Classification: Basic Concepts

Cam Tu Nguyen

阮锦绣

Software Institute, Nanjing University
nguyenct@lamda.nju.edu.cn
ncamtu@gmail.com

Outline

- Basic Concepts
  - Classification: Definition
  - Examples of classification tasks
  - General Approach to Classification
- Basic Classification Techniques
  - Decision Tree Induction
  - Naïve Bayesian Classification
- Classification Evaluation

Classification: Definition

- Given a collection of data objects (**training set**)
  - Each object contains a set of **attributes**, one of the attributes is the **class** (**categorical attribute**)

- Find a **model** for class attribute as a function of the values of other attributes.

- Goal: previously unseen objects should be assigned a class as accurately as possible.
  - For evaluation: a **test set** is used to determine the performance of the model. Different evaluation measures include **accuracy**, **recall**, **precision**, **AUC**, etc.
  - Usually, the given data set is divided into training and testing sets, where **training set** is used to build the model, and **test set** is used to validate it.

Examples of Classification Tasks

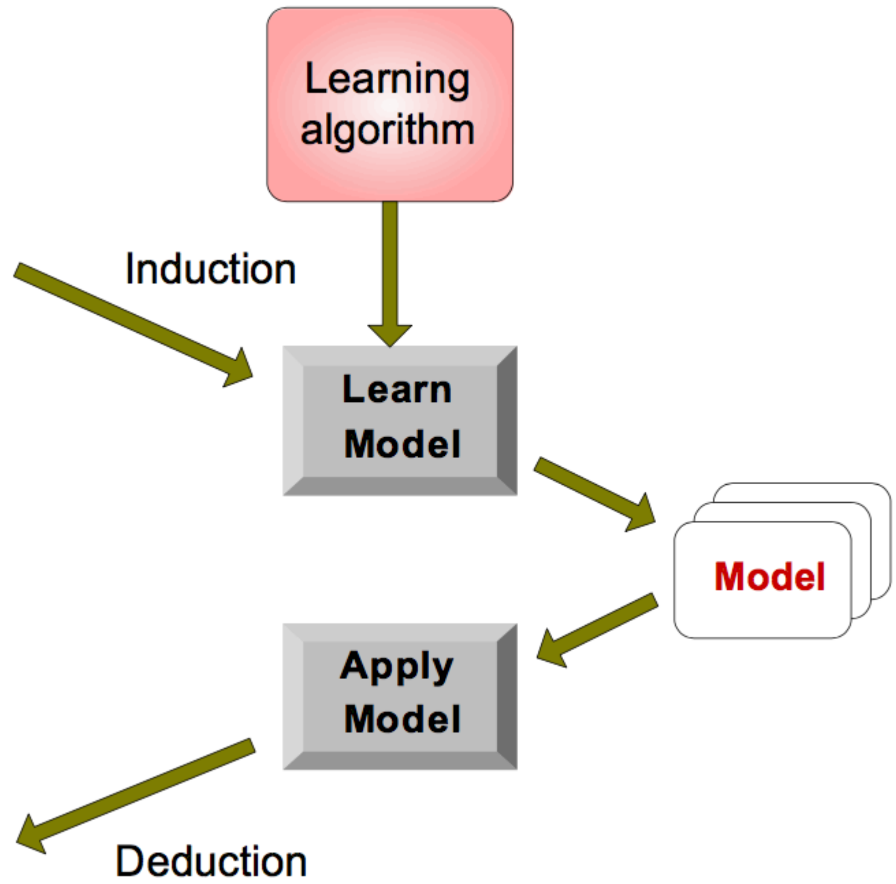
- Predicting tumor cells as **benign** or **malignant**
- Classifying credit card transactions as **legit** or **fraud**
- A marketing manager needs data analysis to help guess whether a customer with a given profile will **buy** a new computer (classes are buy/not buy).
- A bank loans officer need analysis of her data to learn which loan application is risky for her bank (classes are risky/not risky).
- Categorizing news stories as finance, weather, entertainment, sports, etc.

