Association rules

Why association rules (in data mining)

- Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository.
- An example of an association rule would be "If a customer buys a dozen eggs, he is 80% likely to also purchase milk."

Market basket analysis

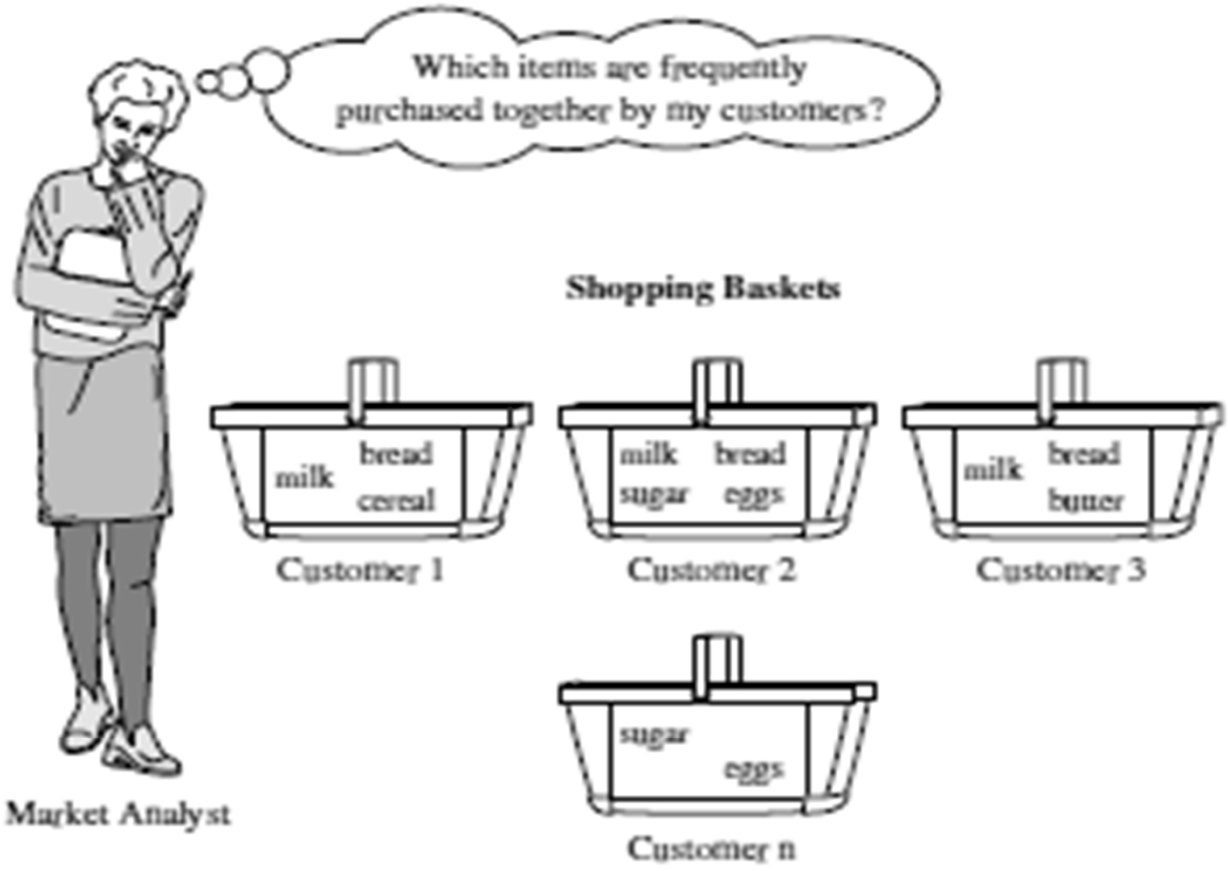


Figure 5.1 Market basket analysis.

