling with replacement
  - A sample of size N contains 63.2% of the original data
  - Records not in the sample becomes part of test set