General Approach to Classification

- Classification task: two-step approach

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

General Approach to Classification

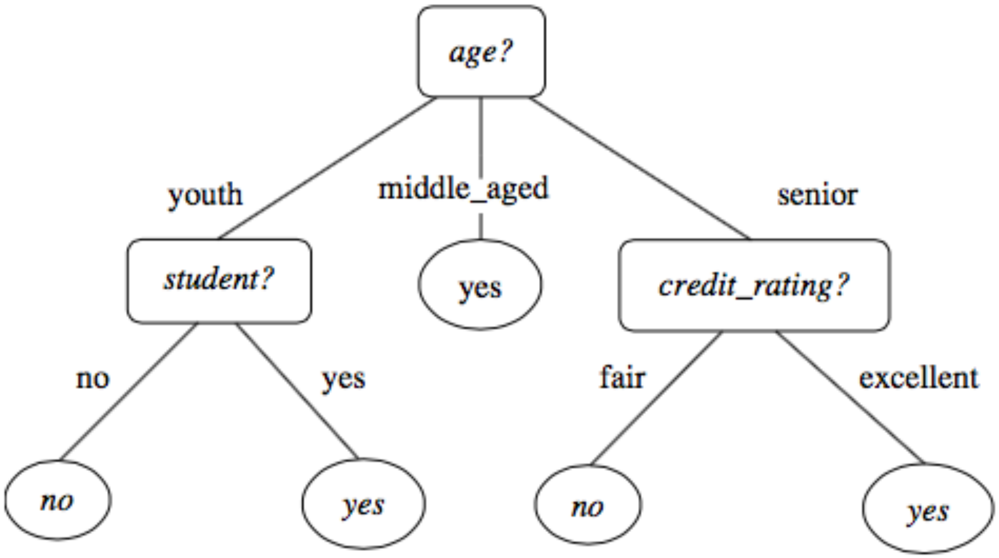
- Common Learning Algorithms: supervised learning methods:
  - Decision Trees
  - Naïve Bayes
  - Rule-based Methods
  - Neural Network
  - Support Vector Machine
  - Lazy Learners (K-nearest neighbors).
  - Ensemble Methods

Outline

- Basic Concepts
  - Classification: Definition
  - Examples of classification tasks
  - General Approach to Classification
- Basic Classification Techniques
  - Decision Tree Induction
  - Naïve Bayesian Classification
- Classification Evaluation
- Techniques to Improve Classification Accuracy

Decision Tree Induction

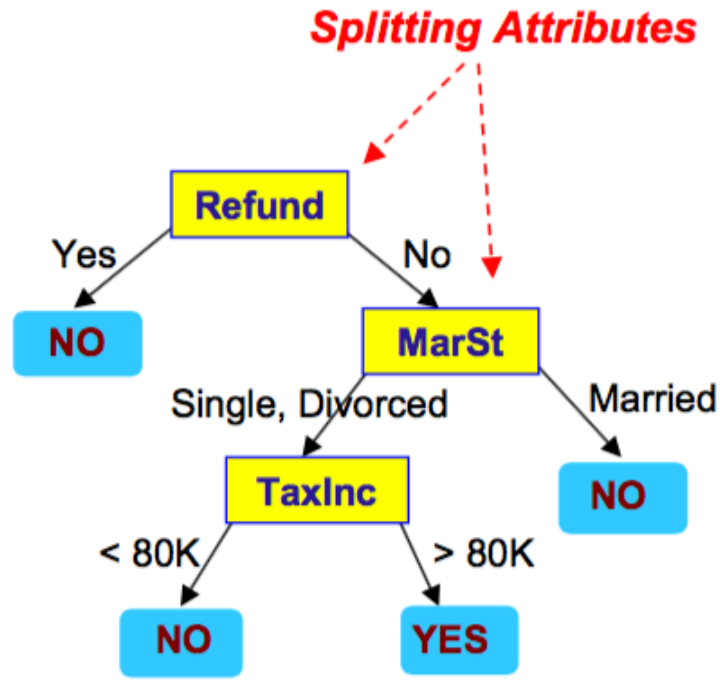
- Decision tree
  - **Internal node**: a test on an attribute
  - **Branch**: represent an outcome of a test
  - **Leaf Node**: holds a class label