- Rule support and confidence are two measures of rule .
- Association rule are considered interesting if they satisfy both min.support threshold and min.confidence threshold .
- Such threshold can be set by users or domain expert

$$computer \Rightarrow antivirus\_software \ [support = 2\%, confidence = 60\%]$$

- An association rule has two parts, an antecedent (if) and a consequent (then).
- An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.
- In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in shopping data analysis, product clustering, catalog design and store layout etc….

An association rule

- An itemset is a set of items.
  - E.g., {milk, bread, corn} is an itemset.
- A k-itemset is an itemset with k items.
- An association rule is about relationships between two disjoint itemsets X and Y

  $$X \Rightarrow Y$$

- It presents the pattern when X occurs, Y also occurs

Use of Association Rules

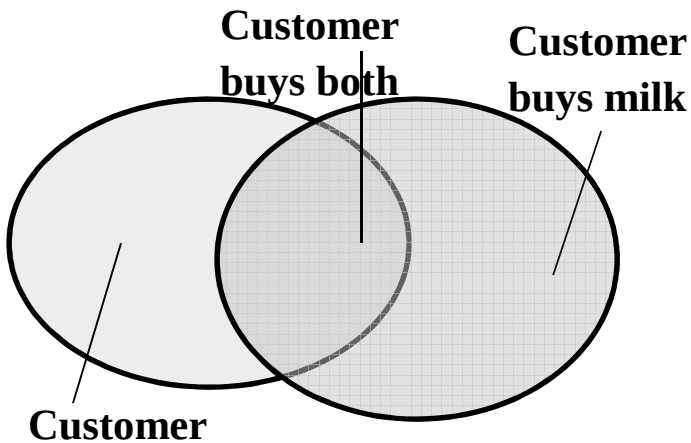
- Association rules do not represent any sort of causality or correlation between the two itemsets.
  - $X \Rightarrow Y$ does not mean X causes Y,
  - $X \Rightarrow Y$ can be different from $Y \Rightarrow X$, unlike correlation
- Association rules assist in marketing, targeted advertising, floor planning, inventory control,…

Frequent Pattern Analysis

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Market data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis….

Basic Concepts: Frequent Patterns and Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F



- Itemset X = $\{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let $sup_{min} = 50\%$, $conf_{min} = 50\%$

Association rules:

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent item set generation is still computationally expensive

Frequent pattern mining can be classified in various ways, based on the following criteria:

1.Based on the completeness of patterns to be mined.: closed frequent item set, maximal frequent item set , constrained set, approximate set etc…

2. Based on the levels of abstraction involved in the rule set eg:

i. buys(X, “computer”))=> buys(X, “HP printer”)

ii. buys(X, “laptop computer”))buys=>(X, “HP printer”)

3. Based on the number of data dimensions involved in the rule : single dimension / multi dimension eg: buys(X, “computer”))=>buys(X, “antivirus software”)

ii. age(X, “30: : :39”)^income(X, “42K: : :48K”))buys(X, “high resolution TV”)

4. Based on the types of values handled in the rule: eg :

Boolean association rule/ quantitative association rule.

5. Based on the kinds of rules to be mined: eg: correlation rules /strong gradient relationships

Eg: "The average sales from Sony Digital Camera increase over 16% when sold together with Sony Laptop Computer": both Sony Digital Camera and Sony Laptop Computer are siblings, where the parent itemset is Sony.

- 6. Based on the kinds of patterns to be mined: sequential pattern/ structured pattern eg : customers may tend to first buy a PC, followed by a digital camera, and then a memory card.

