PENNSTATE

Chapter 4-5
Classification Analysis

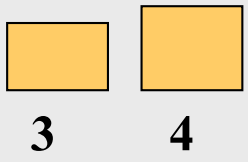
Presentation extended from the slides of the textbook, Introduction to Data Mining by Tan et al. and supplementary material

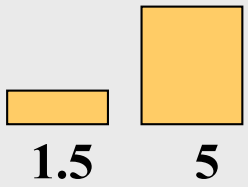
Overview

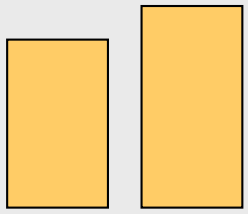
- Basics
- Decision Trees
- Model Overfitting
- Model Evaluation
- Rule-Based Classifier
- Instance-Based Classifier
- Bayesian Classifiers
- Artificial Neural Networks Optional what is doing
- Support Vector Machines Optional

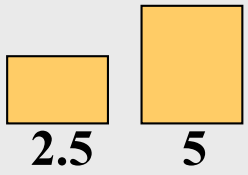
Toy Problem 1

Examples of class A

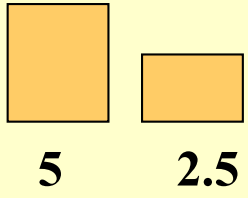


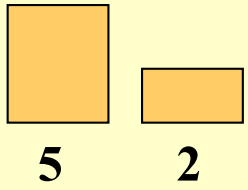


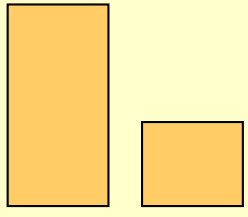


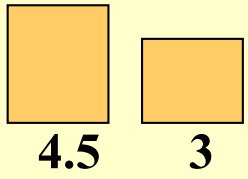


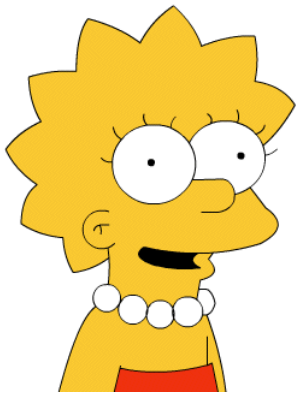
Examples of class B



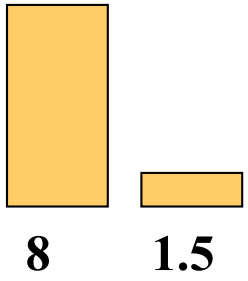




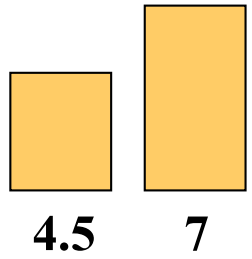




What class is this object?

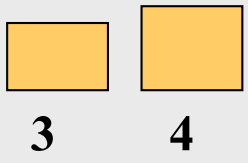


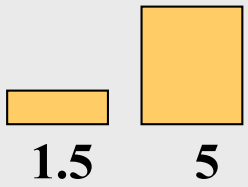
What about this one, A or B?

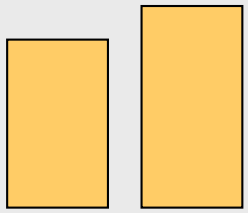


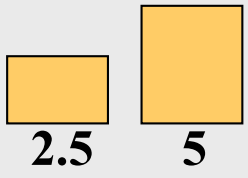
Toy Problem 1

Examples of class A

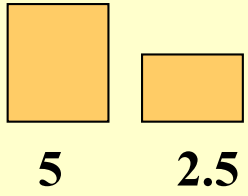


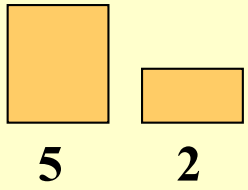


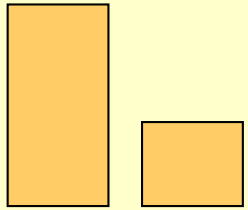


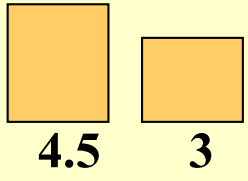


Examples of class B

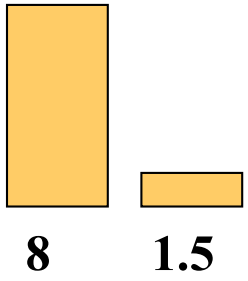








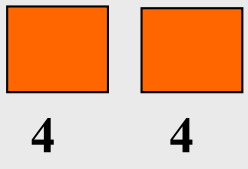
This is a B!

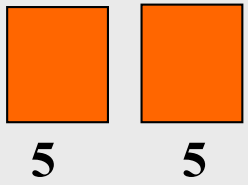


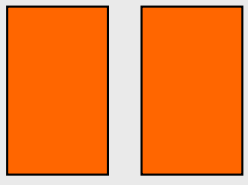
Here is the rule. If the left bar is smaller than the right bar, it is an A, otherwise it is a B.

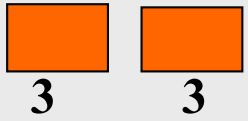
Toy Problem 2

Examples of class A

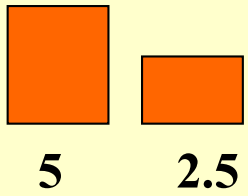


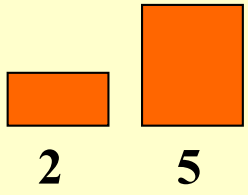


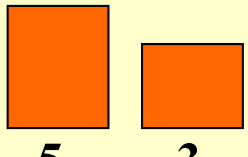


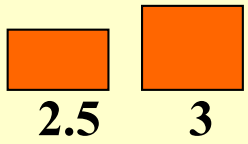


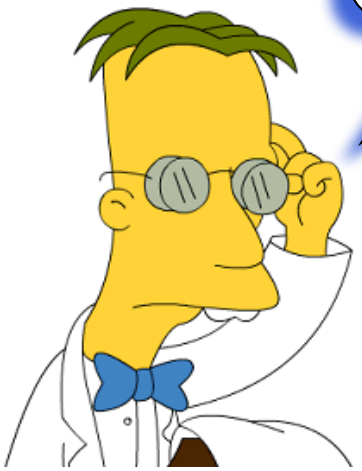
Examples of class B

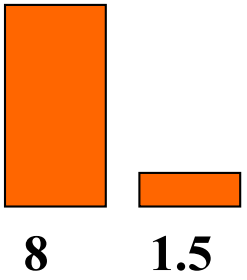


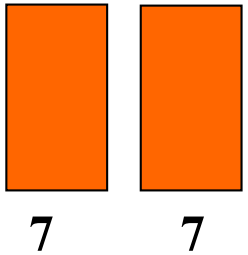






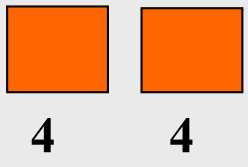


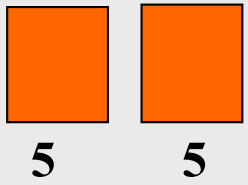


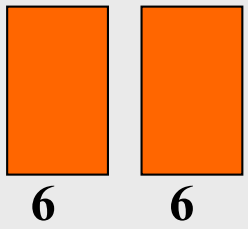


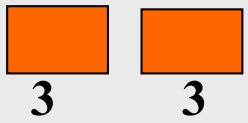
Toy Problem 2

Examples of class A

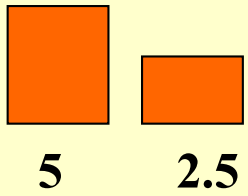


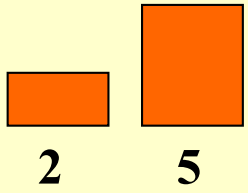


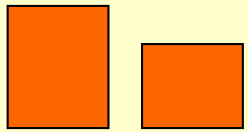


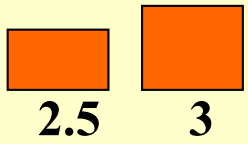


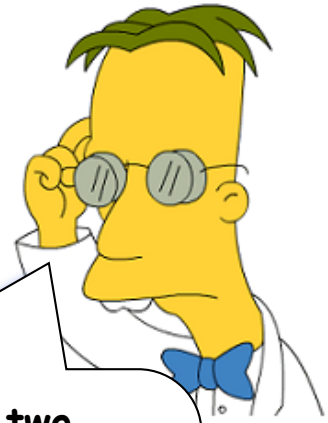
Examples of class B





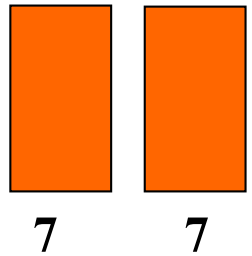






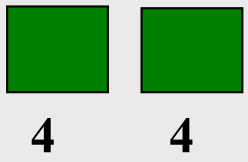
The rule is as follows, if the two bars are equal sizes, it is an **A**. Otherwise it is a **B**.

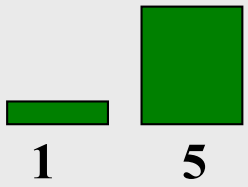
So this one is an **A**.

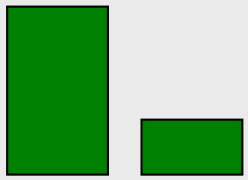


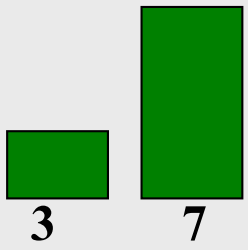
Toy Problem 3

Examples of class A

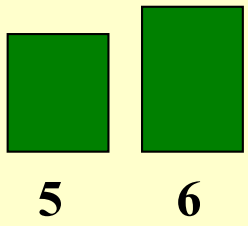


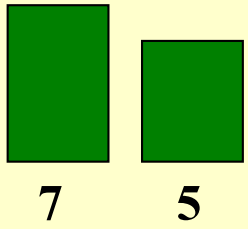


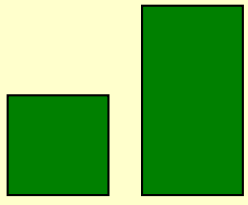


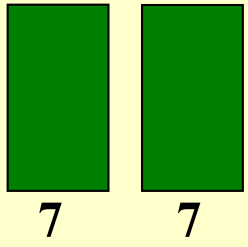


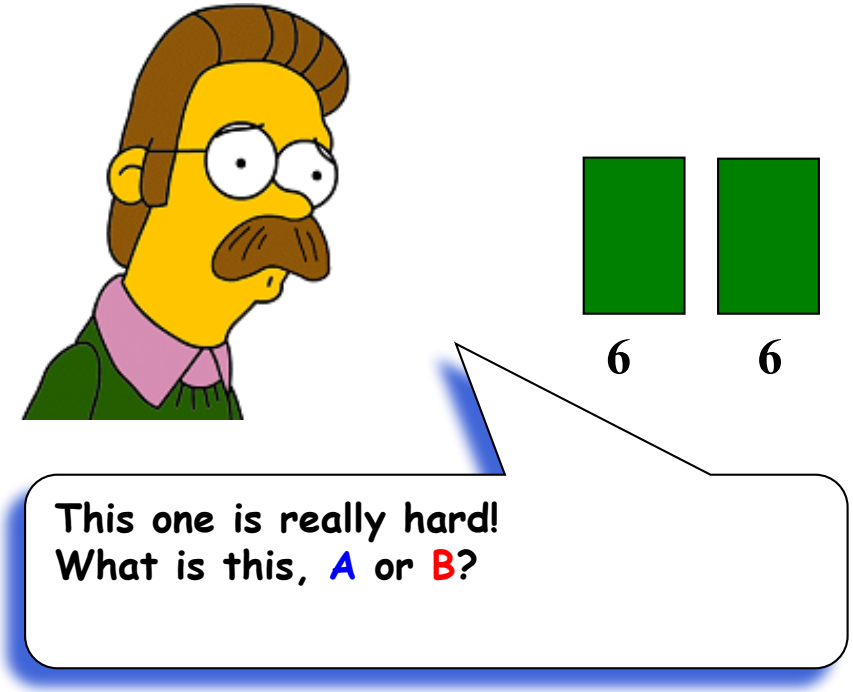
Examples of class B





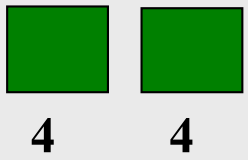


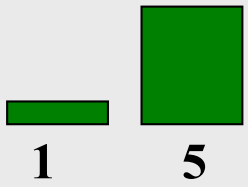


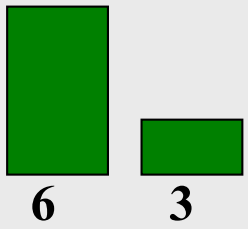


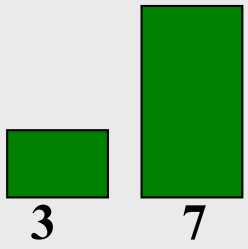
Toy Problem 3

Examples of class A

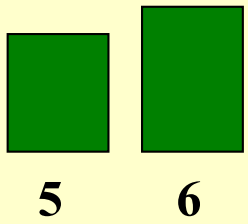


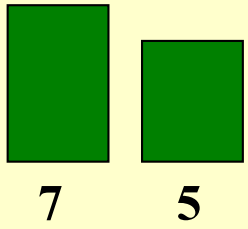


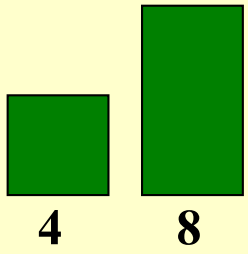


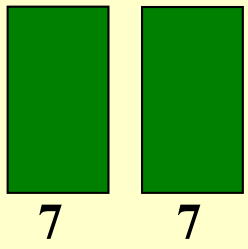


Examples of class B

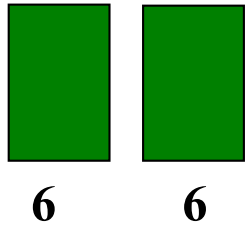








It is a B!

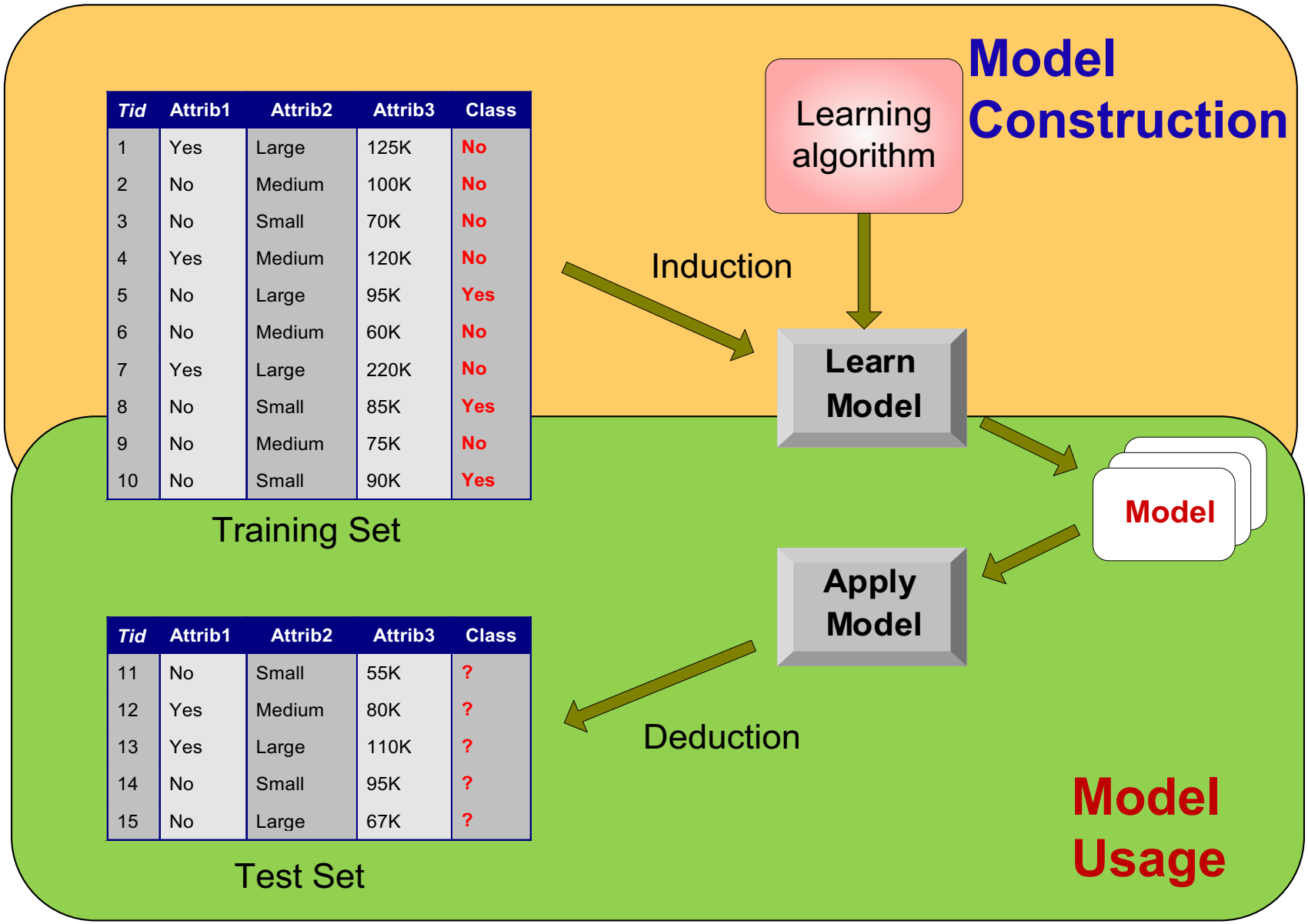


The rule is as follows, if the square of the sum of the two bars is less than or equal to 10, it is an A. Otherwise it is a B.