Decision Tree (more example)

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

Training Data

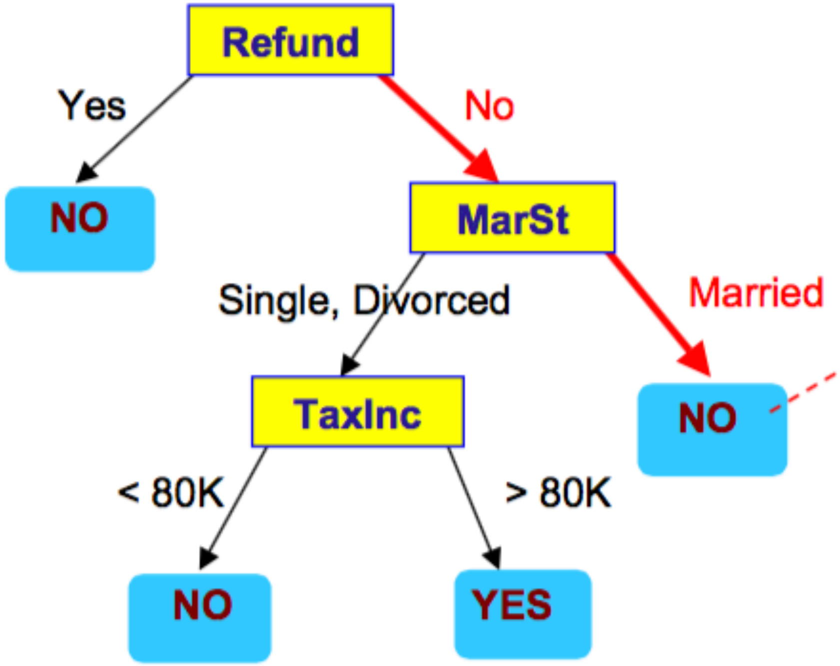


Model: Decision Tree

How are decision trees used for classification?

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

Why are decision tree classifiers so popular?

- The construction of decision trees doesn't require domain-knowledge, or parameter setting
  - Suitable for **exploratory knowledge discovery**
- Decision trees can handle **multidimensional data**
- The representation is intuitive.
- Decision trees can be easily converted to classification rules.

Building Decision Trees: basic Operators

- Attribute Selection Measures
  - How to select attributes for tree nodes

- Tree pruning
  - Large decision trees may be over fitting, many branches reflect noises, or outliers
  - Tree pruning is to remove these branches to improve classification accuracy on unseen data.

Decision Tree Induction

- Many Algorithms
  - ID3 proposed by J. Ross Quinlan in late 1970s and early 1980s
  - C4.5, a successor of ID3, also proposed by Quinlan
  - CART (Classification and Regression Tree) proposed in 1984 by a group of statisticians (L. Breiman, J. Friedman, R. Olshen and C. Stone).

- ID3, C4.5 and CART adapt a greedy approach
  - Split the records based on an attribute test that optimizes certain criterion.
  - Top-down approach, which starts with a training and recursively partitioned into smaller subsets as the tree is being built.

Decision Tree Induction

Generate_decision_tree(D, attribute_list)

(1) create a node N;
(2) **if** tuples in D are all of the same class, C, **then**
(3)   return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5)   return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
    multiway splits allowed **then** // not restricted to binary trees
(9)   *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11)   let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12)   **if** D_j is empty **then**
(13)   attach a leaf labeled with the majority class in D to node N;
(14)   **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
endfor
(15) return N;

Decision Tree Induction

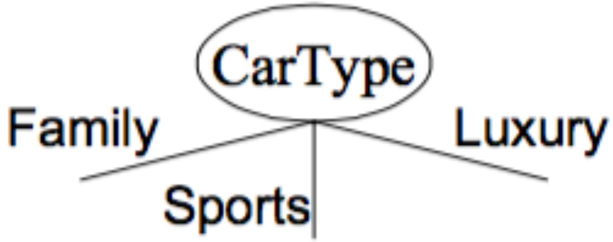
- Issues
  - Determine how to split the records (Attribute_selection_methods)
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop the splitting (step 2 and 3; step 4; step 12 and 13).
    - Stop expanding a node when all the records belong the same class
    - Stop expanding a node when all the records have similar attribute values
    - Early termination