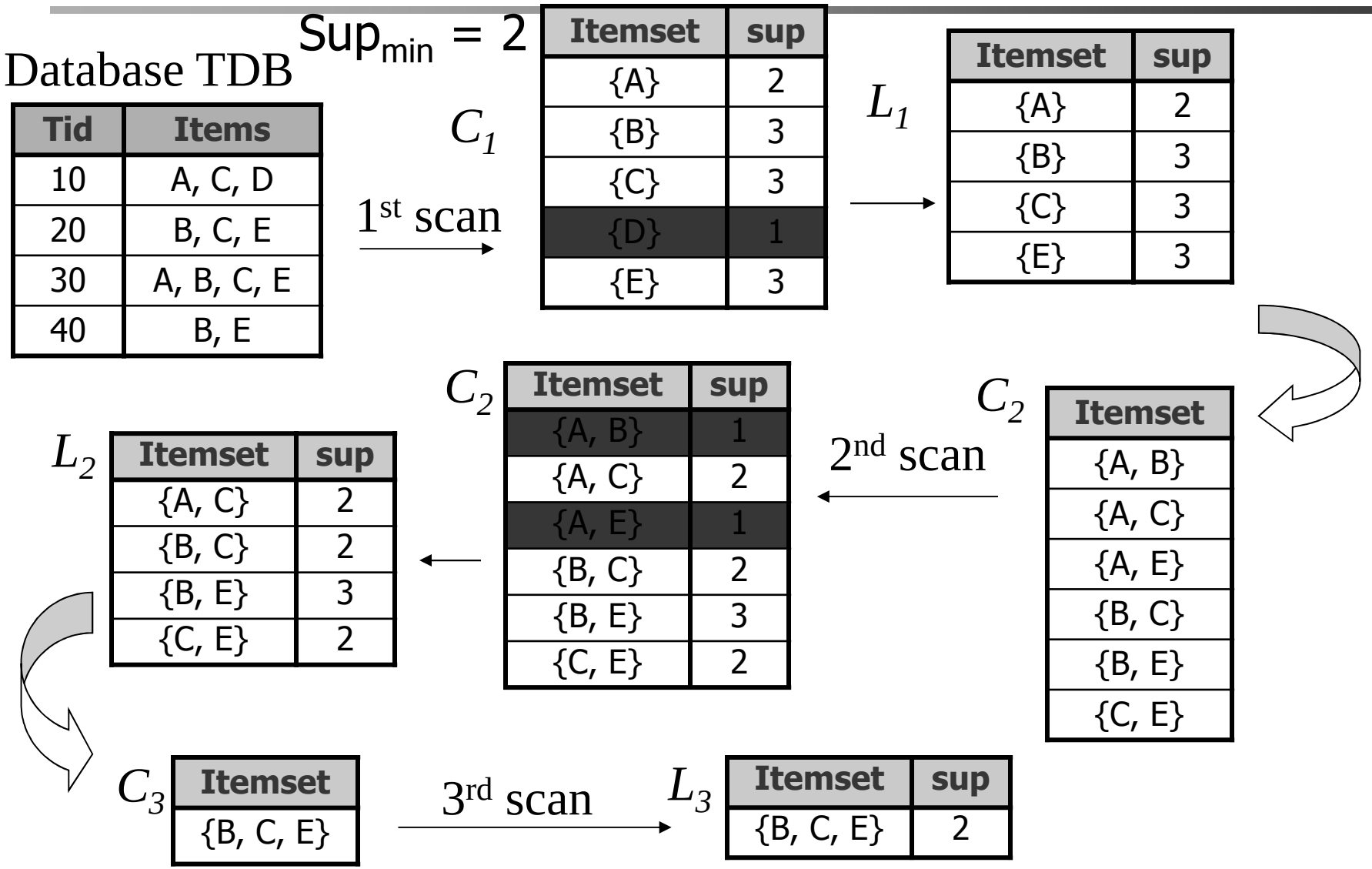
Scalable Methods for Mining Frequent Patterns

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{egg,milk,fruits}** is frequent, so is **{egg,milk}**
  - i.e., every transaction having {**egg,milk,fruits**} also contains {**egg,milk**}
- Scalable mining methods: Three major approaches
  - Apriori
  - Freq. pattern growth
  - Vertical data format approach

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example



Apriori Algorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent of each candidate by scanning the DBCount the support
    - Eliminate candidates that are infrequent, leaving only those that are frequent

Note: This algorithm makes several passes over the transaction list

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc*, *abd*, *acd*, *ace*, *bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4={*abcd*}

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Minimize number of candidates
  - Facilitate support counting of candidates

Hashing: Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

Transaction Reduction: Transactions does not contain any frequent k-itemsets cannot contain any frequent k+1 itemset. Mark or Remove such transaction from subsequent scan.