Classification: Definition

- Given: a collection of records (*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

Training Set

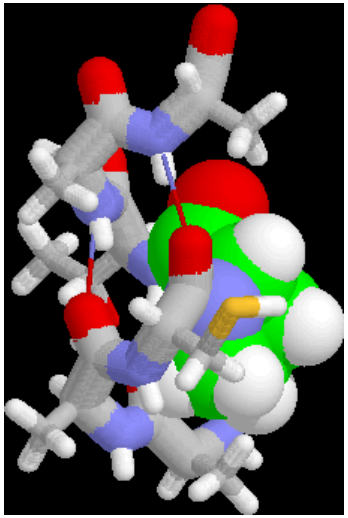
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

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

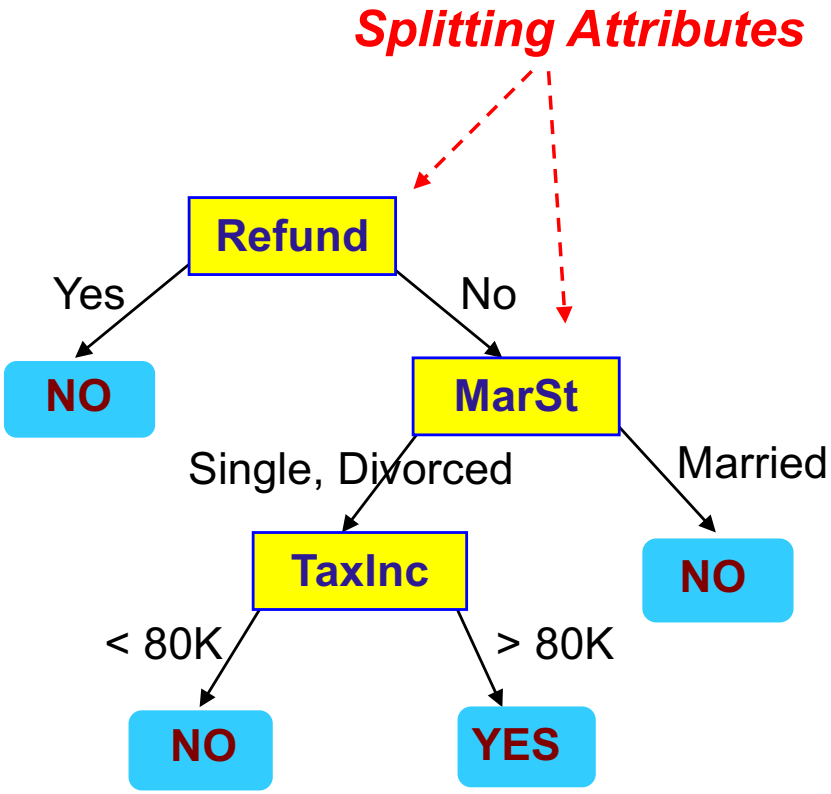
Classification Techniques

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

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

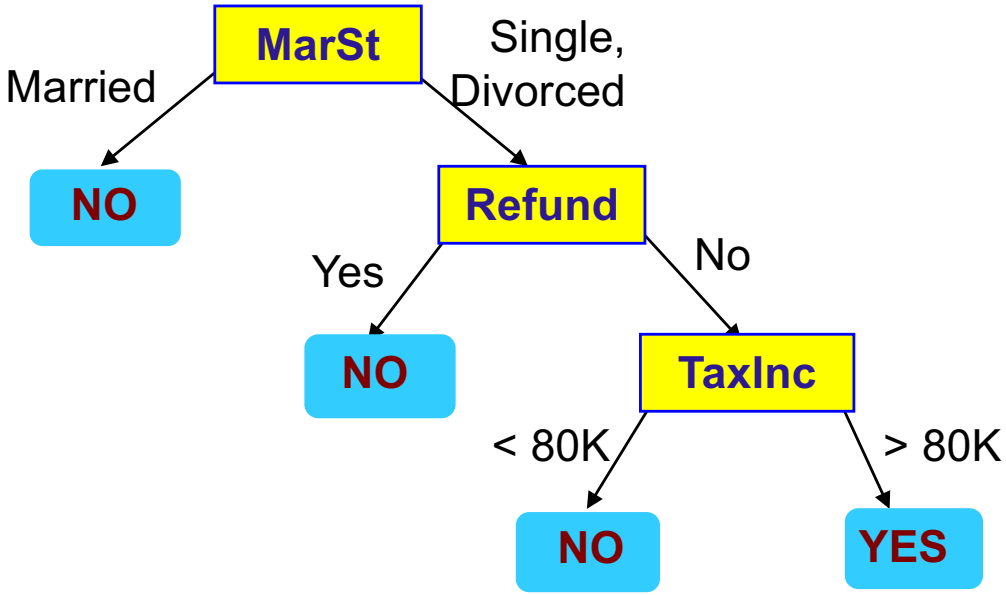
Training Data



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!

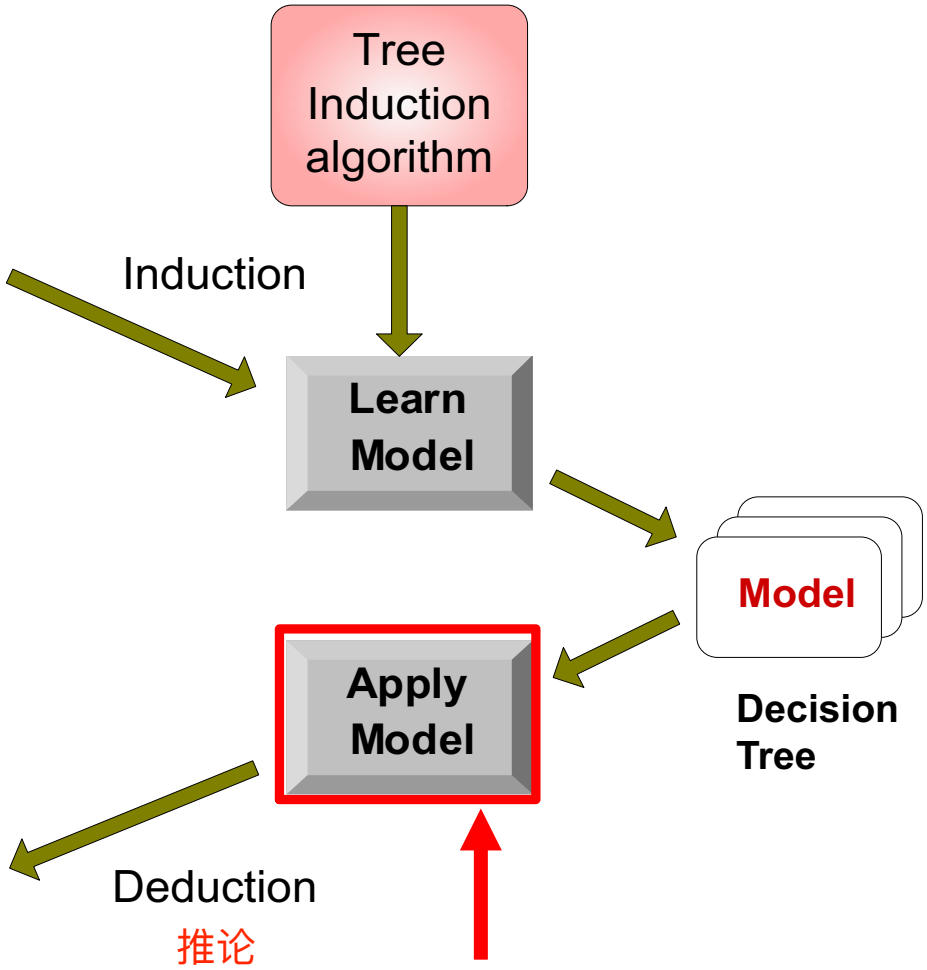
Decision Tree Classification Task

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

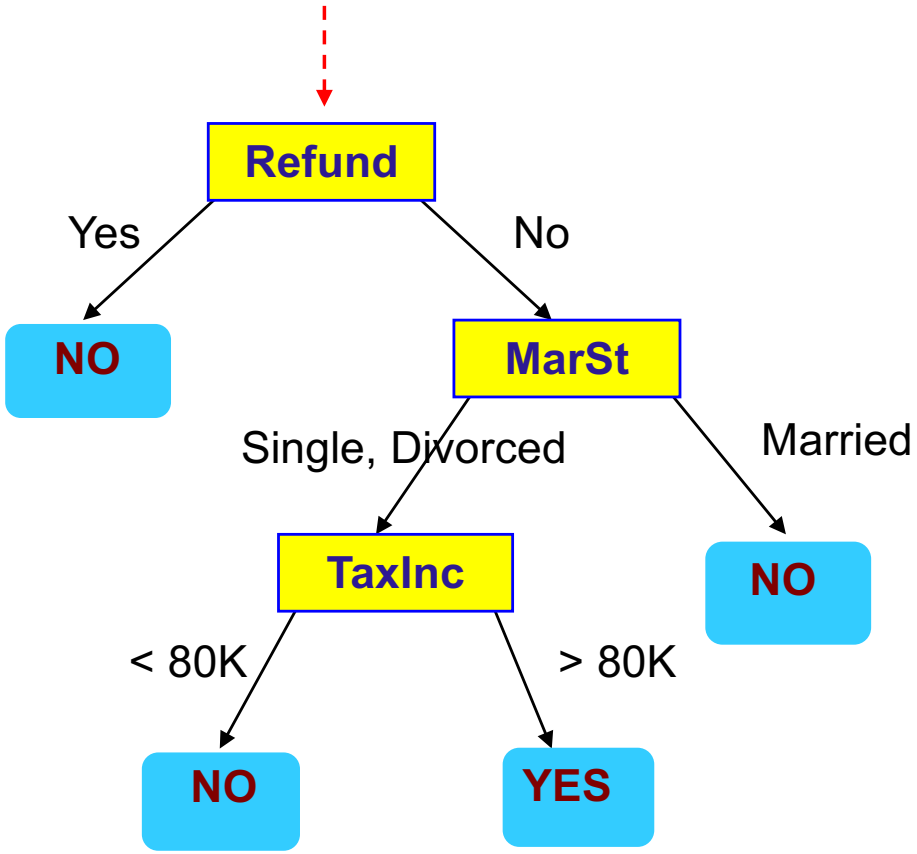
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

Start from the root of tree.



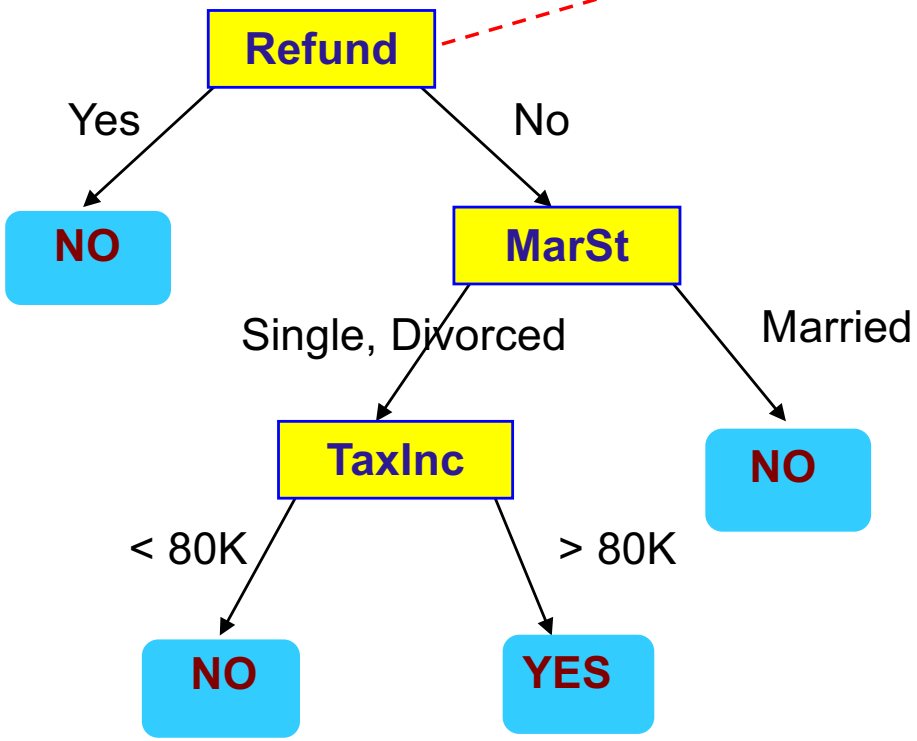
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

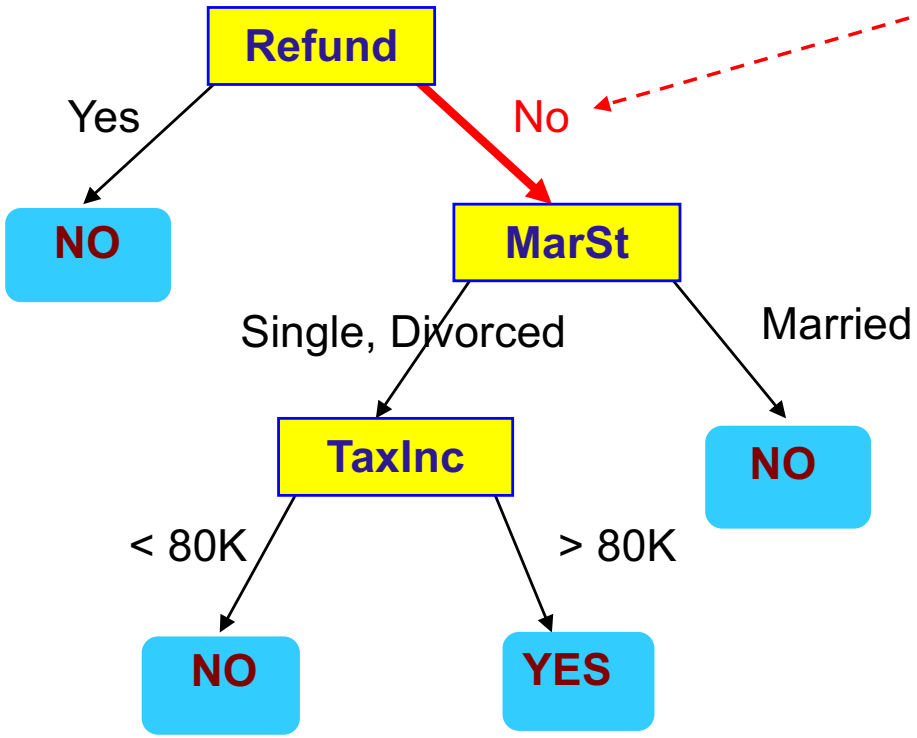
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

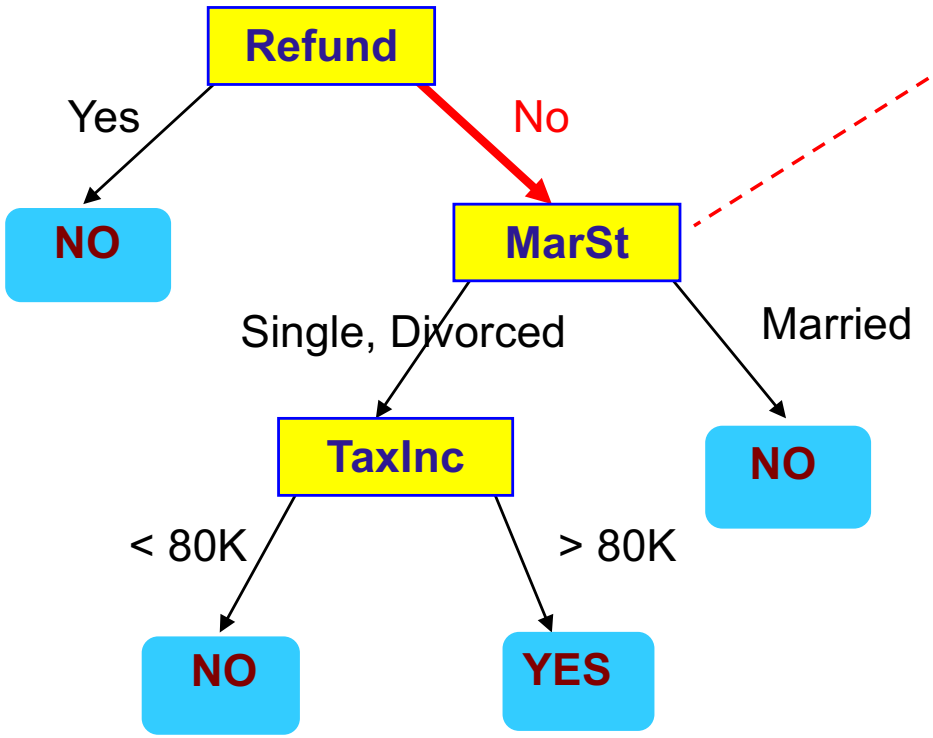
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

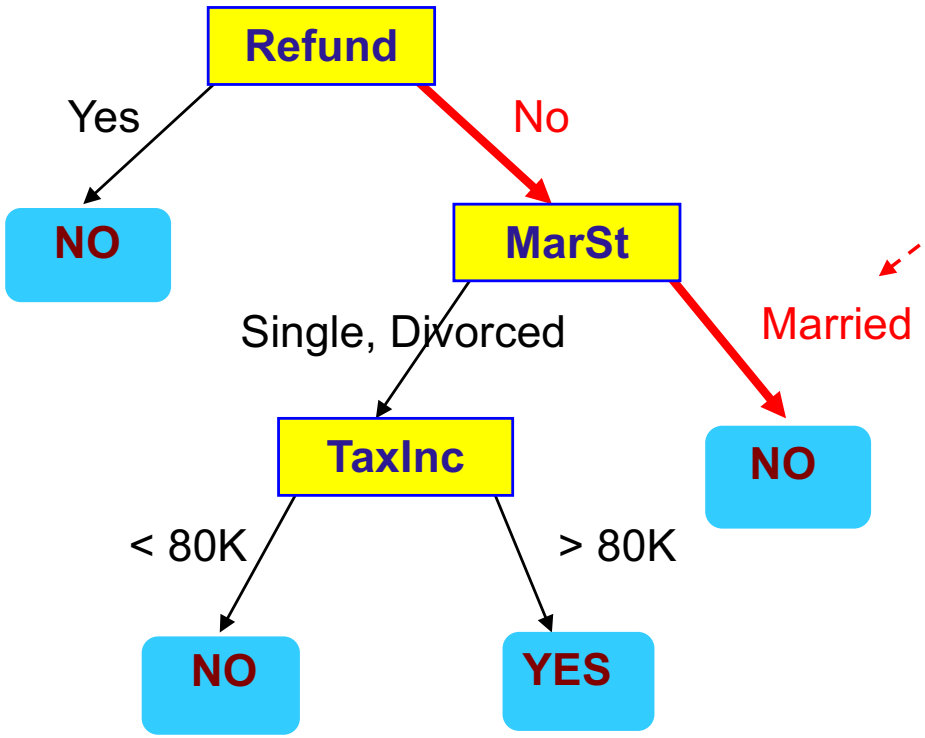
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

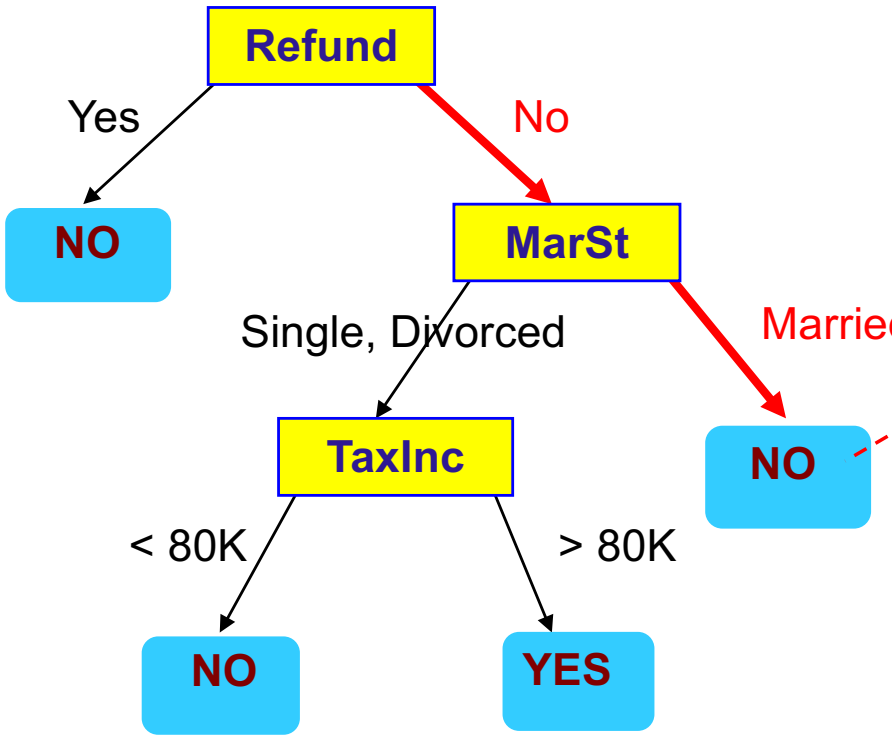
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