Decision Tree Induction

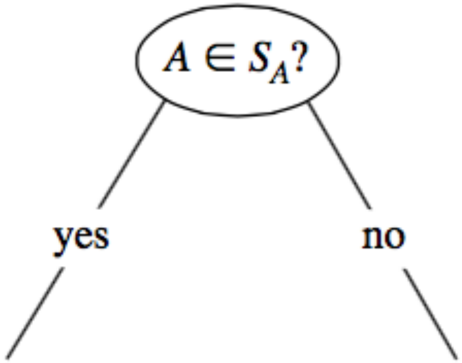
- How to specify test conditions?
  - Depending on attribute types
    - Nominal
    - Ordinal
    - Continuous
  - Depending on number of ways to split
    - 2-way split
    - Multi-way split

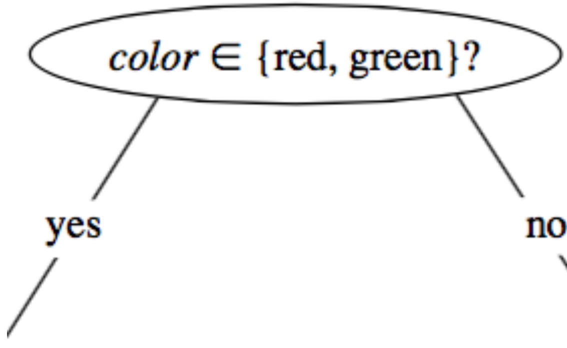
Decision Tree Induction

- 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.



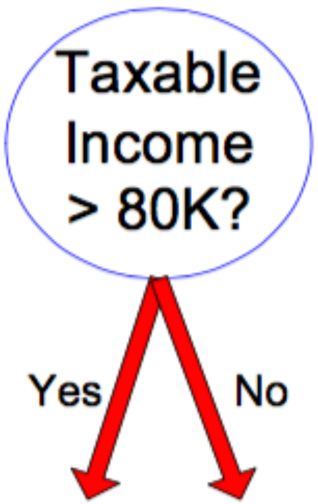


Decision Tree Induction

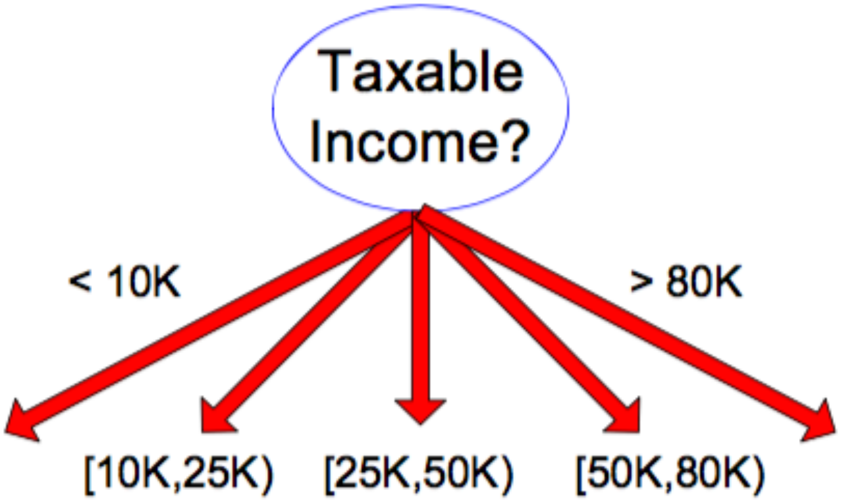
- Splitting Based on Continuous Attribute
  - **Discretization** to form an ordinal categorical attribute
    - **Static:** discretize once at the beginning
    - **Dynamic:** ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles) or clustering.
  - **Binary Decision:** (A < v) or (A >= v)
    - Consider all possible splits and finds the best cut.
    - Can be more compute intensive.