Partition: Scan Database Only Twice

- Any item set that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: Partition database and find local frequent patterns
  - Scan 2: Consolidate global frequent patterns

Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample.
- Scan database again to find missed frequent patterns in the data set.

Dynamic Itemset Counting: Reduce Number of Scans

- Data Base is portioned into blocks marked by start points.
- New candidate itemset can be added at any start point.

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach
- Depth-first search
  - Avoid explicit candidate generation
- Major philosophy: Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

Construct FP-tree from a Transaction Database

TID	Items bought (ordered)	frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

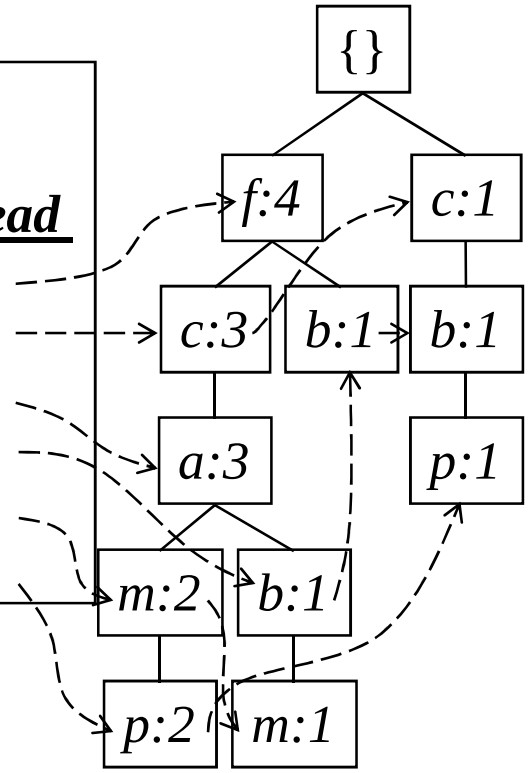
min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