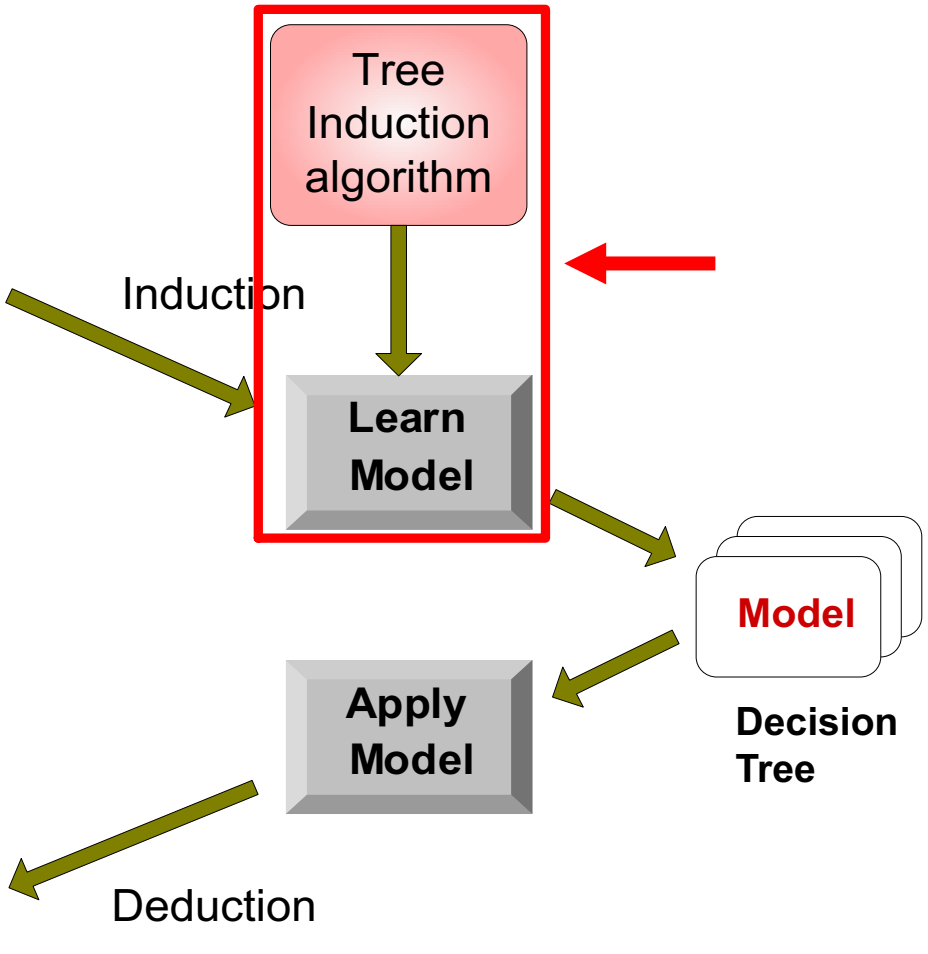
Decision Tree Induction Task

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



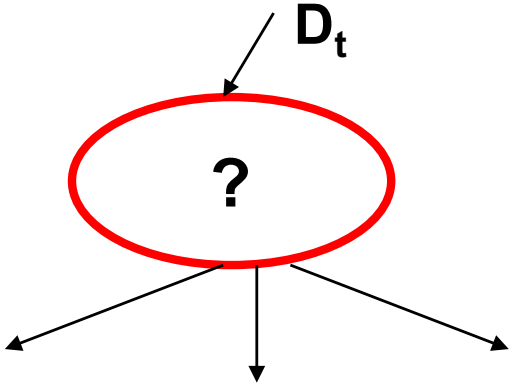
Decision Tree Induction

- 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

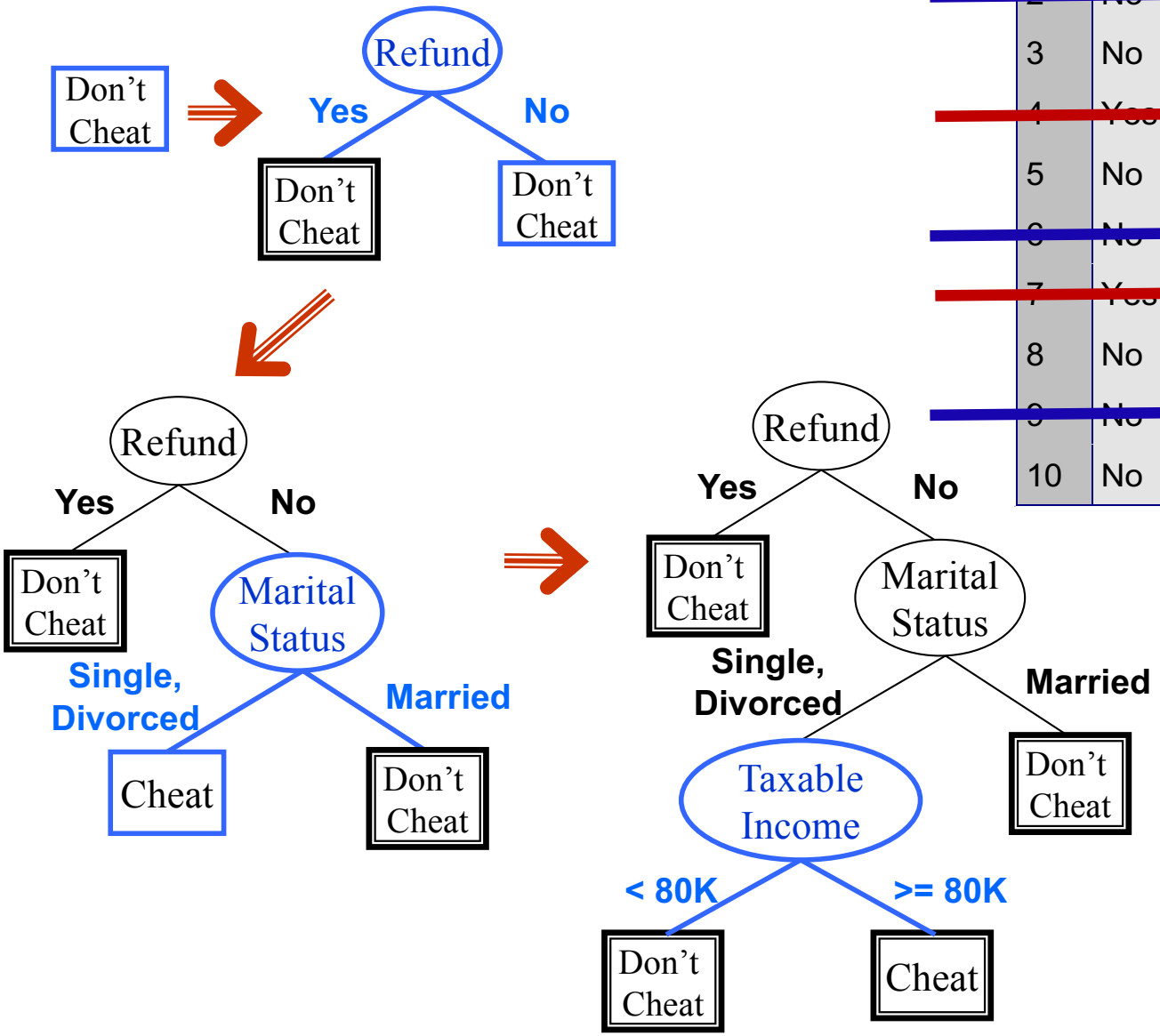
- Let D_t be the set of training records that reach a node t. *Recursively* apply the following:
  - If D_t contains records that all belong to class y_t, then t is a leaf node labeled as y_t
  - If D_t is empty, t is a leaf node labeled by default class, y_d, (majority class of parent node)
  - If D_t 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*

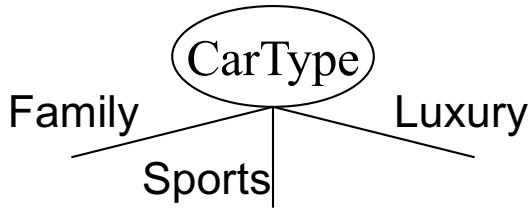
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

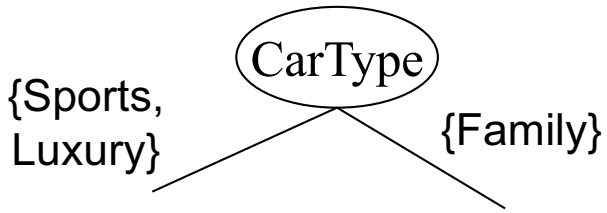
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

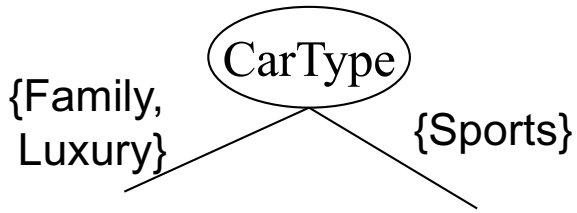
- Multi-way split: Use as many partitions as distinct values.



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

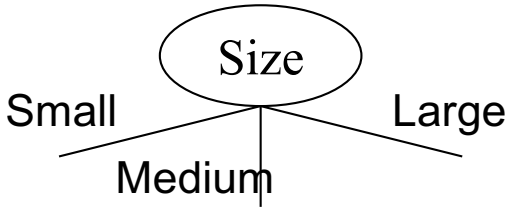


OR

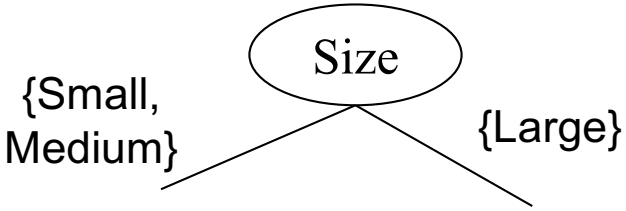


Splitting Based on Ordinal Attributes

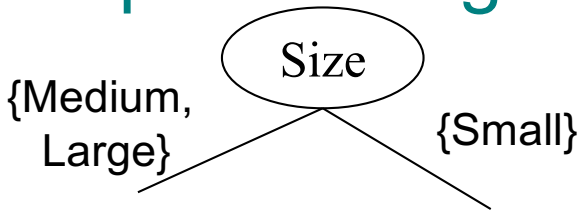
- Multi-way split: Use as many partitions as distinct values.



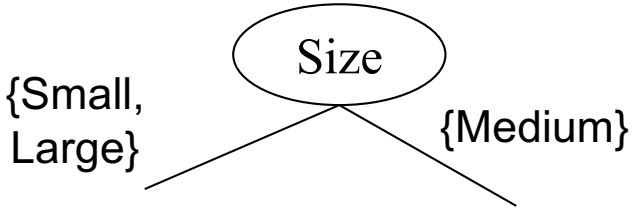
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



OR



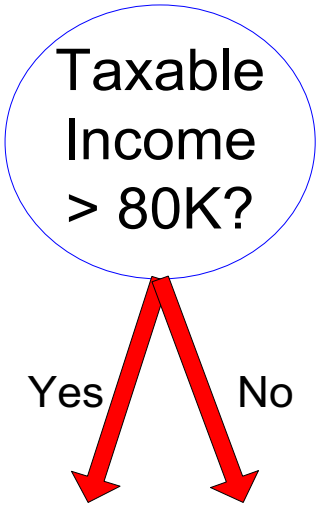
- What about this split?



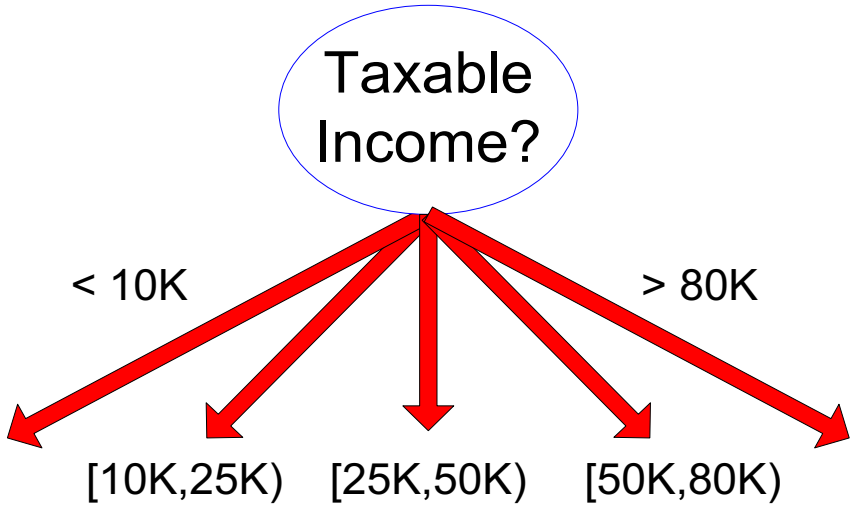
Splitting Based on Continuous Attributes

- 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

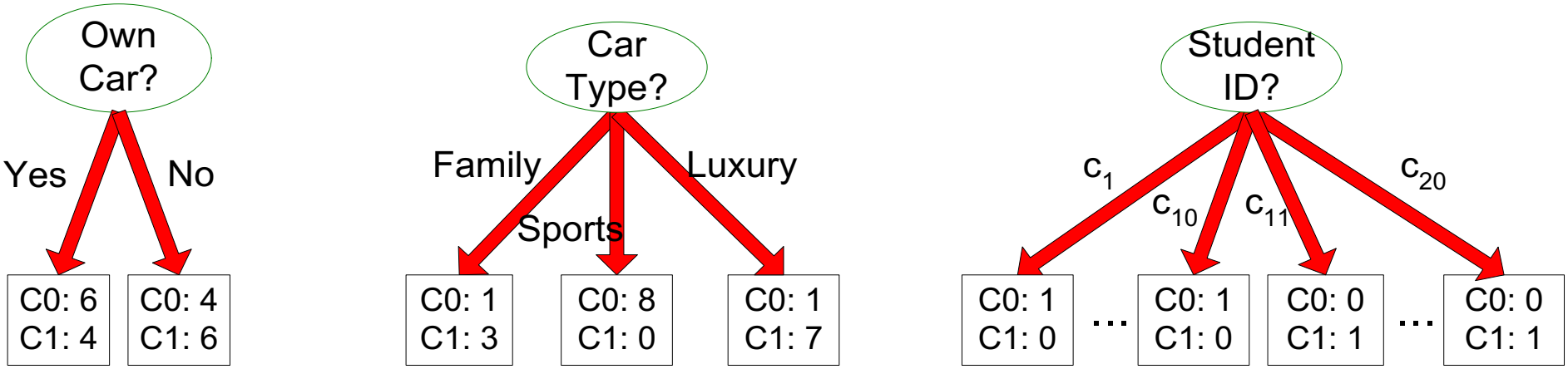


(ii) Multi-way split

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

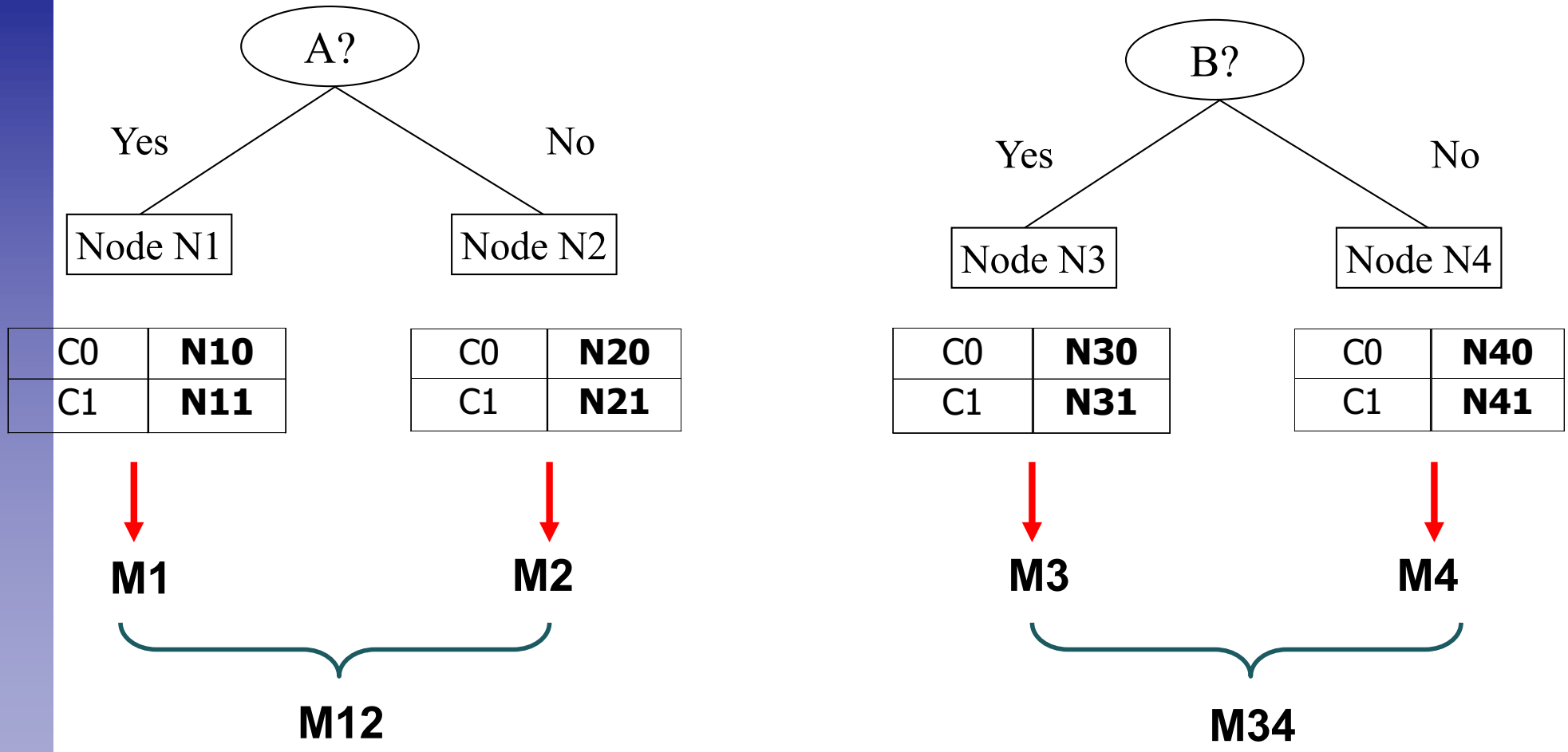
- Gini Index
- Entropy
- Misclassification error

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

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

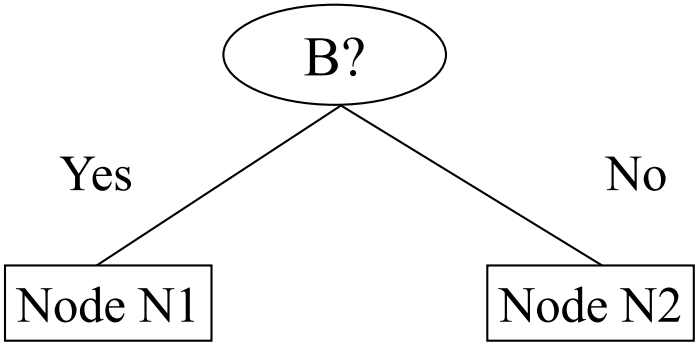
where, n_i = number of records at child i,

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)

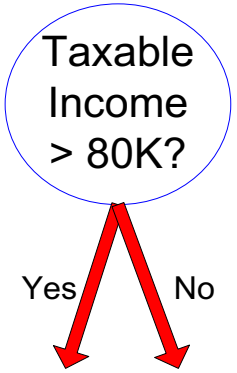
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	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
  - No. of possible splitting values = No. 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! Repetition of work.