Decision Tree Induction

- Splitting Based on Continuous Attribute



(i) Binary split



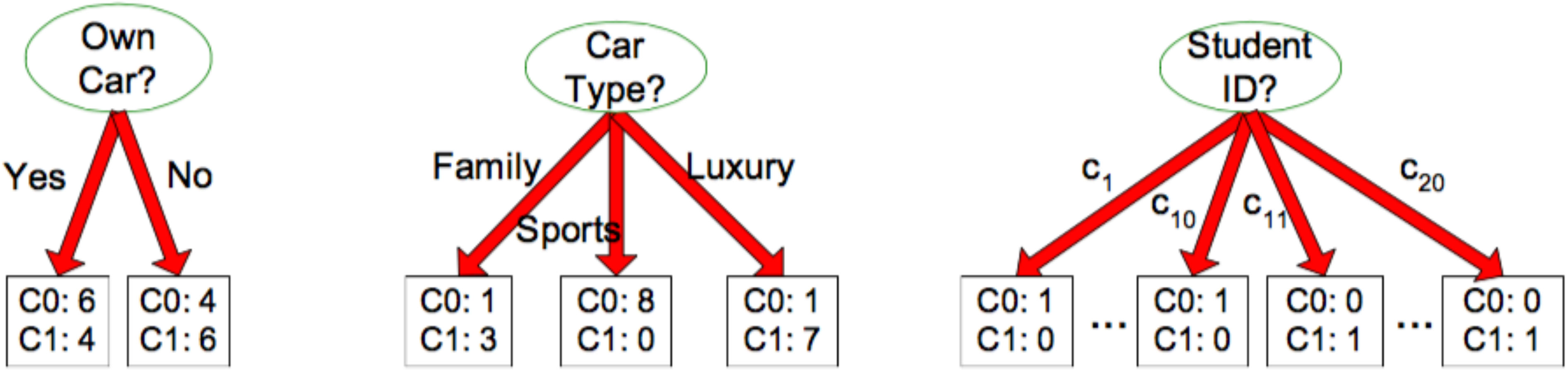
(ii) Multi-way split

Decision Tree Induction

- Issues
  - Determine how to split the records (Attribute_selection_methods)
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop the splitting (step 2 and 3; step 4; step 12 and 13).
    - Stop expanding a node when all the records belong the same class
    - Stop expanding a node when all the records have similar attribute values
    - Early termination

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

Measures of Node Impurity

- Based on INFO
  - Information Gain (used in ID3)
  - Gain Ratio (used in C4.5)

- GINI Index
  - Used in CART

Splitting Criteria based on Info

- Entropy at a given node t:

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

  - *P(j|t) is the relative frequency of class j at node t.*
  - Entropy(t) is also called Info(t)

- Entropy measures homogeneity of a node
  - Maximum (*log* n_c) when records are equally distributed among all classes implying least information.
  - Minimum (0.0) when all records belong to one class, implying most information.

Examples for computing Entropy

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

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 Criteria based on Info

- Information Gain

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

  - Parent node *p is split into k partitions*; n_i is the number of objects in partition *i*.
  - Measure reduction in entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Criteria based on Info

- Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

  - Parent node *p is split into k partitions*; n_i is the number of objects in partition *i*.
  - Adjusts Information Gain by the entropy of the partition (SplitINFO). Higher entropy partitioning (larger number of small partitions) is penalized.
  - Used in C4.5.
  - Designed to overcome the disadvantage of Information Gain.

Splitting based on GINI

- Measure of Impurity: GINI
- Gini Index for a given node t:

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

  - (NOTE: *p(j|t)* is the relative frequency of class j at node t)
  - Maximum $(1-1/n_c)$ when records are equally distributed among all classes, implying least interesting information.
  - Maximum (0.0) 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	

Examples for computing GINI

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

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

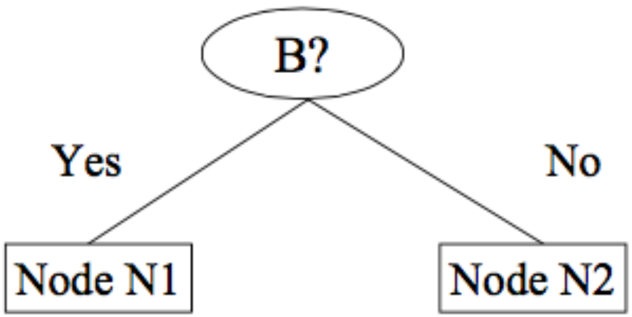
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