F-list=f-c-a-b-m-p

Header Table

Item	frequency	head
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

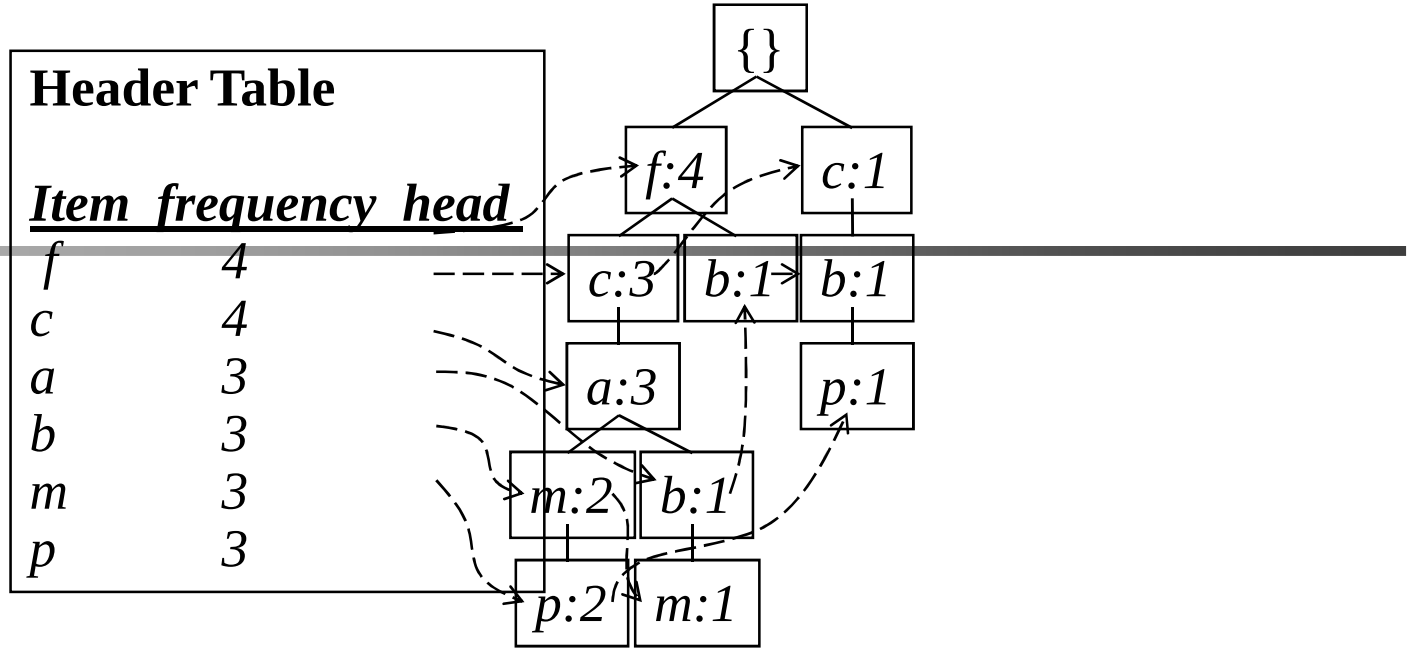


Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item *p*
- Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional pattern base

Mining Frequent Patterns With FP-trees

- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path — single path will generate all the combinations of its sub-paths, each of which is a frequent pattern



Conditional (sub) pattern bases

Item	*Cond. pattern base*	*Cond. FP-tree*	*F P Generated*
c	*f:3*	*f:3*	*fc:3*
a	*fc:3*	*fc:3*	*fa:3, ca:3, fca:3*
b	*fca:1, f:1, c:1*		
m	*fca:2, fcab:1*	*fca:3*	*fm:3, cm:3, am:3, fcm:3, fam:3, cam:3, fcam:3*
p	*fcam:2, cb:1*	*c:3*	*cp:3*

Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

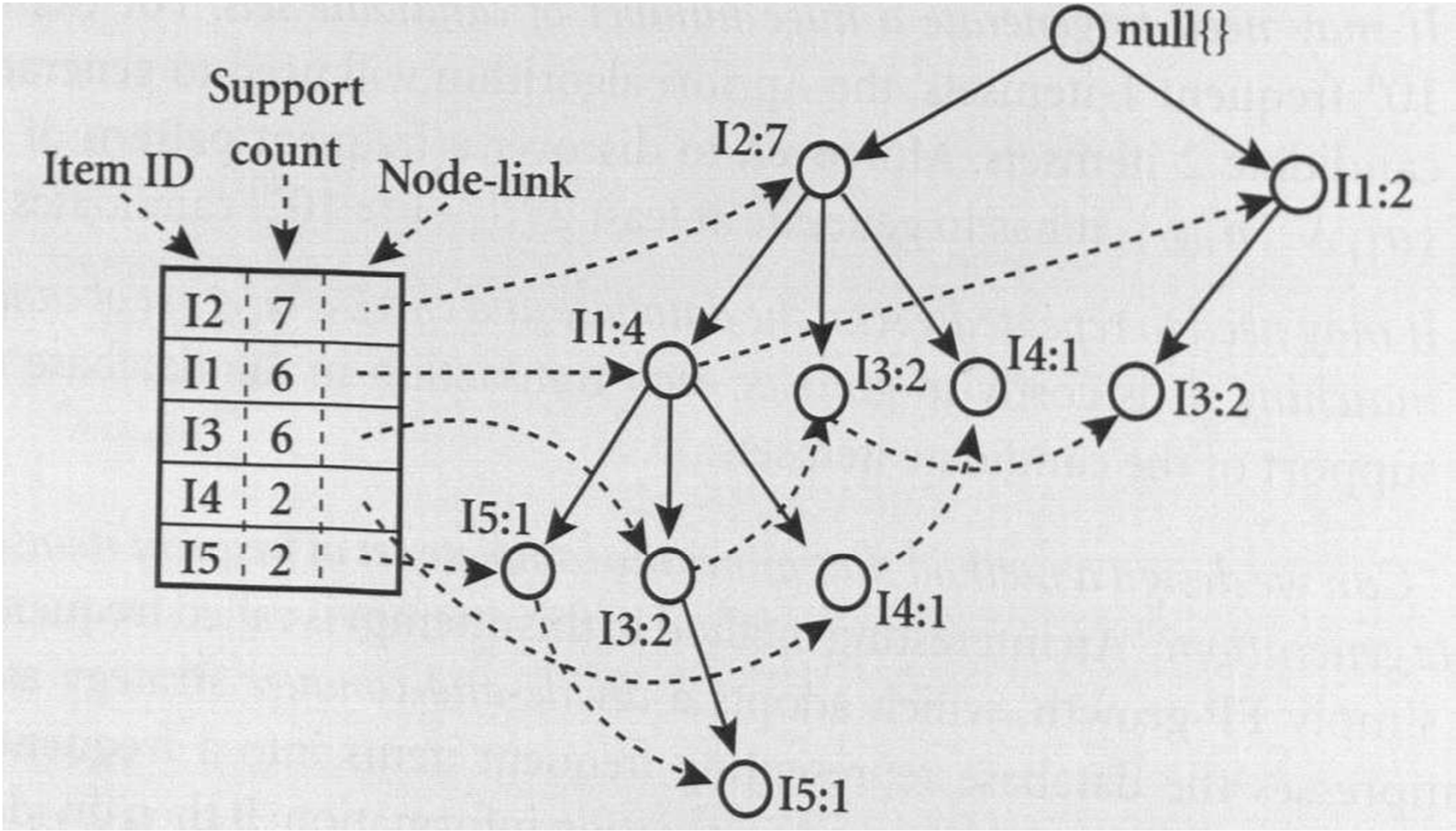
Step-1:

Item	Count
I1	6
I2	7
I3	6
I4	2
I5	2

Step-2: Arrange Transaction in descending order

TID	List of item (Before)	List of item (After)
T100	I1, I2, I5 I2,	I2,I1,I5
T200	I4	I2,I4
T300	I2, I3	I2,I3
T400	I1, I2, I4 I1,	I2,I1,I4
T500	I3	I1,I3
T600	I2, I3	I2,I3
T700	I1, I3	I1,I3
T800	I1, I2, I3, I5	I2,I1,I3,I5
T900	I1, I2, I3	I2,I1,I3

FP-TREE


Item ID
Support count
Node-link
I2 7
I1 6
I3 6
I4 2
I5 2
null{}
I2:7
I1:2
I1:4
I3:2
I4:1
I3:2
I5:1
I3:2
I4:1
I5:1

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Pattern Generated
I5	{{I2, I1:1}, {I2, I1, I3:1}}	(I2:2, I1:2)	{I2, I5:2}, {I1, I5:2}, {I2, I1, I5:2}
I4	{{I2, I1:2}, {I2:1}}	(I2:2)	{I2, I4:2}
I3	{{I2, I1:2}, {I2:2}, {I1:2}}	(I2:4, I1:2), (I1:2),	{I2, I3:4}, {I1, I3:4}, {I2, I1, I3:2}
I1	{{I2:4}}	(I2:4)	{I2, I1:4}

Mining frequent itemsets using vertical data format

- Transforming the horizontal data format of the transaction database D into a vertical data format:

Itemset	TID_set
I1	{T100, T400, T500, T700, T800, T900}
I2	{T100, T200, T300, T400, T600, T800, T900}
I3	{T300, T500, T600, T700, T800, T900}
I4	{T200, T400}
I5	{T100, T800}

Generating Association Rules from Frequent Itemsets

- *Strong* association rules satisfy both minimum support and minimum confidence.