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 (0.0) 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

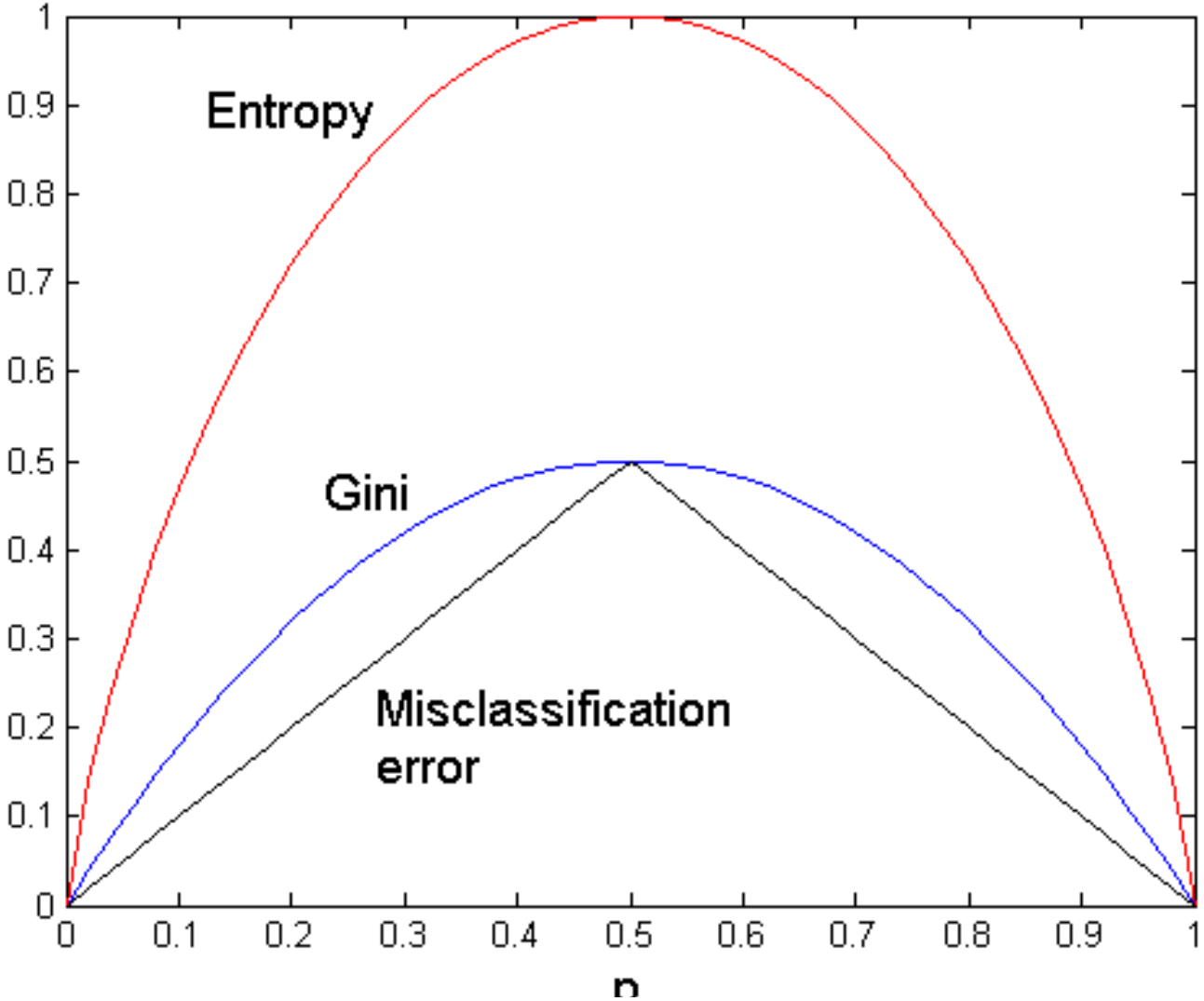
C1	**2**
C2	**4**

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

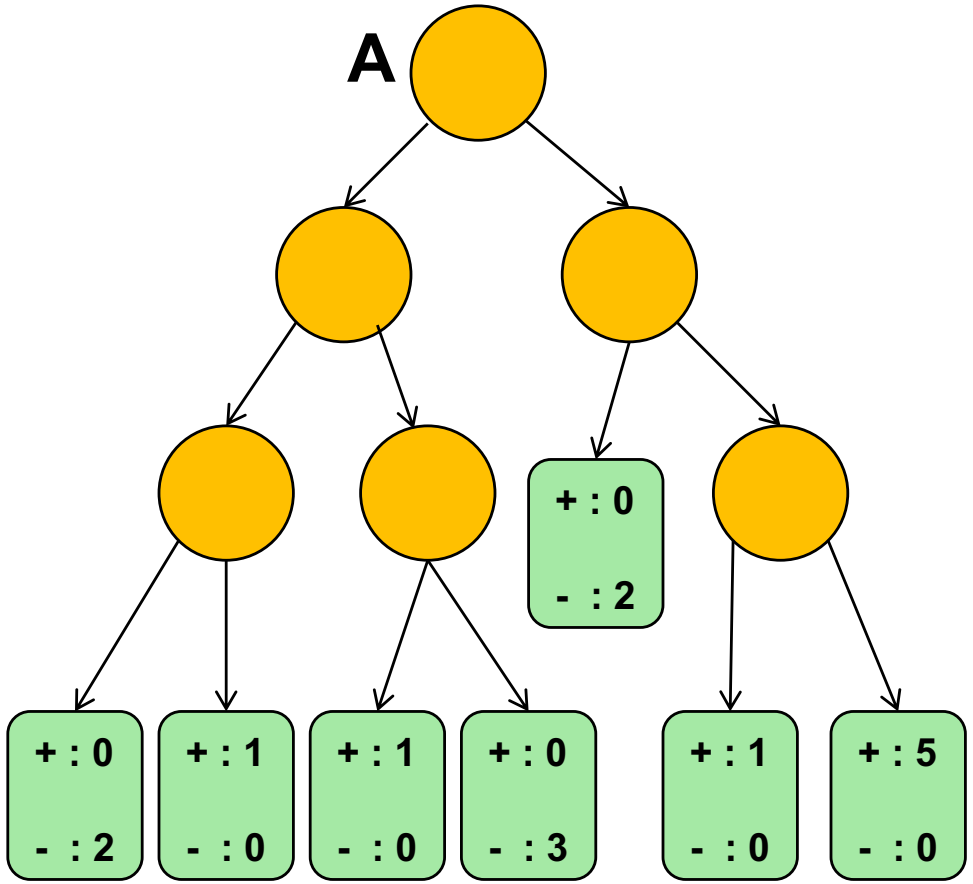
- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

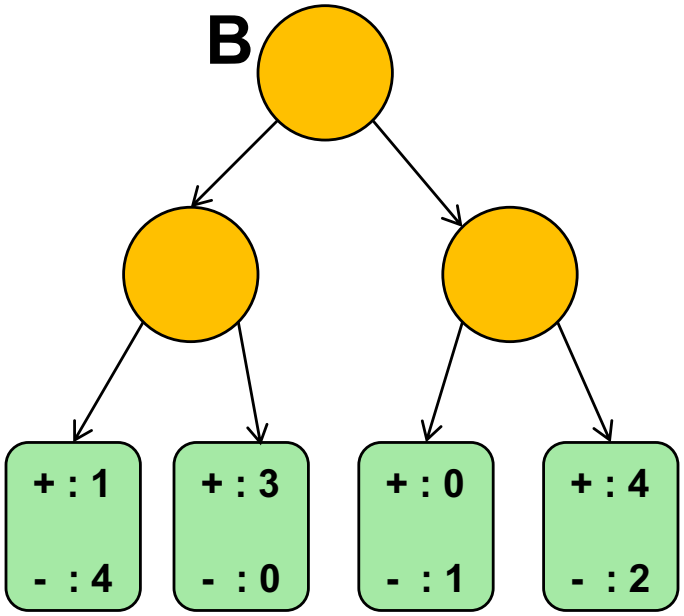
- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Practical Issues of Classification

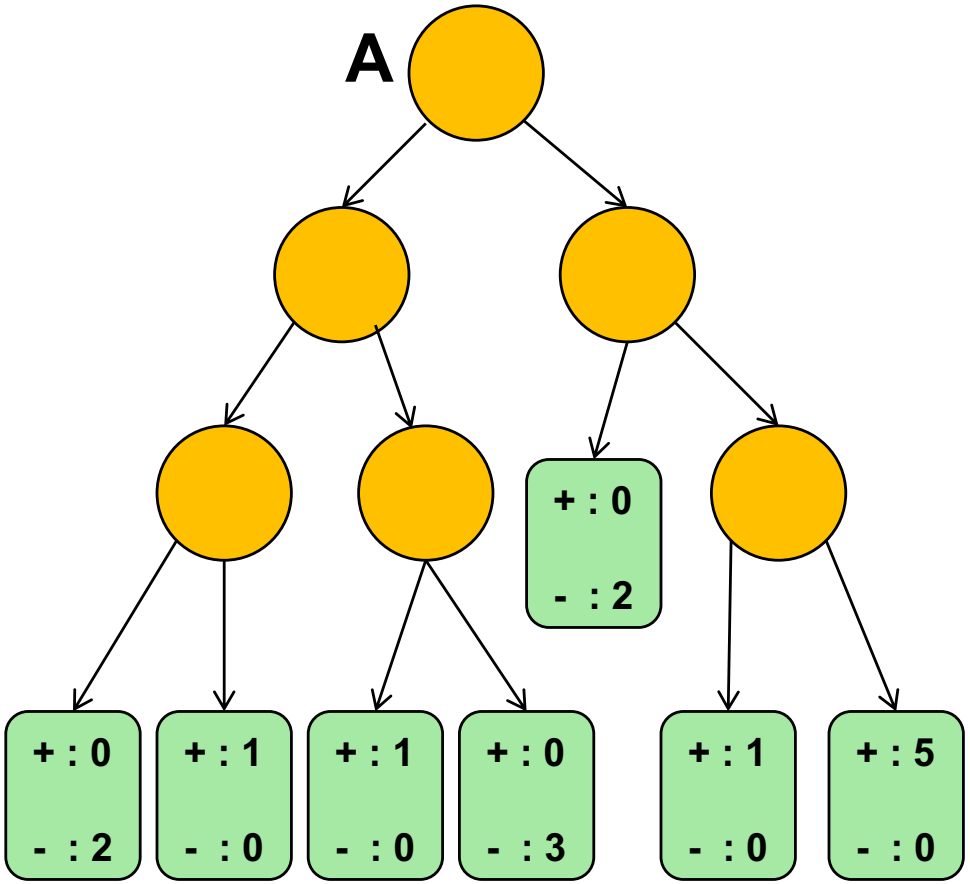
- Underfitting and Overfitting
- Performance Evaluation
  - Costs of Classification

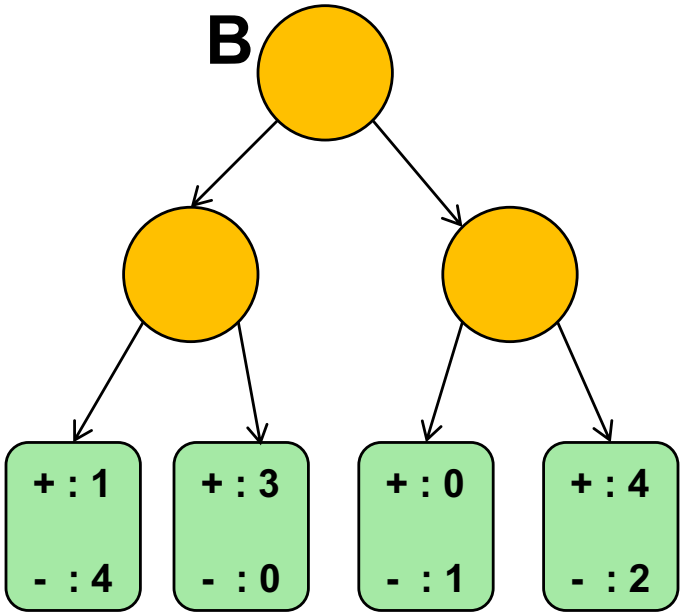
Which one is better?





Training Errors



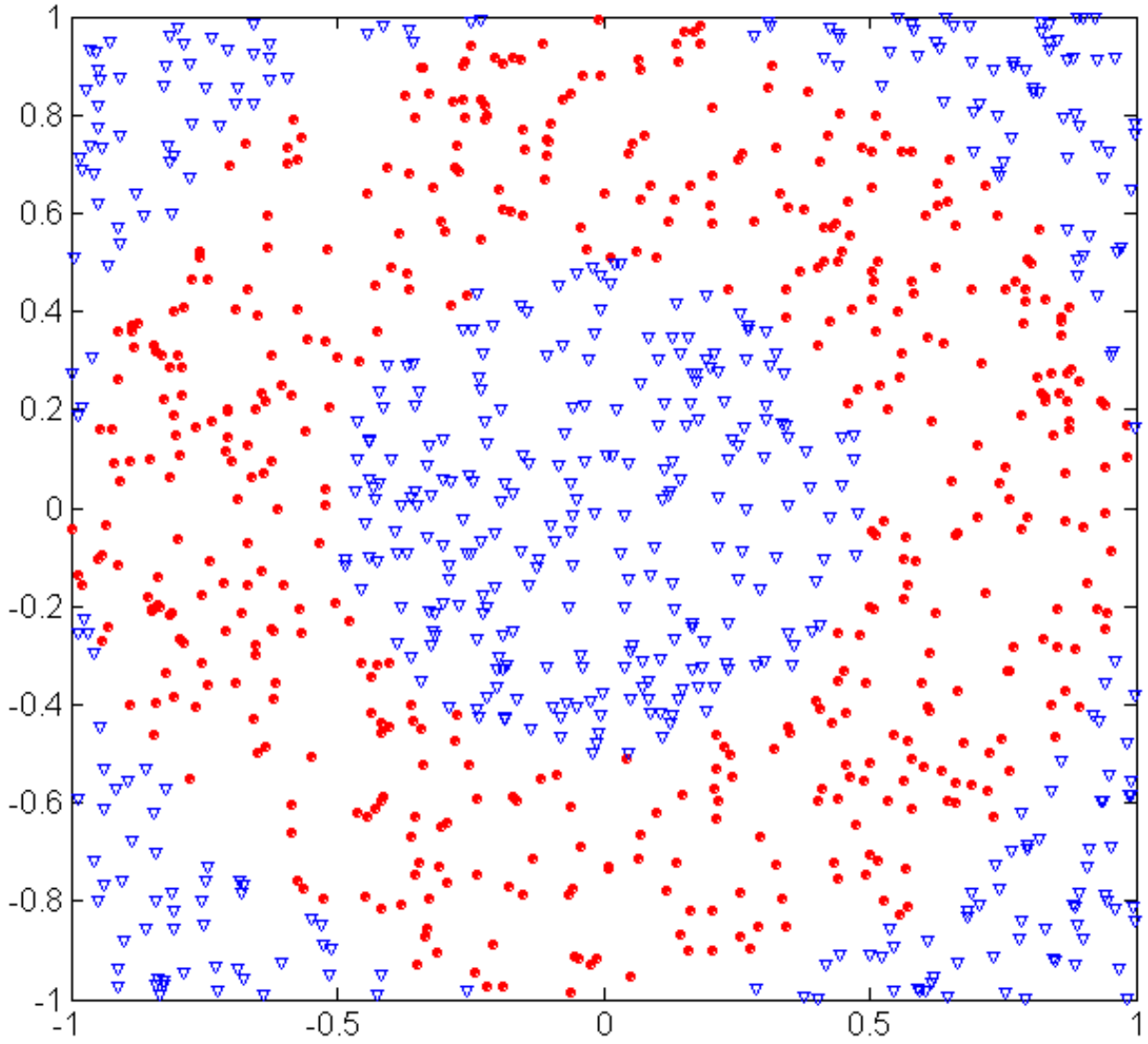


- Training error rates:
  - A = 0/15 = 0
  - B = 3/15 = 0.2

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:

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

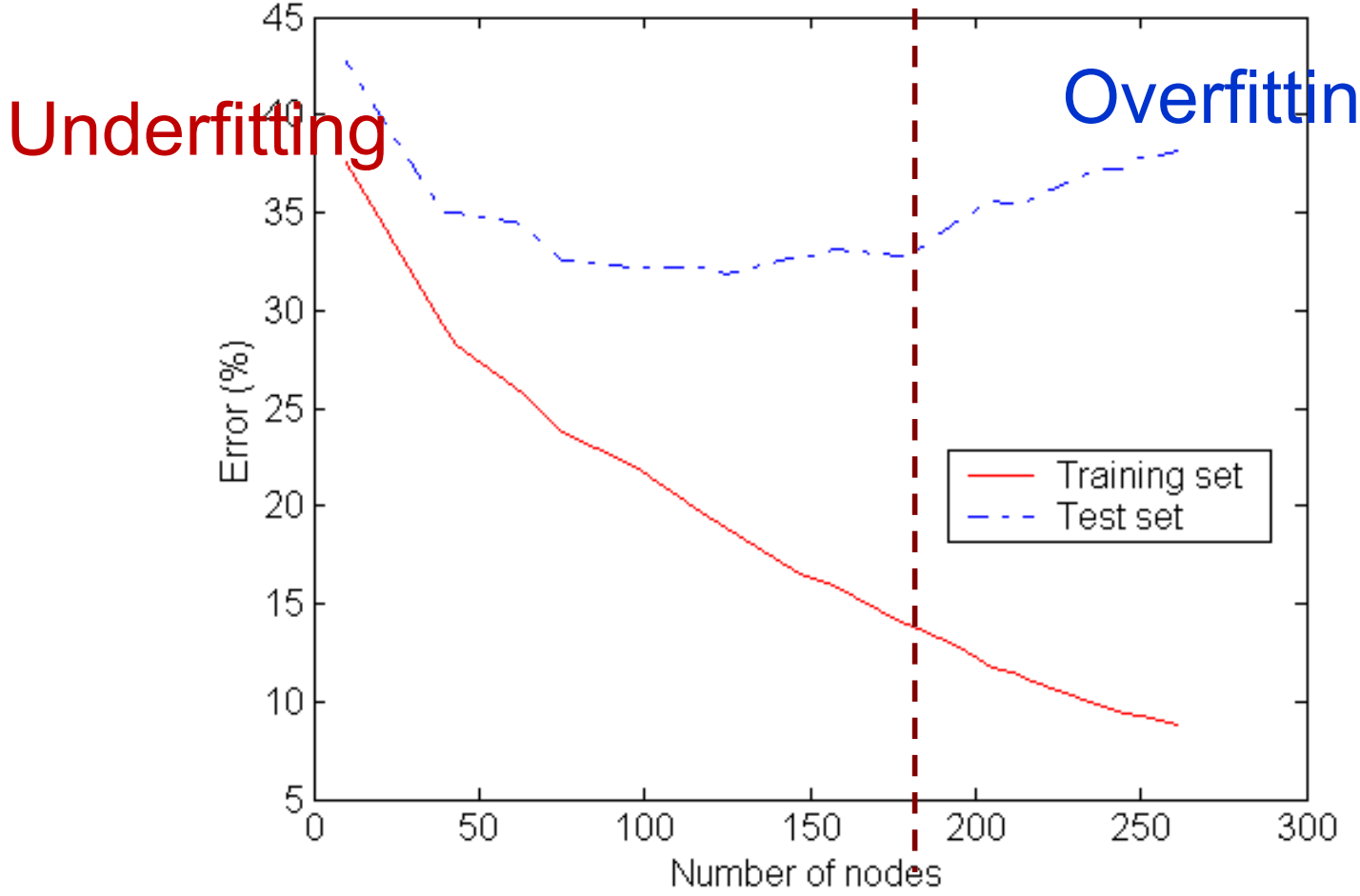
Triangular points:

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

or

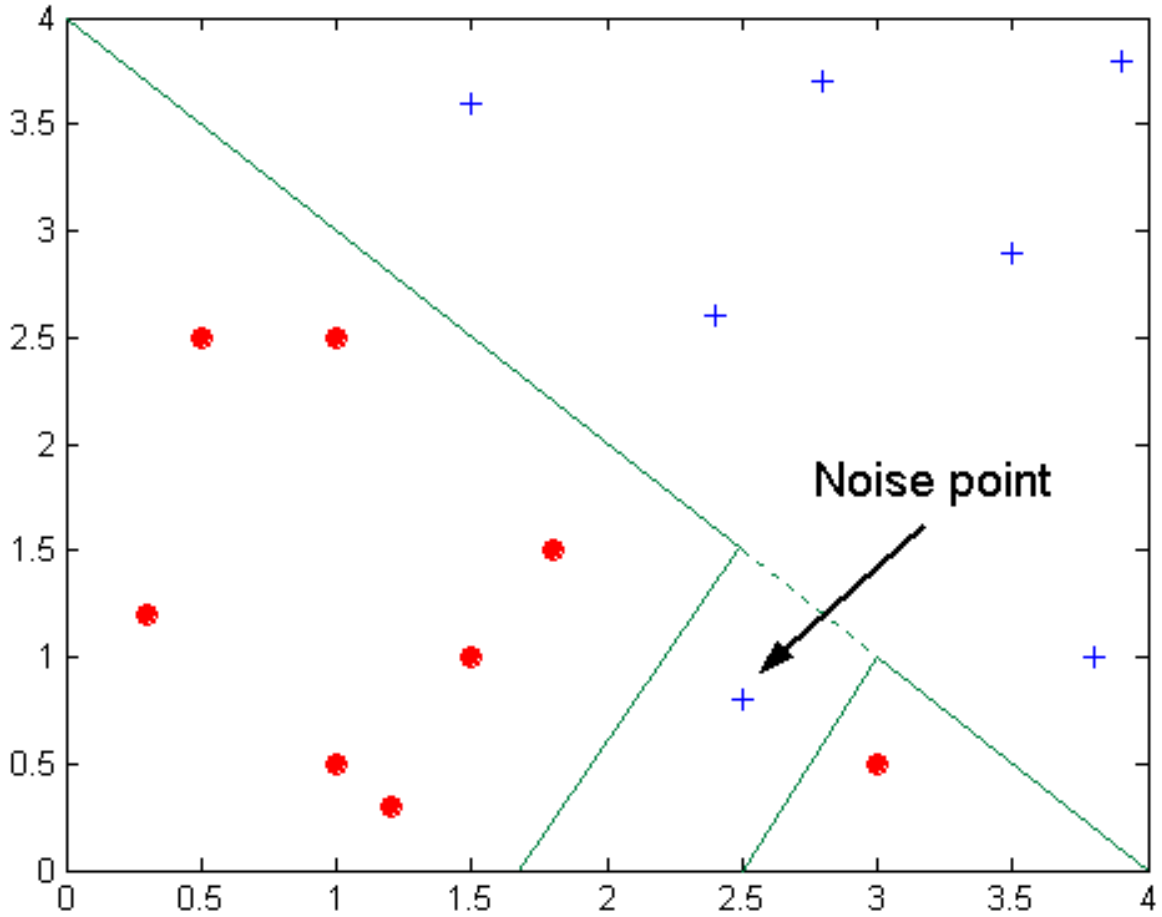
$\text{sqrt}(x_1^2+x_2^2) < 1$

Overfitting and Underfitting



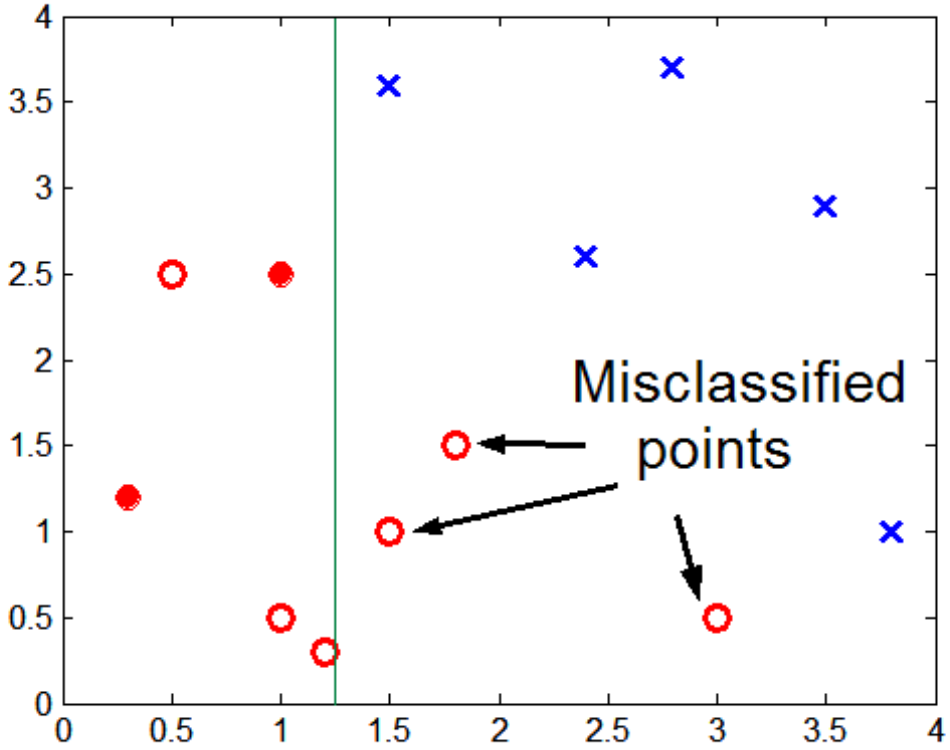
Underfitting: when model is too simple (not capturing the data very well), both training and test errors are large

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 decision 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 or MDL) 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

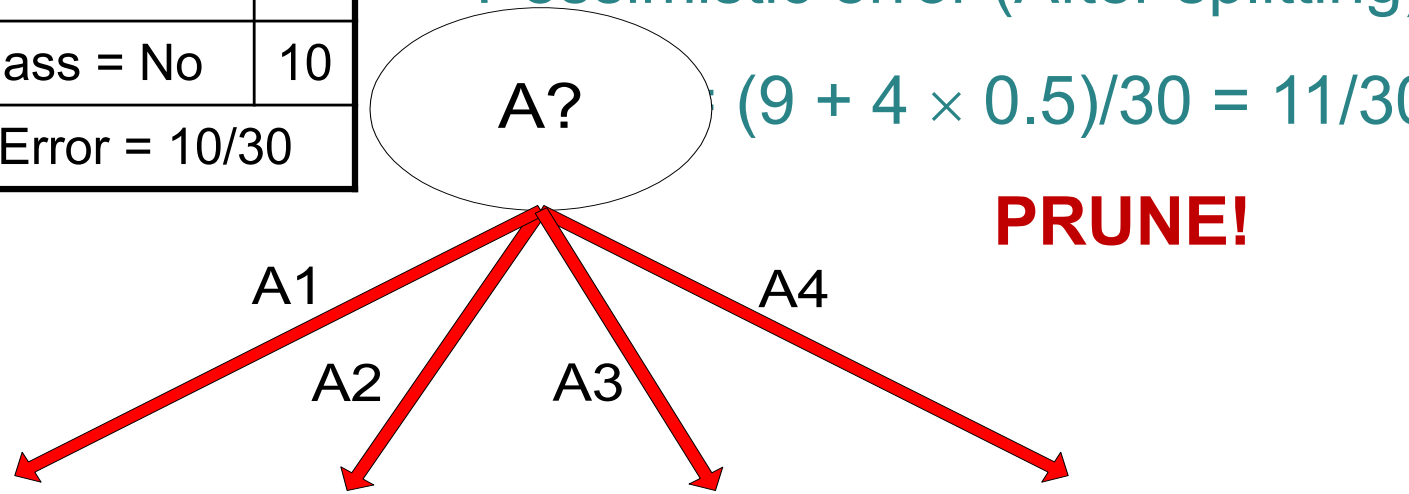
Training Error (After splitting) = 9/30

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

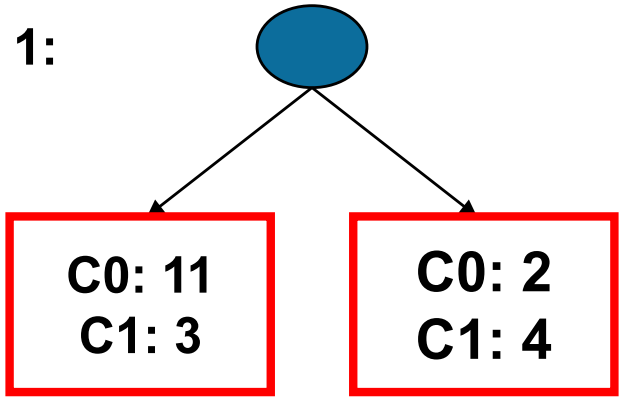
Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

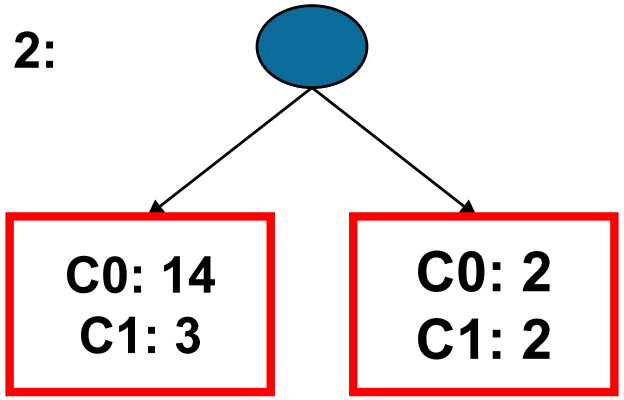
Examples of Post-pruning

- 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

Case 1:



Case 2:



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

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

Computing Cost 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

Accuracy = 80%

Cost = 3910

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

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

$N = a + b + c + d$

$Accuracy = (a + d)/N$

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

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

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

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

Cost-Sensitive 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 for Performance Evaluation

- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

- *Performance is application data dependent!*

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
  - oversampling vs undersampling
- 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

Rule-Based Classifier

- Classify records by using a collection of "if…then…" rules
- Rule: (*Condition*) $\rightarrow$ *y*
  - where
    - *Condition* is a conjunctions of attributes
    - *y* is the class label
  - *LHS*: rule antecedent or *condition*
  - *RHS: rule consequent*
  - Examples of classification rules:
    - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Tax Evasion=No

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Application of Rule-Based Classifier

- A rule ***r*** *covers* an instance **x** if the attributes of the instance satisfy the condition of the rule
  - Also say instance **x** *triggers* the rule ***r***.

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

Rule Coverage and Accuracy

- *Coverage* of a rule:
  - Fraction of records that satisfy the antecedent of a rule
- *Accuracy* of a rule:
  - Fraction of records that satisfy both the antecedent and consequent of a rule

Tid	Refund	Marital Status	Taxable Income	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**

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal (works)

A turtle triggers both R4 and R5 (contradictory, need to be resolved)

A dogfish shark triggers none of the rules (doesn't work)

Characteristics of Rule-Based Classifier

Ideally, a rule-based classifier has the following:

- *Mutually exclusive* rules
  - Every record is covered by at most one rule
  - If not, a record may trigger more than one rule
  - Solution?
    - Ordered rule set or Voting Schemes
- *Exhaustive* rules
  - Each record is covered by at least one rule
  - If not, a record may not trigger any rules
  - Solution?
    - Use a default class

Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules is triggered, it is assigned to the default class

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Rule Ordering Schemes

- Rule-based ordering
  - Individual rules are ranked based on certain rule *quality measure*
- Class-based ordering
  - Rules that belong to the *same class* appear together

Rule-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Class-based Ordering

No

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Married}) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

Building Classification Rules

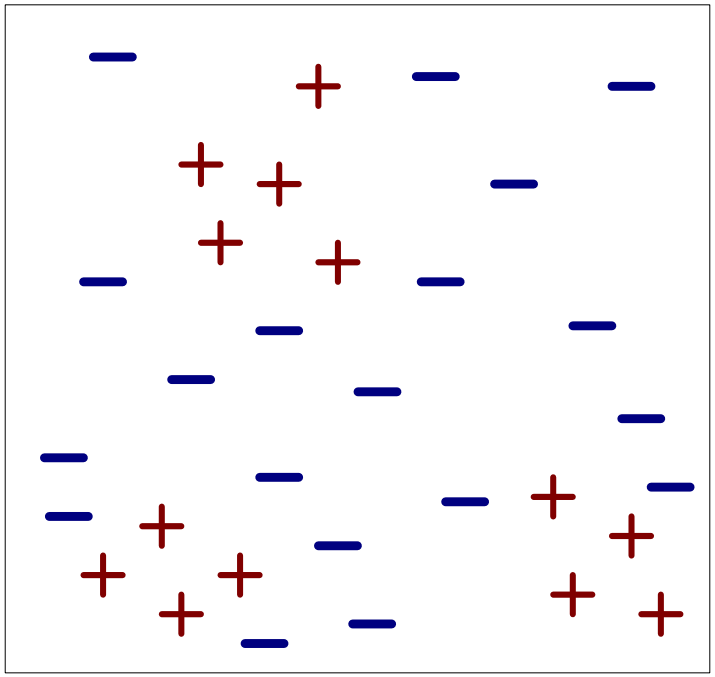
- Direct Method:
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte's 1R
- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - e.g: C4.5rules

Direct Method: Sequential Covering

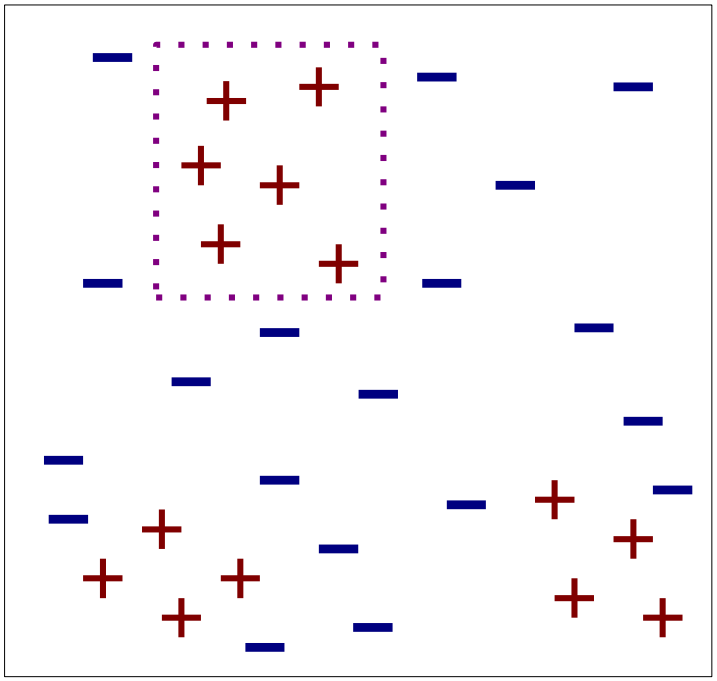
For each class (one class at a time)

1. Start from an empty rule
2. Grow a rule in a greedy fashion based on certain evaluation measure
   - Find a rule that covers many positive examples and none/few negative examples
   - Expensive due to exponential size of search space
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Example of Sequential Covering

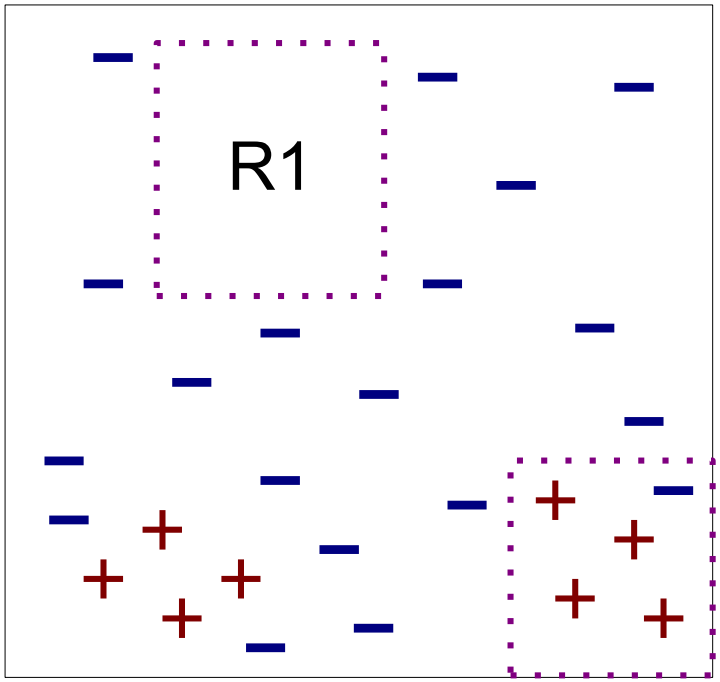


(i) Original Data

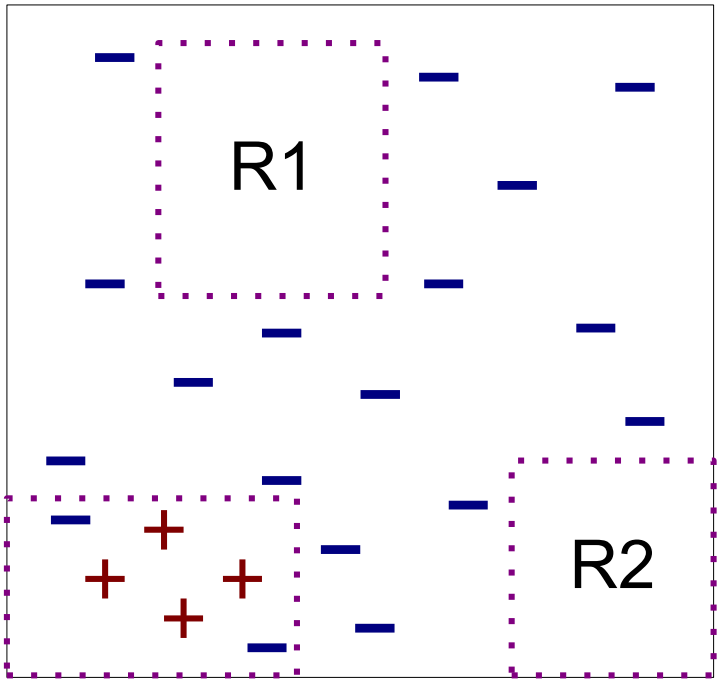


(ii) Step 1

Example of Sequential Covering…



(iii) Step 2



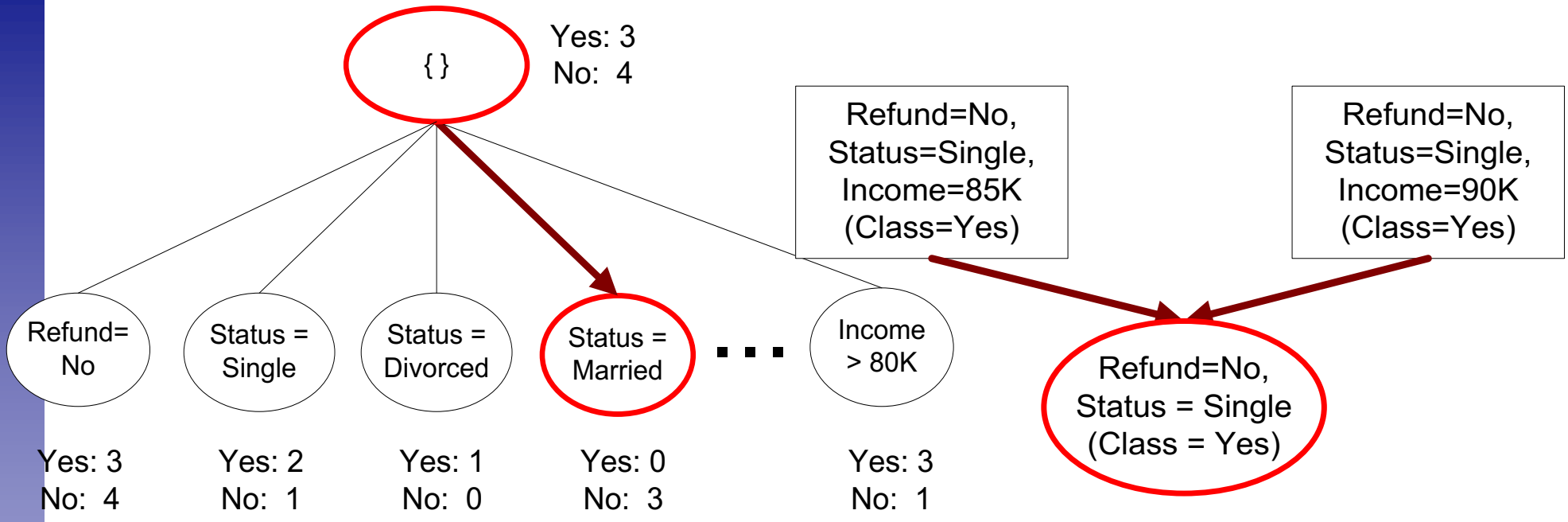
(iv) Step 3

Aspects of Sequential Covering

- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



Start with an empty rule for the targeted class. Sequentially add new conjunct to improve the rule quality.