- Split 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)
$= 1-(1/5)^2-(4/5)^2$
$=0.319$

	N1	N2
C1	5	1
C2	2	4
GINI=0.371		

GINI(Split)
$= 7/12*0.408+5/12*0.319$
$=0.371$

Categorical Attributes: Computing GINI Index

- 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)

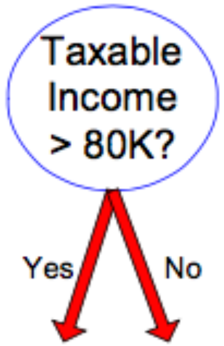
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

Continuous Attributes: Computing GINI Index

- Use Binary Decisions based on one value
- Several choices for the spliting value
  - Number of possible splitting values = number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A<v and A >= v
- Simple method to chose best v
  - For each v, scan the database to gather count matrix and compute its GINI index.
  - Computational 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



Continuous Attributes: Computing GINI Index

- 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

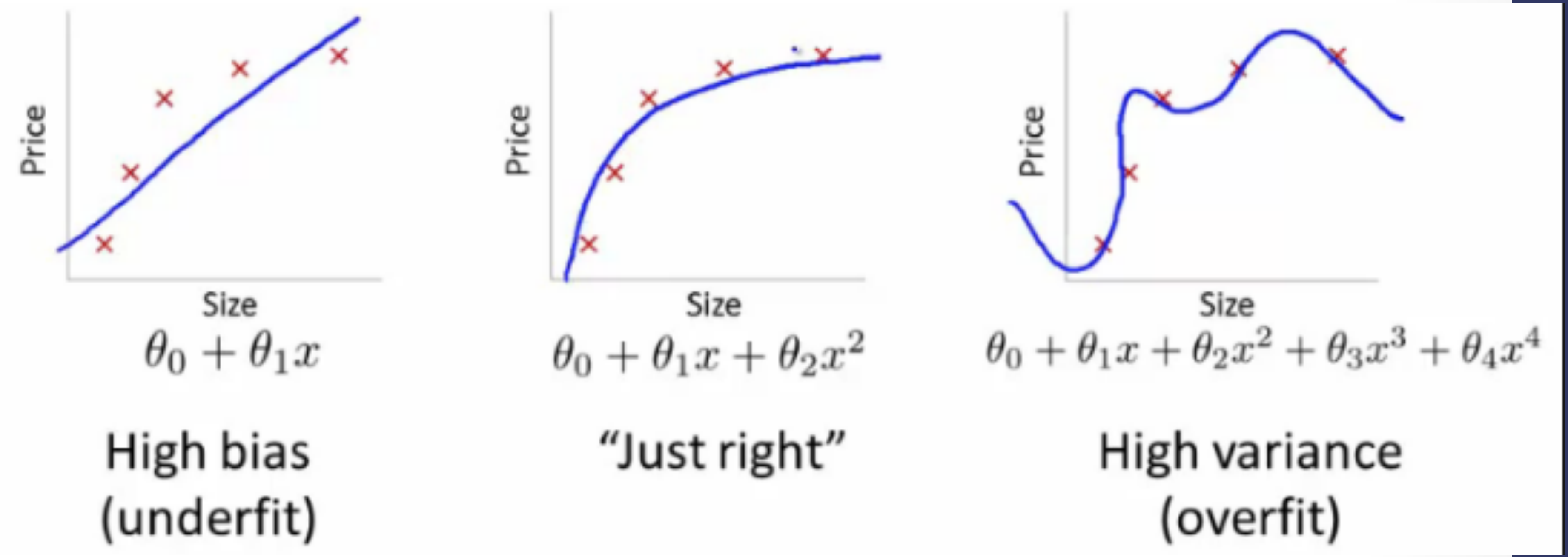
Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values →	60	70	75	85	90	95	100	120	125	220

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	

Practical Issues of Classification

- Under fitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting



Underfitting and Overfitting

- Underfitting: when model is too simple, both training and testing errors are high.

- Overfitting:
  - The model cannot generalize well to new data.
  - Overfitting results in decision trees that are more complex than necessary.
  - Training error no longer provides a good estimate on how well the tree will perform on previously unseen records.

Estimating Generalization Errors

- Re-substitution errors: error on training
- Generalization errors: error on testing
- Methods for estimating generalization errors:
  - Reduced error pruning (REP)
    - Uses validation data set to estimate generalization error.

Occam's Razor

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

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

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

How to Address Overfitting

- Pre-pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a full-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:
    - 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 chi-square test)
    - Stop if expanding the current node doesn't imporve impurity measures (GINI or information gain)

How to Address Overfitting

- Post-prunning
  - Grow decision tree to its entirely
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error 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.

Handling Missing Attribute Values

- Missing values affect decision tree construction in different ways
  - Affects how impurity measures are computed
  - Affects how to distribute instances with missing value to child nodes.
  - Affects how a test instance with missing value is classified.

Handling Missing Attribute Values

- Computing Impurity Measure

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	?	Single	90K	Yes

Missing value

Before Splitting:

Entropy(Parent)
= -0.3 log(0.3)-(0.7)log(0.7) = 0.8813

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

Entropy(Refund=Yes) = 0

Entropy(Refund=No)
= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183

Entropy(Children)
= 0.3 (0) + 0.6 (0.9183) = 0.551

Gain = 0.9 × (0.8813 – 0.551) = 0.3303

Handling Missing Attribute Values

- Distribute Instances

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

Refund

Yes

Class=Yes	0
Class=No	3

No

Cheat=Yes	2
Cheat=No	4

Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes

Refund

Yes

Class=Yes	0 + 3/9
Class=No	3

No

Class=Yes	2 + 6/9
Class=No	4

Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

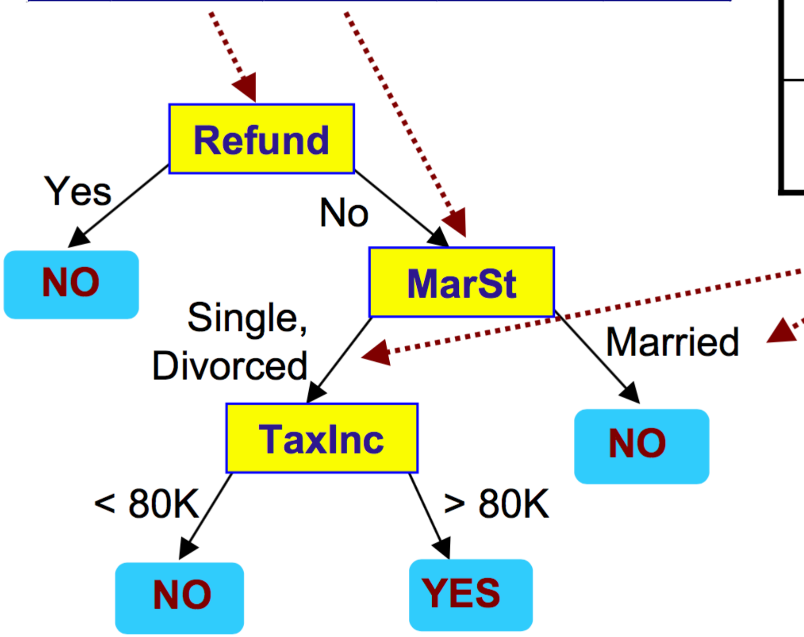
Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Handling Missing Attribute Values

- Classifying Instances

New record:

Tid	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?