Start with a positive example for the targeted class. Remove conjuncts to cover more positive examples.

Rule Growing (Examples)

- CN2 Algorithm:
  - Start from an empty conjunct: {}
  - Add conjuncts that minimizes the entropy measure: {A}, {A,B}, …
  - Determine the rule consequent by taking majority class of instances covered by the rule
- RIPPER Algorithm:
  - Start from an empty rule: {} => class
  - Add conjuncts that maximizes FOIL's information gain
    - R0: {} => class (initial rule)
    - R1: {A} => class (rule after adding conjunct)
    - Gain(R0, R1) = t [log (p1/(p1+n1)) – log (p0/(p0 + n0))]
    - where t: # of positive instances covered by both R0 and R1

      p0: number of positive instances covered by R0

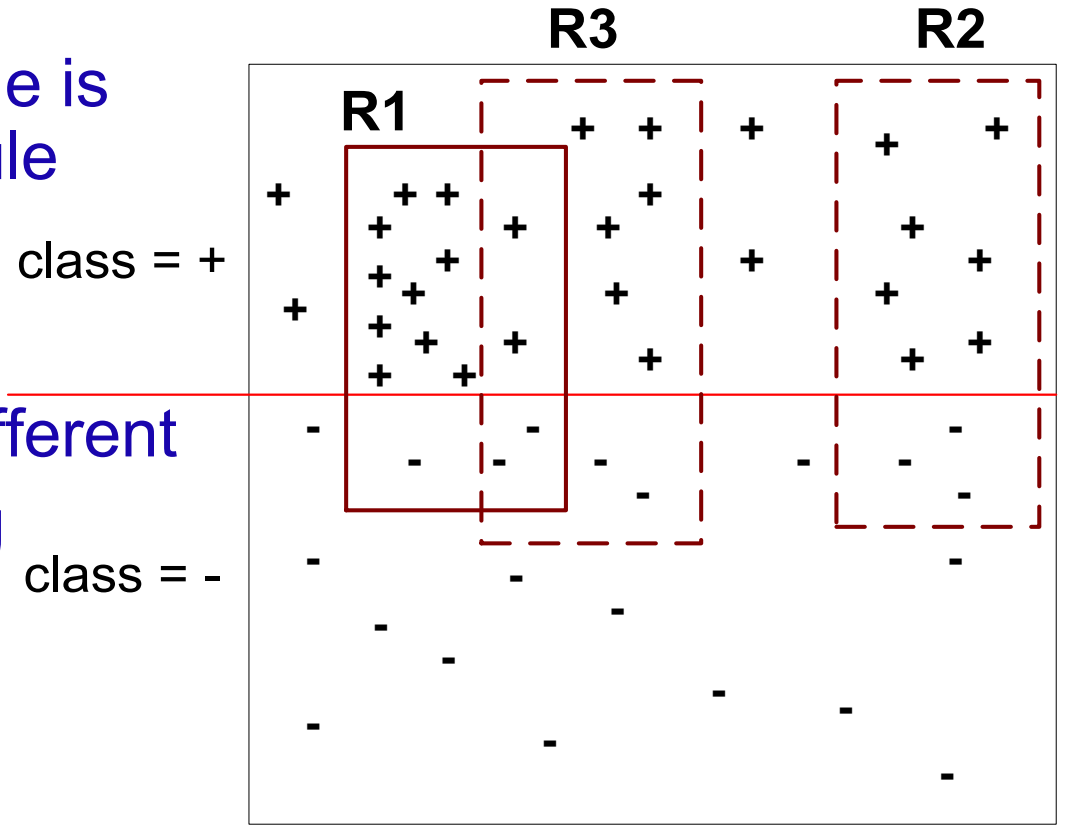
      n0: number of negative instances covered by R0

      p1: number of positive instances covered by R1

      n1: number of negative instances covered by R1

Instance Elimination

- Why do we need to eliminate instances?
  - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
  - Ensure next rule be different
  - Prevent overestmating accuracy of rules
- Why do we remove negative instances?
  - Prevent underestimating accuracy of rules
  - Compare rules R2 and R3 in the diagram



Rule Evaluation

- Metrics:
  - Accuracy $= \frac{n_c}{n}$
  - Laplace $= \frac{n_c + 1}{n + k}$
  - M-estimate $= \frac{n_c + kp}{n + k}$

n : Number of instances covered by rule

n_c : Number of class c instances covered by rule

k : Number of classes

p : Prior probability

Stopping Criterion and Rule Pruning

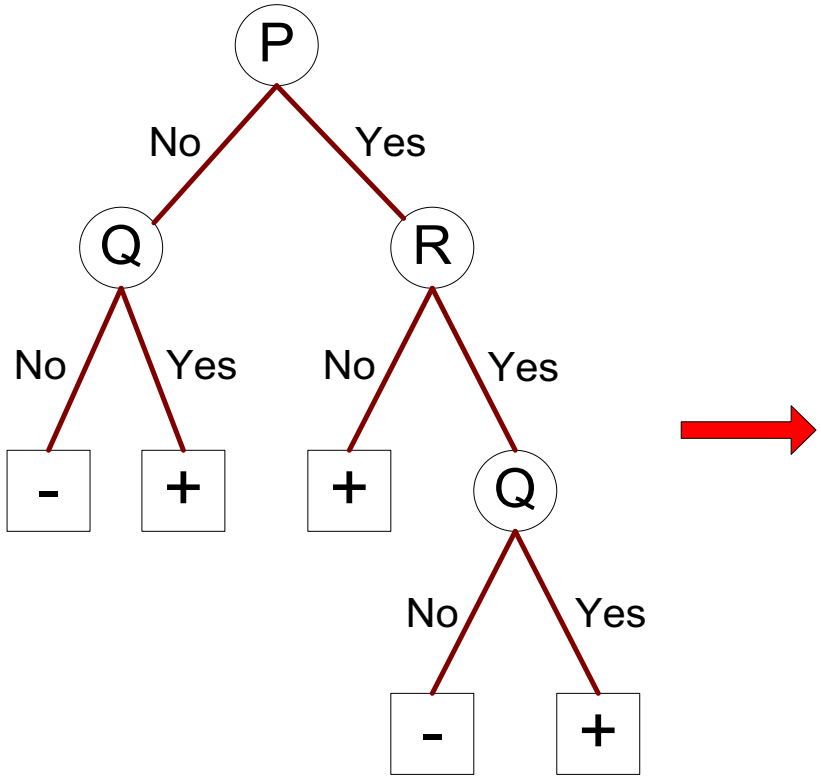
- Stopping criterion
  - Compute the gain
  - If gain is not significant, discard the new rule

- Rule Pruning
  - Similar to post-pruning of decision trees
  - Reduced Error Pruning:
    - Remove one of the conjuncts in the rule
    - Compare error rate on validation set before and after pruning
    - If error improves, prune the conjunct

Summary of Direct Method

- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

Indirect Methods (Decision Tree)



Rule Set

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

Advantages of Rule-Based Classifiers

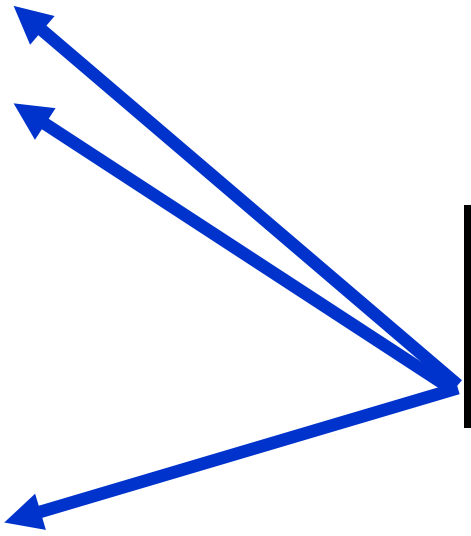
- As highly expressive as decision trees
- Easy to interpret (descriptive)
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

Instance-Based Classifiers

- Store the training records (instances)
- Use training records to predict the class label of unseen cases

Set of Stored Cases

Atr1	……...	AtrN	Class
			A
			B
			B
			C
			A
			C
			B



Unseen Case

Atr1	……...	AtrN

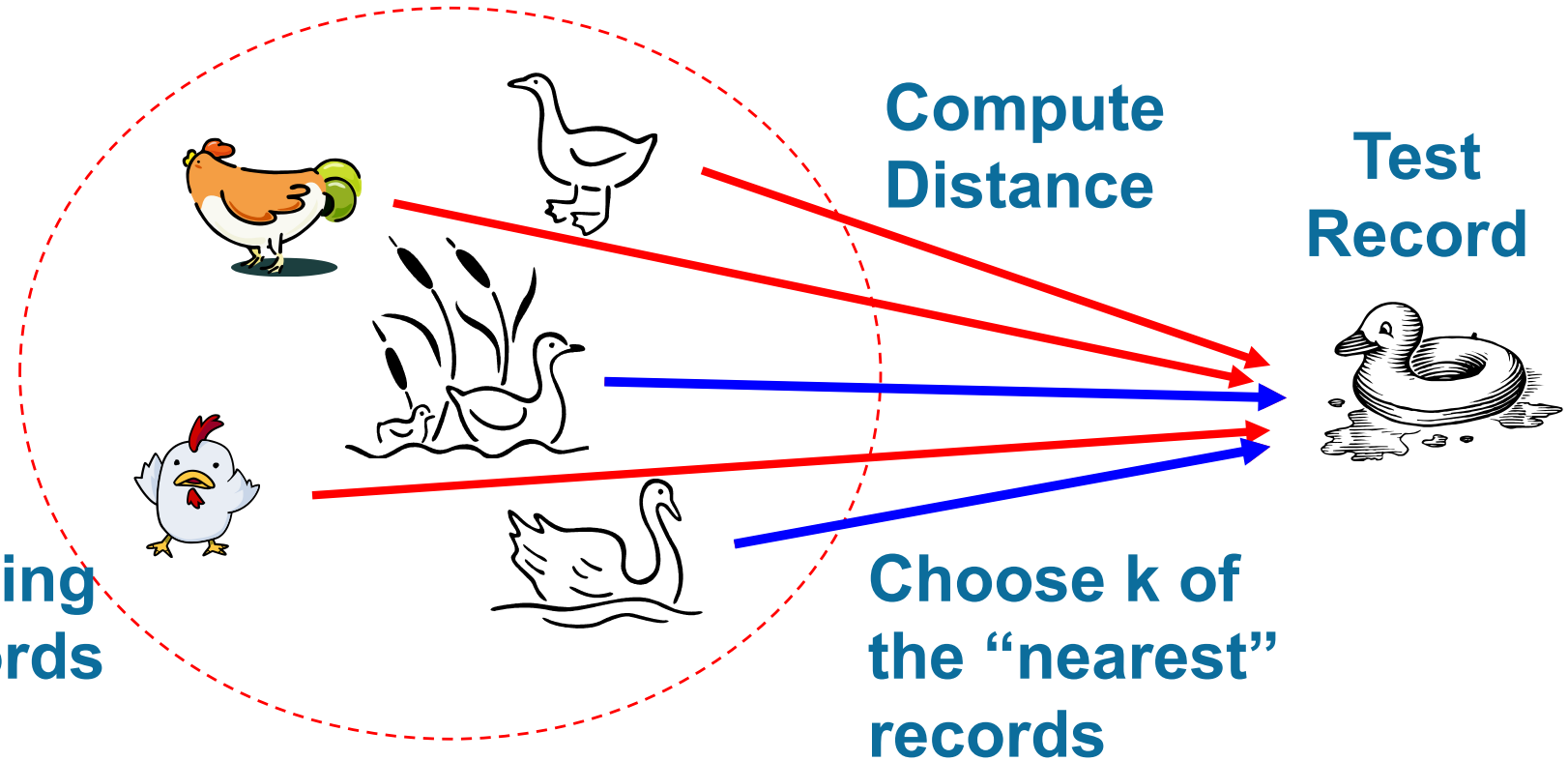
Instance Based Classifiers

- Rote-learner: Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

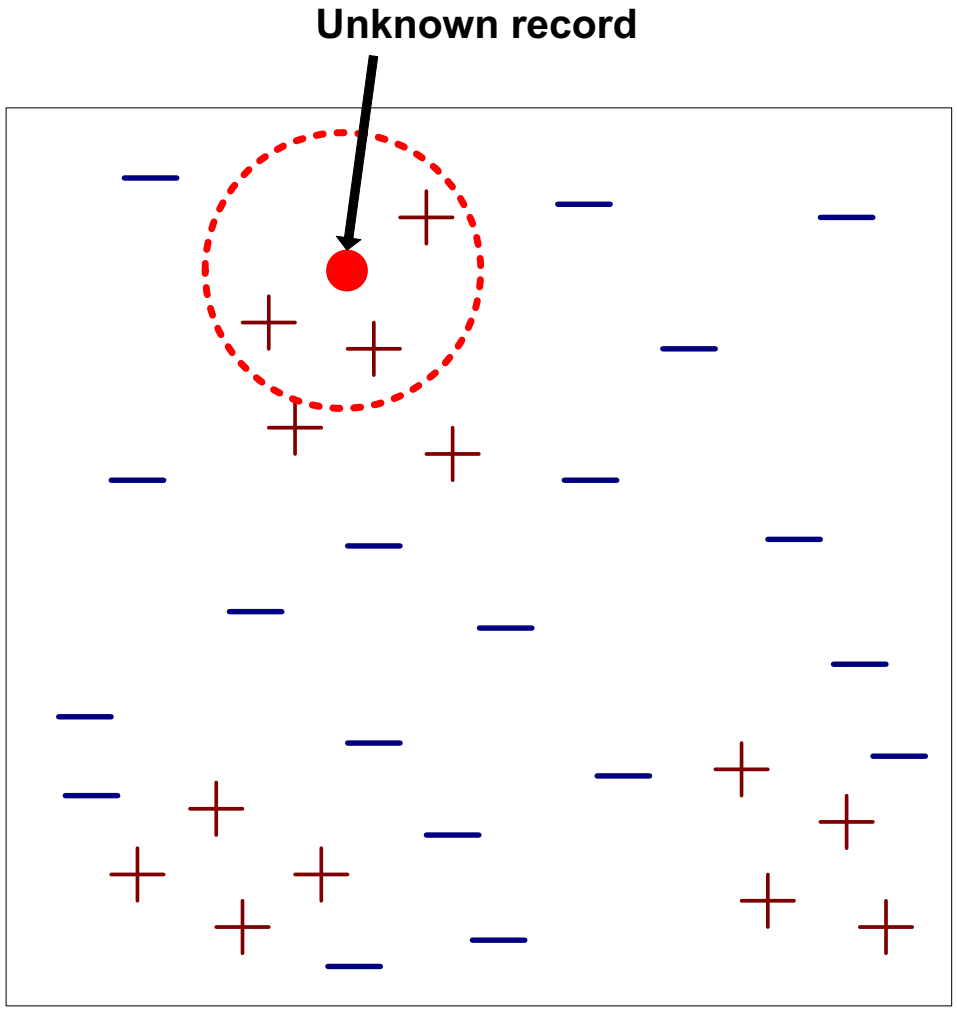
- Nearest neighbor: Uses k closest/similar points (nearest neighbors) to perform classification

Nearest Neighbor Classifiers

- Basic idea:
  - *If it walks like a duck, quacks like a duck, then it's probably a duck*

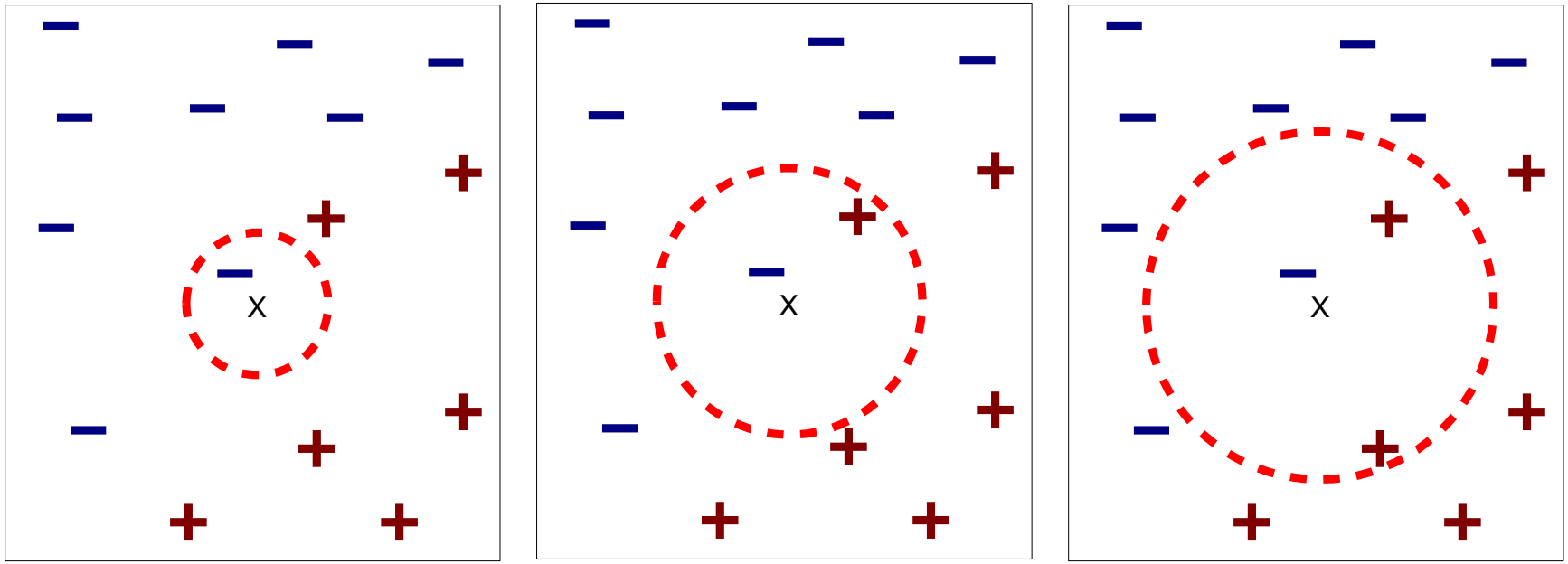


Nearest-Neighbor Classifiers



- Requirements:
  - The set of stored records
  - Distance Metric
  - K, the number of NN
- For an unknown record:
  - Compute distance to other training records
  - Identify k NNs
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