	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	6/9	1	1	2.67
Total	3.67	2	1	6.67

Probability that Marital Status = Married is 3.67/6.67

Probability that Marital Status ={Single,Divorced} is 3/6.67

Outline

- Basic Concepts
  - Classification: Definition
  - Examples of classification tasks
  - General Approach to Classification
- Basic Classification Techniques
  - Decision Tree Induction
  - Naïve Bayesian Classification
- Classification Evaluation

Bayesian Classifiers

- Statistical classifiers
  - Predict class membership probabilities
  - Based on Bayes' theorem

- Naïve Bayesian classifier is a simple Bayesian classifier
  - Comparable in performance with decision tree and selected neural network classifiers
  - High accuracy and speed when applied to large databases.
  - Using **class-conditional independence** to simplify the computations involved.

Bayes' Theorem

- The theorem is named after Thomas Bayes, a British statistician and philosopher in the 18th century.

- Bayes' Theorem

$$P(H|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|H)P(H)}{P(\boldsymbol{X})}.$$

  - **X:** a data object, represented by an attribute vectors (feature vector)
  - **H**: some hypothesis such as **X** belongs a specific class **C**
  - P(H|X): posterior probability
  - P(H): prior probability of H

Naïve Bayesian Classification

- Let D = $\{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_n, y_n)\}$ be a training set of data objects and its labels
  - $\mathbf{X}_i=(x_1, x_2, \ldots, x_p)$ is a p-dimensional attribute vector
  - The class label of the object ith is $y_i \in \{C_1, C_2, \ldots, C_m\}$ where $C_1, C_2, \ldots, C_m$ are *m* predefined classes.
- Given a data object **X,** naïve Bayes predicts that it belongs to the class C_i if and only if

$$P(C_i|\boldsymbol{X}) > P(C_j|\boldsymbol{X}) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

Recall that $P(C_i|\boldsymbol{X}) = \dfrac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})}$

Where *P(**X**)* is constant for all classes; $P(C_i)$ can be assumed as equally likely or calculated from training data set (as the percentage of class C_i in the training data set).

Naïve Bayesian Classification

- How to calculate $P(\boldsymbol{X}|C_i)$?
- Class-conditional independence assumption

$$P(X|C_i) = \prod_{k=1}^{p} P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i)...P(x_p|C_i)$$

  - If attribute A_k is categorical, P(xk|Ci) is the number of instances of class C_i in D having the value x_k for the attribute A_k, divided by the $|C_{i,D}|$, the number of instances with class $C_{i.}$
  - If A_k is continuous

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

which is the Gaussian distribution with mean and standard deviation μ_{C_i}, σ_{C_i} calculated for attribute Ak for training instances with class C_i

Naïve Bayesian Classification

- Issue:
  - What happens if we have zero probability for some $P(x_i|C_i)$?
- Laplacian correction
  - Adding one for each count when calculating $P(x_k|C_i)$

Outline

- Basic Concepts
  - Classification: Definition
  - Examples of classification tasks
  - General Approach to Classification
- Basic Classification Techniques
  - Decision Tree Induction
  - Naïve Bayesian Classification
- Classification Evaluation

Classification Evaluation

- Consider binary classification with 2 classes (Positive, Negative)
  - True Positive (TP):
    - The positive instances that are correctly classified
  - True Nagative (TN)
    - The negative instances that are correctly classified
  - False Positive (FP)
    - The negative instances that are incorrectly classified as Positive
  - False Negative (FN)
    - The positive instances that are incorrectly clssified as Negative.

Classification Evaluation

- Confusion Matrix

Predicted class

		yes	*no*	Total
Actual class	*yes*	TP	FN	P
	no	FP	TN	N
	Total	P'	N'	$P+N$

Classification Evaluation

Measure	*Formula*
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

Classification Evaluation

- Given *m* classes, a confusion matrix is a table of at least *mxm*.
  - $CM_{i,j}$ indicates the number of instances of class I that were classified as the class j.

		Prediction				
		Class 1	Class 2	Class 3	...	Class n
Actual	Class 1	Accurate				
	Class 2		Accurate			
	Class 3			Accurate		
	...				Accurate	
	Class n					Accurate

Holdout methods

- Cross validation
- Leave-one-out

Outline

- Basic Concepts
  - Classification: Definition
  - Examples of classification tasks
  - General Approach to Classification
- Basic Classification Techniques
  - Decision Tree Induction
  - Naïve Bayesian Classification
- Classification Evaluation