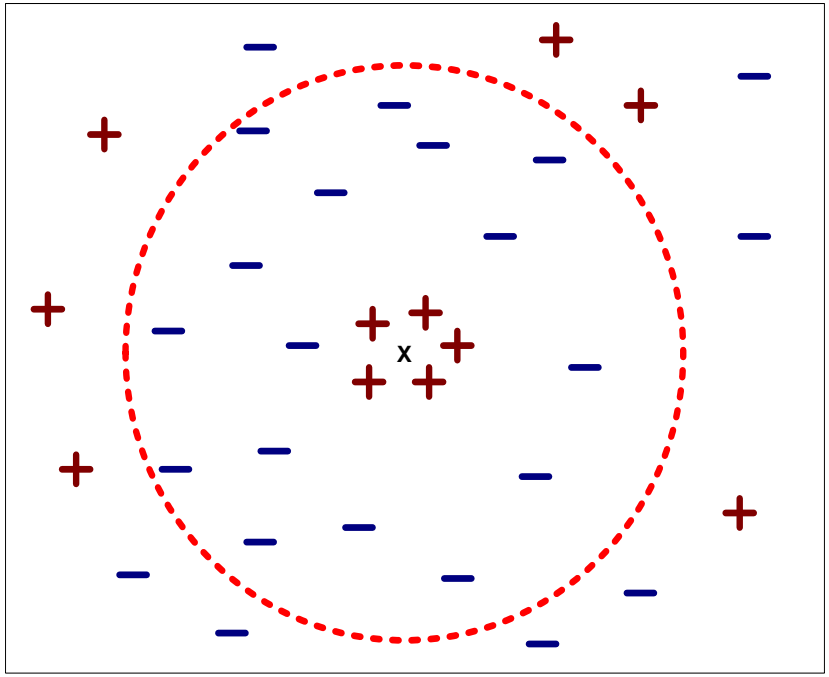


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification…

- Scaling (度量) issues: attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Example:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from $10K to $1M

Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0		1 0 0 0 0 0 0 0 0 0 0 0	
0 1 1 1 1 1 1 1 1 1 1 1	vs	0 0 0 0 0 0 0 0 0 0 0 1	
d = 1.4142		d = 1.4142	

Solution: Normalize the vectors to unit length

Nearest neighbor Classification…

difference and similar between these techs: consider the attri space,

- k-NN classifiers are lazy learners
  - Unlike eager learners such as decision tree induction and rule-based systems
  - It does not build models explicitly, i.e., no training necessary. (Low training cost)
  - Classifying unknown records are relatively expensive. (High classification cost)

Bayes Classifier

- Sometimes the relationship between attributes and the class variable is not deterministic!
- A *probabilistic* framework for solving classification problems
- Conditional Probability: $P(C \mid A) = \frac{P(A,C)}{P(A)}$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given: a record with attributes (A_1, A_2,…,A_n)

  Predict: class C

  - Specifically, *we want to find the value of C that maximizes P(C| A_1, A_2,…,A_n)*

- Can we estimate P(C| A_1, A_2,…,A_n) directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$

- *How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?*

Naïve Bayes Classifier

- Assume *conditional independence* among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C_j) = \Pi\, P(A_i \mid C_j)$
    $= P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
  - A previously unseen data object is classified as C_j if $P(C_j)\, \Pi\, P(A_i \mid C_j)$ is maximal.

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10, P(Yes) = 3/10
- For *discrete* attributes:

  $$P(A_i \mid C_k) = |A_{ik}|/ N_{Ck}$$

  where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - e.g., P(Refund=Yes|Yes)=0

    P(Status=Married|No) = 4/7

How to Estimate Probabilities from Data?

- For *continuous* attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: (A < v) or (A > v)
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a *normal* distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair
- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

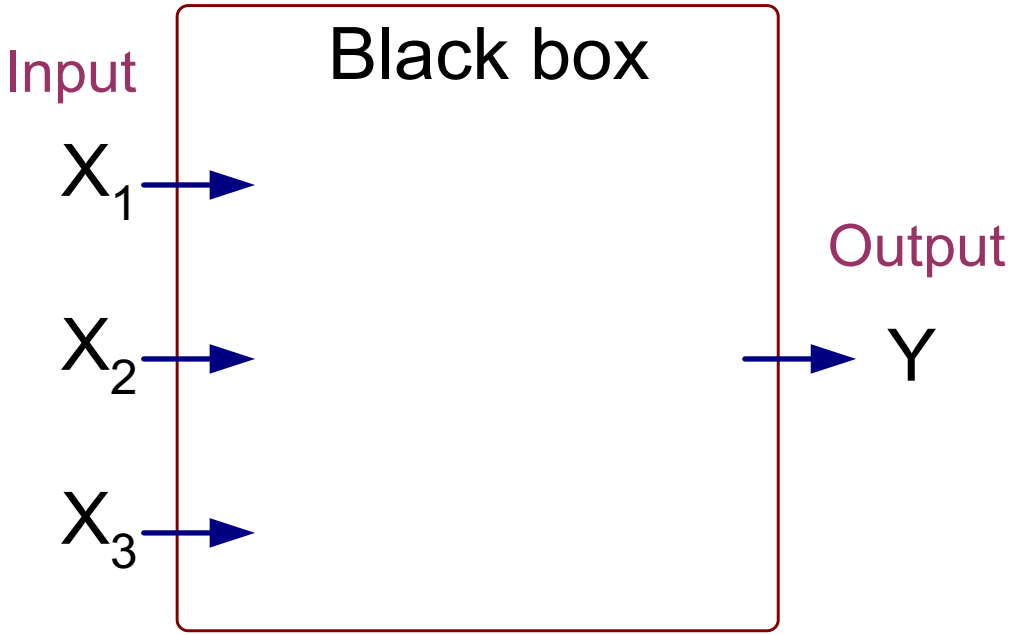
Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

Artificial Neural Networks (ANN)

- Hot topic! Used for deep learning.
- Perceptron: a simple neural network architecture; served as a binary classifier.

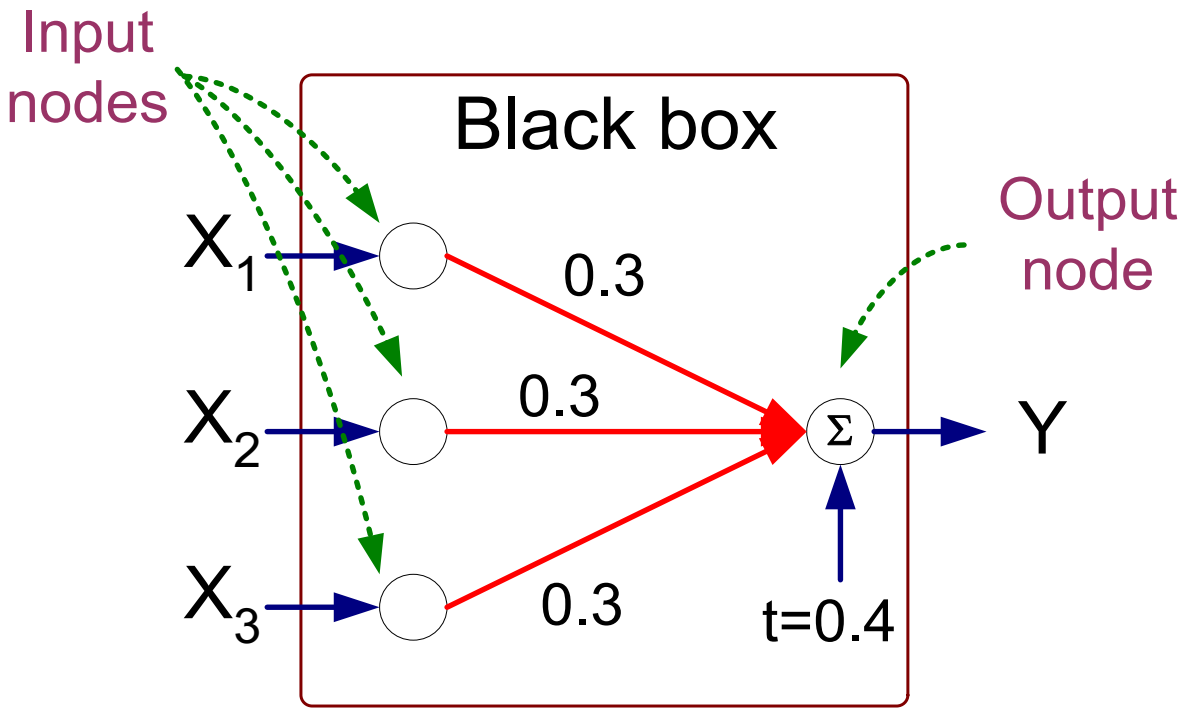
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Perceptron

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

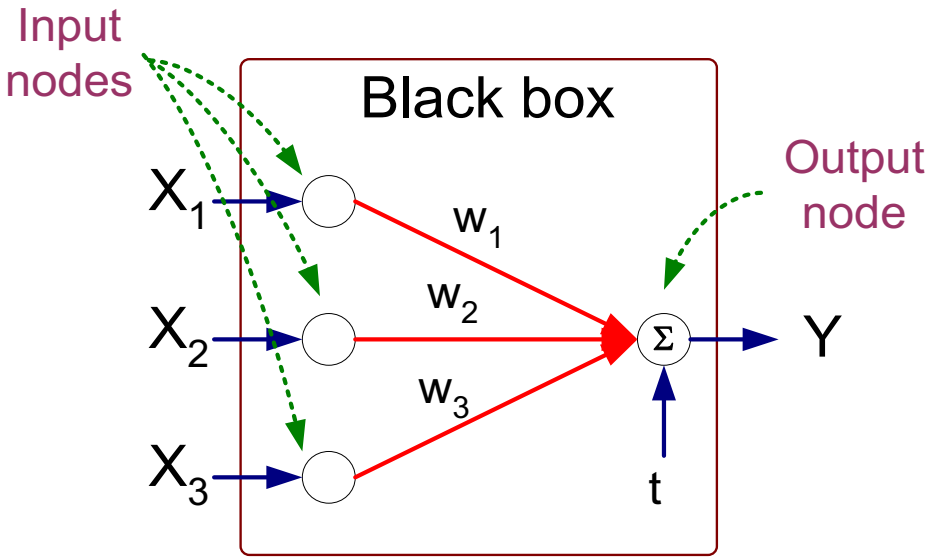


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Perceptron

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold *t*

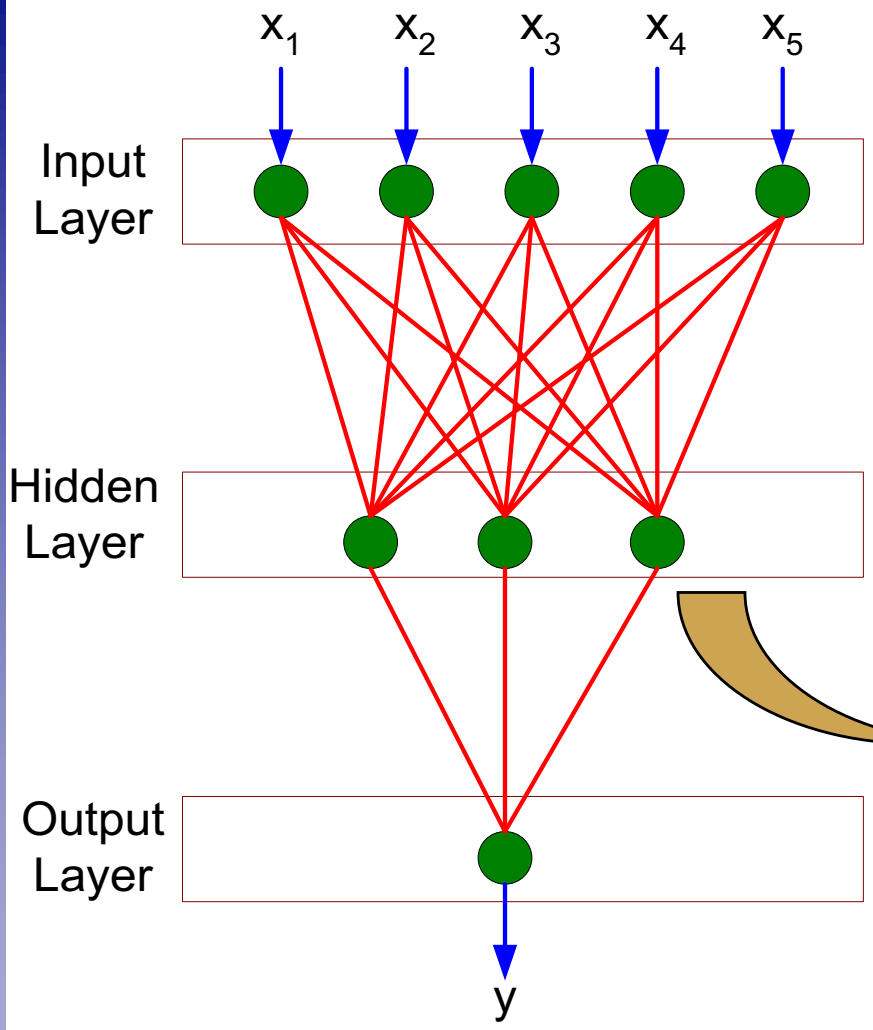


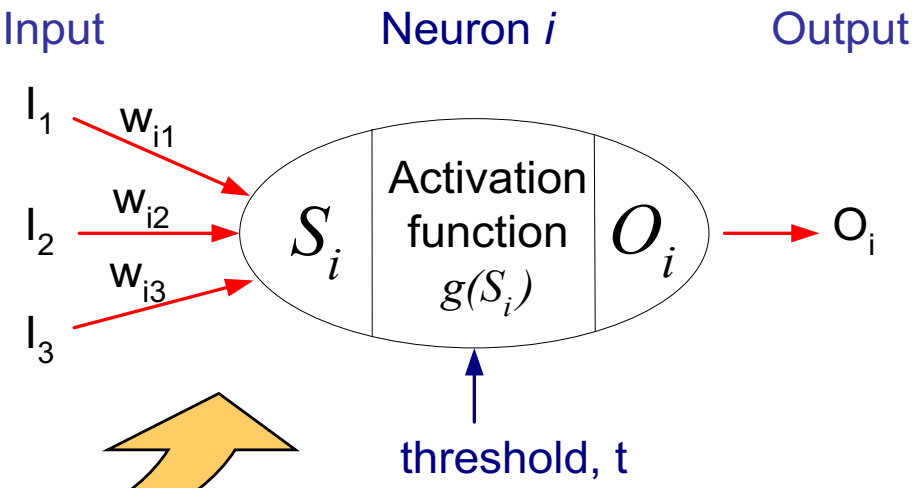
Perceptron Model

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

General Structure of ANN





Training ANN means learning the weights of the neurons

Algorithm for learning ANN

- Initialize the weights (w_0, w_1, …, w_k)
- Adjust the weights such that the output is consistent with class labels of training examples
- Objective function:

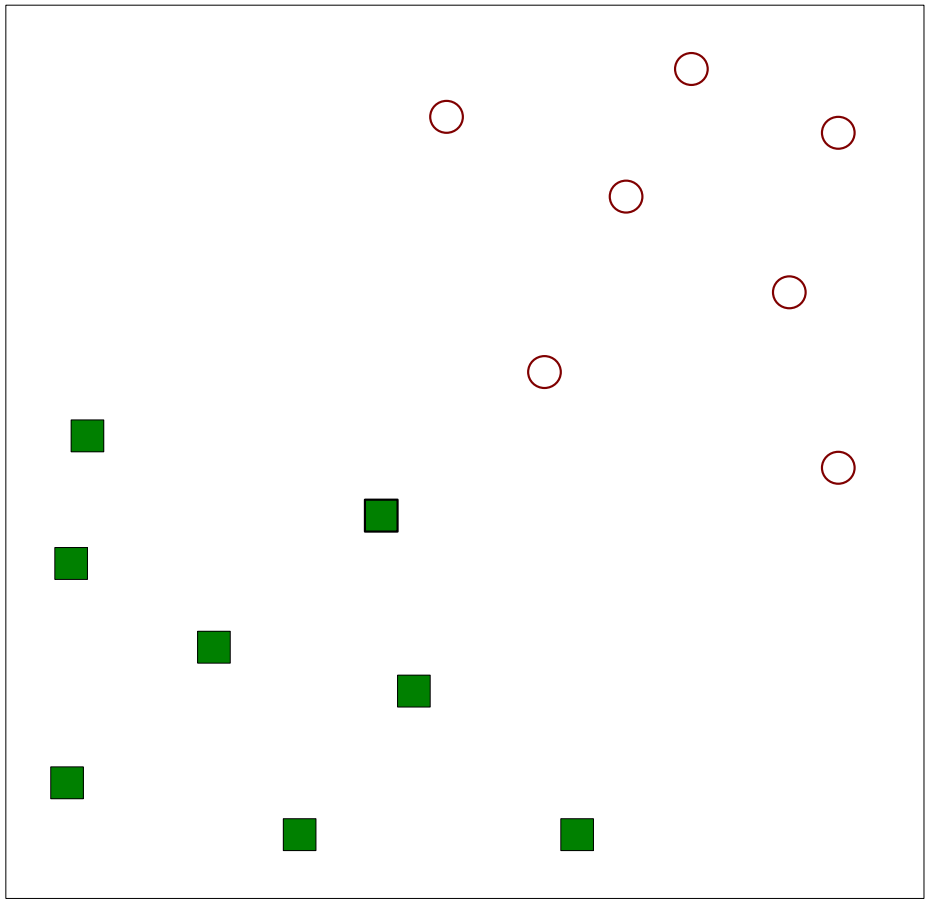
$$E = \sum_i \left[Y_i - f(w_i, X_i) \right]^2$$

- Find the weights w_i's that minimize the above objective function
  - e.g., via *backpropagation* algorithm

Support Vector Machines

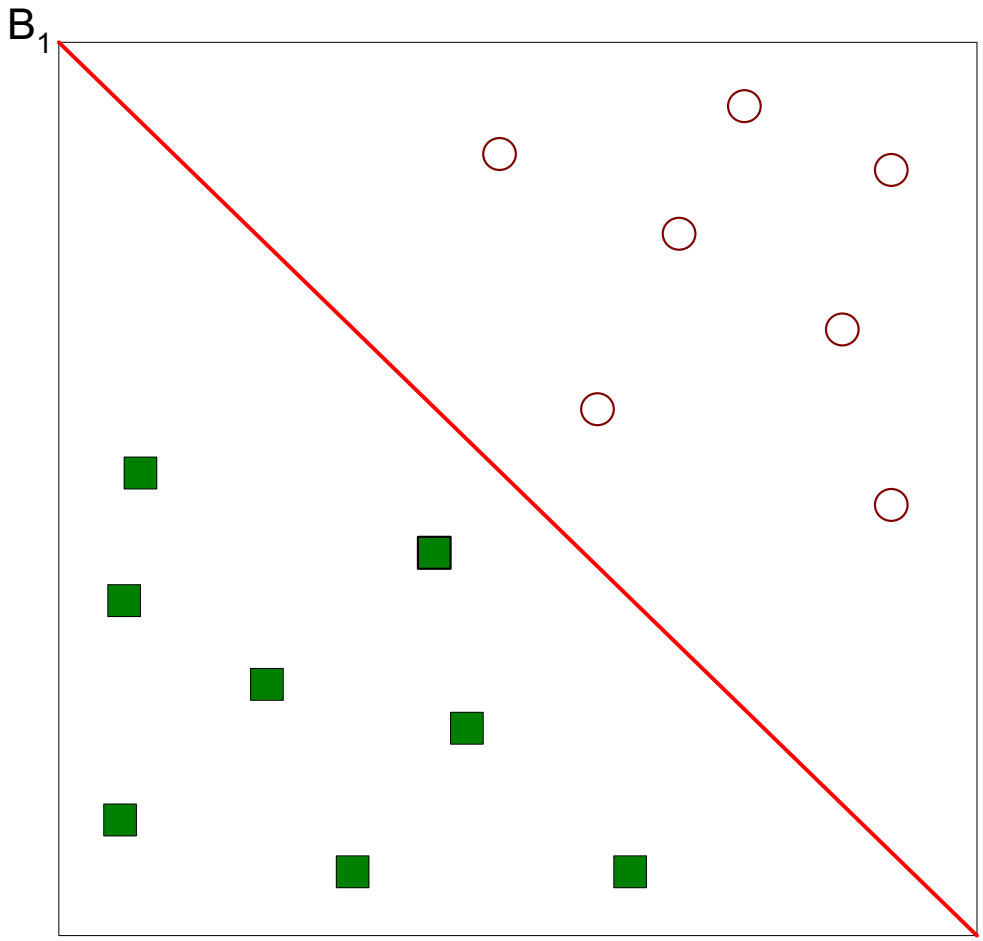
- Rooted in statistical learning theory
- Works very well with high-dimensional data
- Representing the decision boundary by *support vectors* derived from a subset of training examples
- Use the maximal margin hyperplane to linearly separate the data objects of different classes

Support Vector Machines



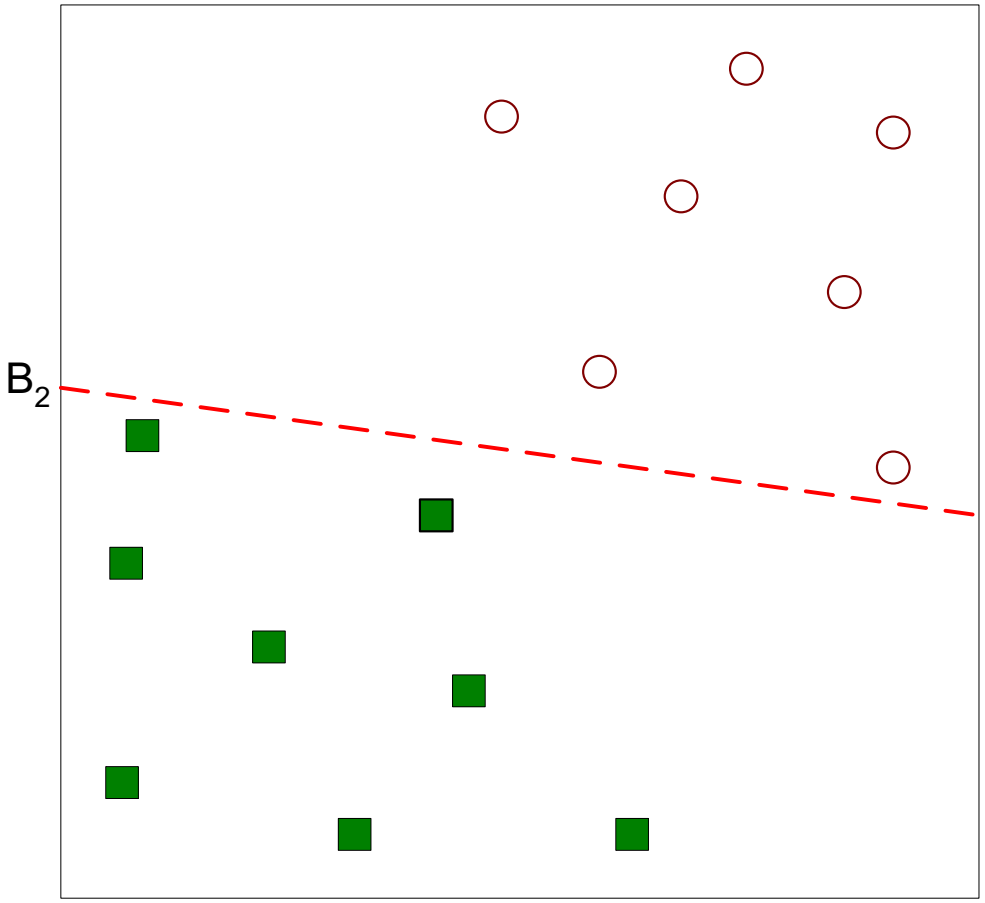
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



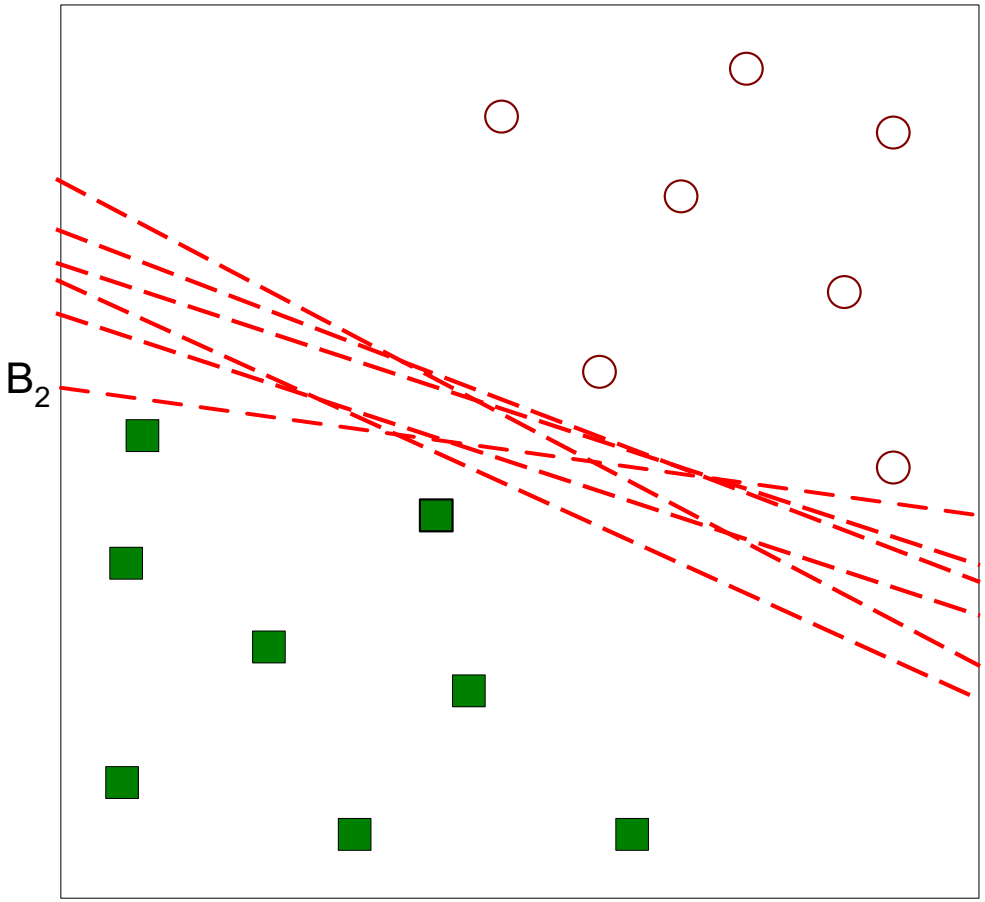
- One Possible Solution

Support Vector Machines



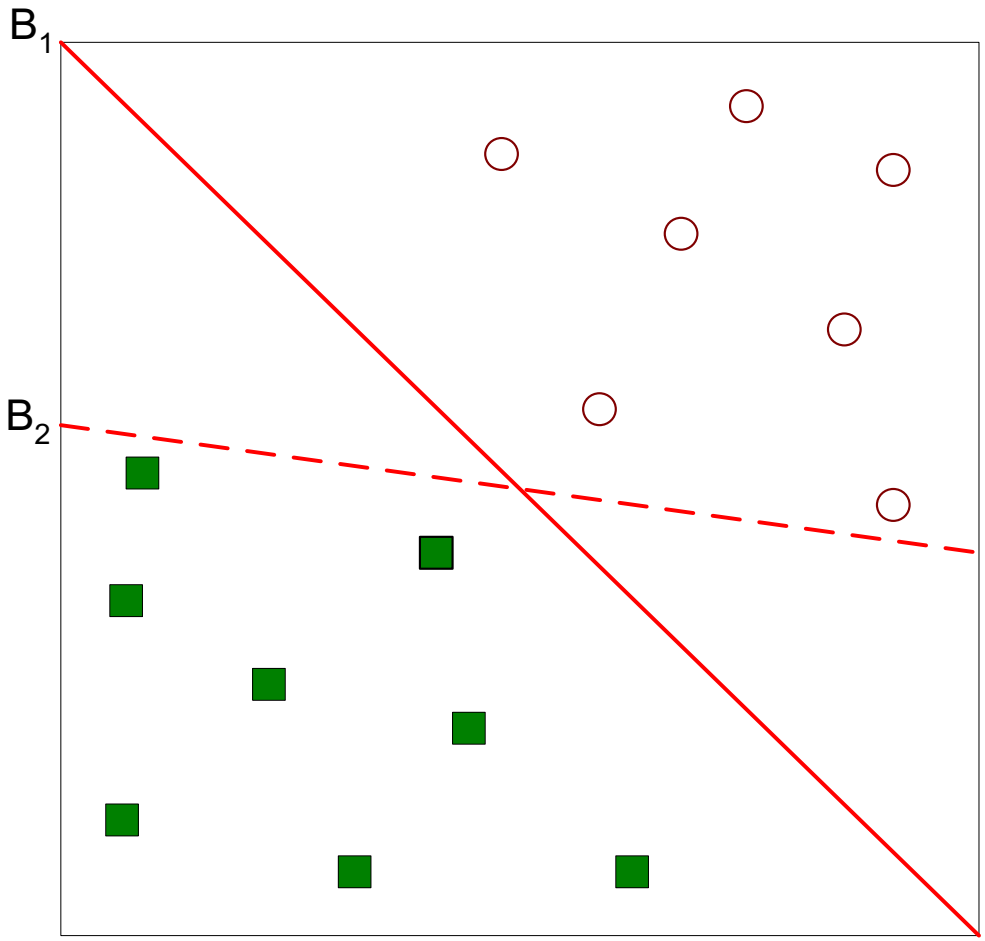
- Another possible solution

Support Vector Machines



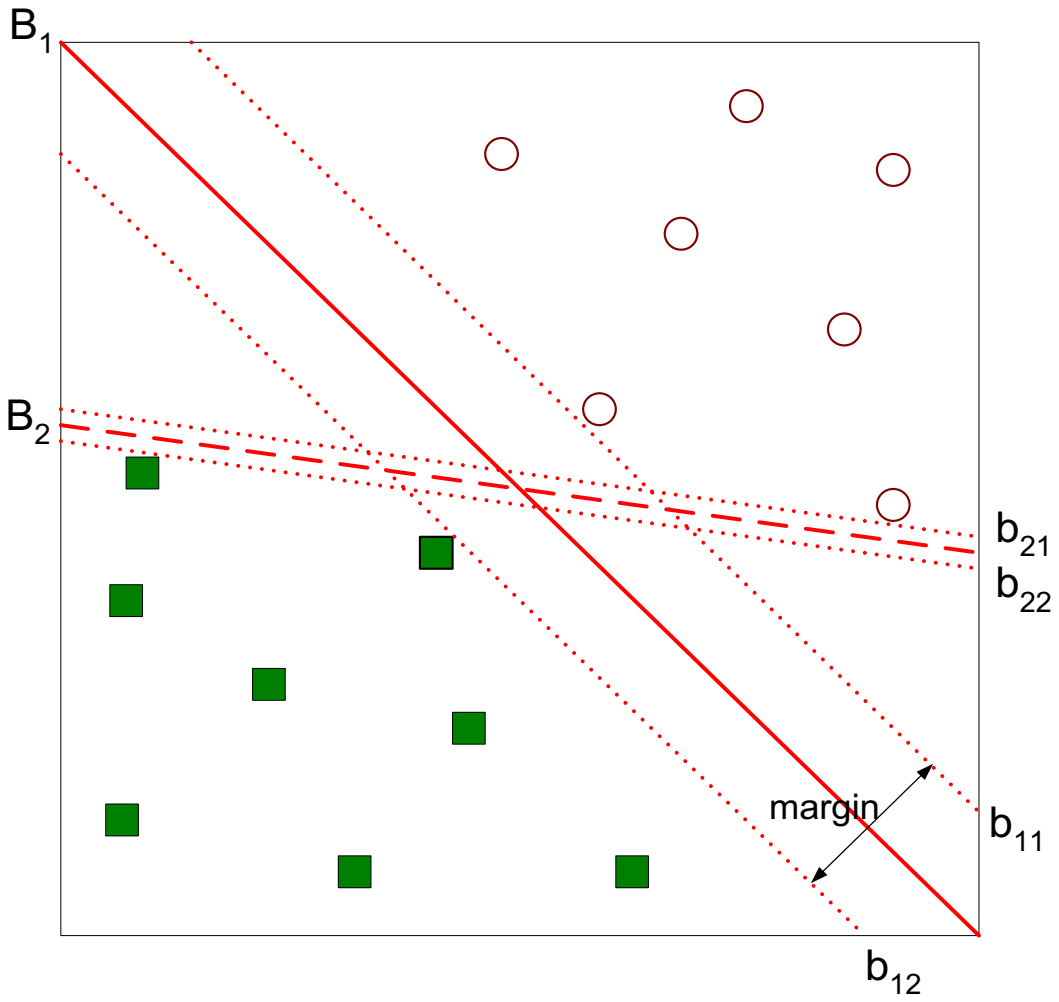
- Other possible solutions

Support Vector Machines



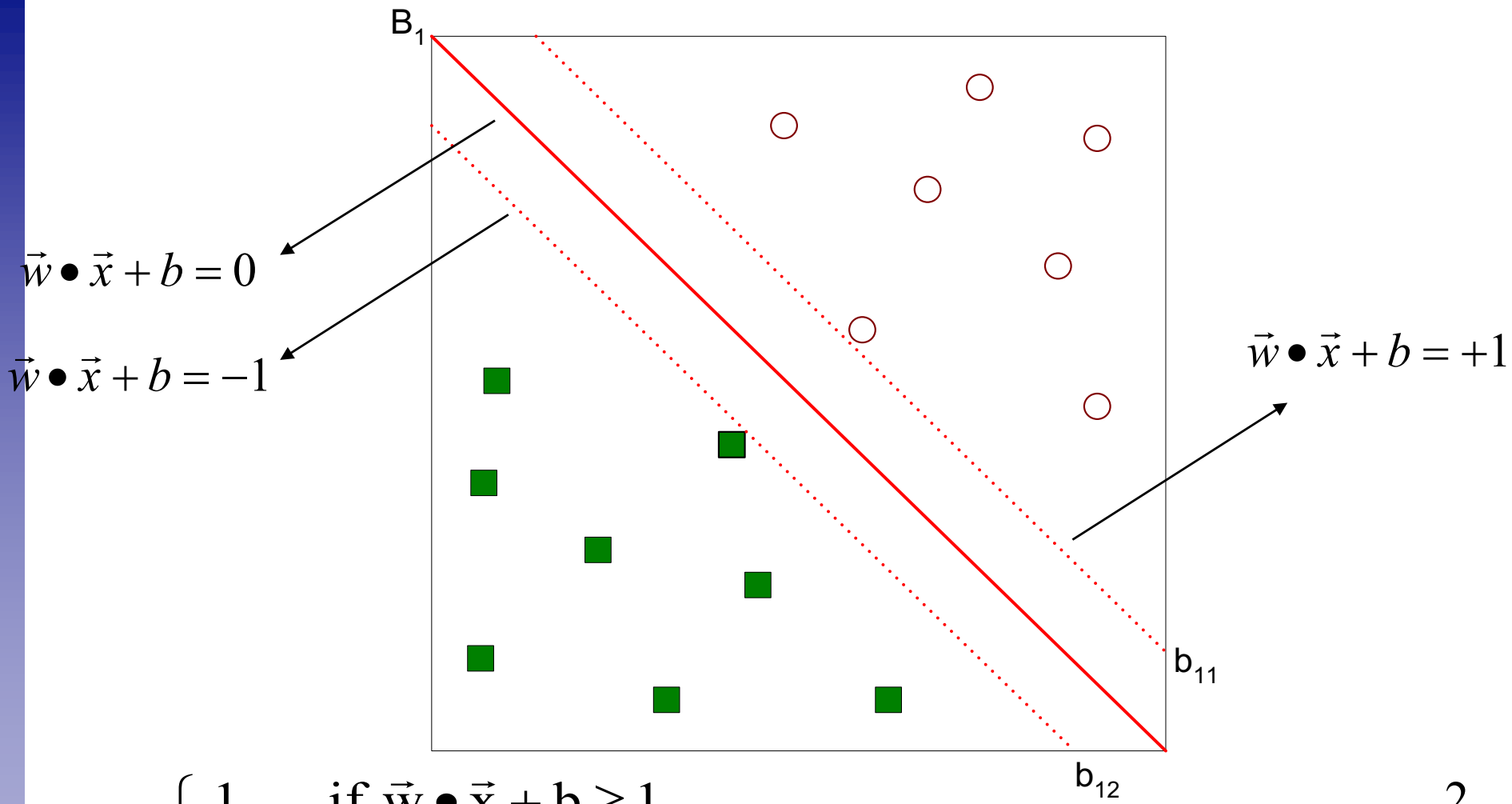
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|}$$

Support Vector Machines

- We want to maximize: $\text{Margin} = \frac{2}{\| \vec{w} \|^2}$
  - Which is equivalent to minimizing: $L(w) = \frac{\| \vec{w} \|^2}{2}$
  - But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

- This is a constrained optimization problem
  - Numerical approaches to solve it (e.g., quadratic programming)