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

- For each frequent itemset l, generate all nonempty subsets of l.
- For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq$ *min_conf*, where *min_conf* is the minimum confidence threshold.

Example 5.4 Generating association rules. Let's try an example based on the transactional data for *AllElectronics* shown in Table 5.1. Suppose the data contain the frequent itemset $l = \{I1, I2, I5\}$. What are the association rules that can be generated from l? The nonempty subsets of l are $\{I1, I2\}$, $\{I1, I5\}$, $\{I2, I5\}$, $\{I1\}$, $\{I2\}$, and $\{I5\}$. The resulting association rules are as shown below, each listed with its confidence:

$$
\begin{aligned}
&I1 \wedge I2 \Rightarrow I5, && \textit{confidence} = 2/4 = 50\% \\
&I1 \wedge I5 \Rightarrow I2, && \textit{confidence} = 2/2 = 100\% \\
&I2 \wedge I5 \Rightarrow I1, && \textit{confidence} = 2/2 = 100\% \\
&I1 \Rightarrow I2 \wedge I5, && \textit{confidence} = 2/6 = 33\% \\
&I2 \Rightarrow I1 \wedge I5, && \textit{confidence} = 2/7 = 29\% \\
&I5 \Rightarrow I1 \wedge I2, && \textit{confidence} = 2/2 = 100\%
\end{aligned}
$$

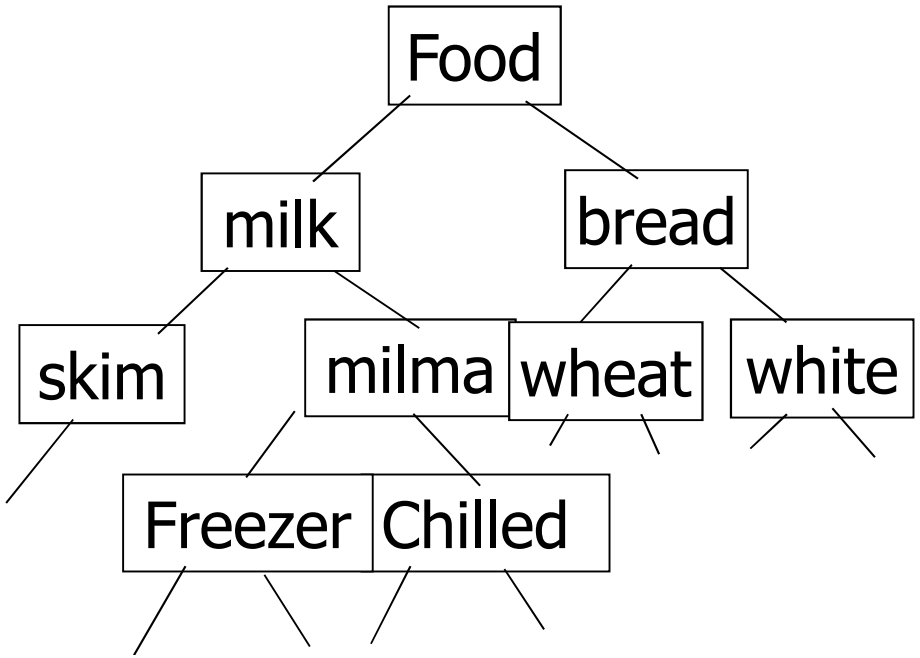
If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong. Note that, unlike conventional classification rules, association rules can contain more than one conjunct in the right-hand side of the rule. ∎

Mining Various Kinds of Association Rules

1. Mining multilevel association
2. Mining multidimensional association
3. Mining quantitative association
4. Mining interesting correlation patterns

1. Mining Multiple-Level Association Rules

- Items often form hierarchies
- Items at the lower level are expected to have lower support



Multilevel Associatons

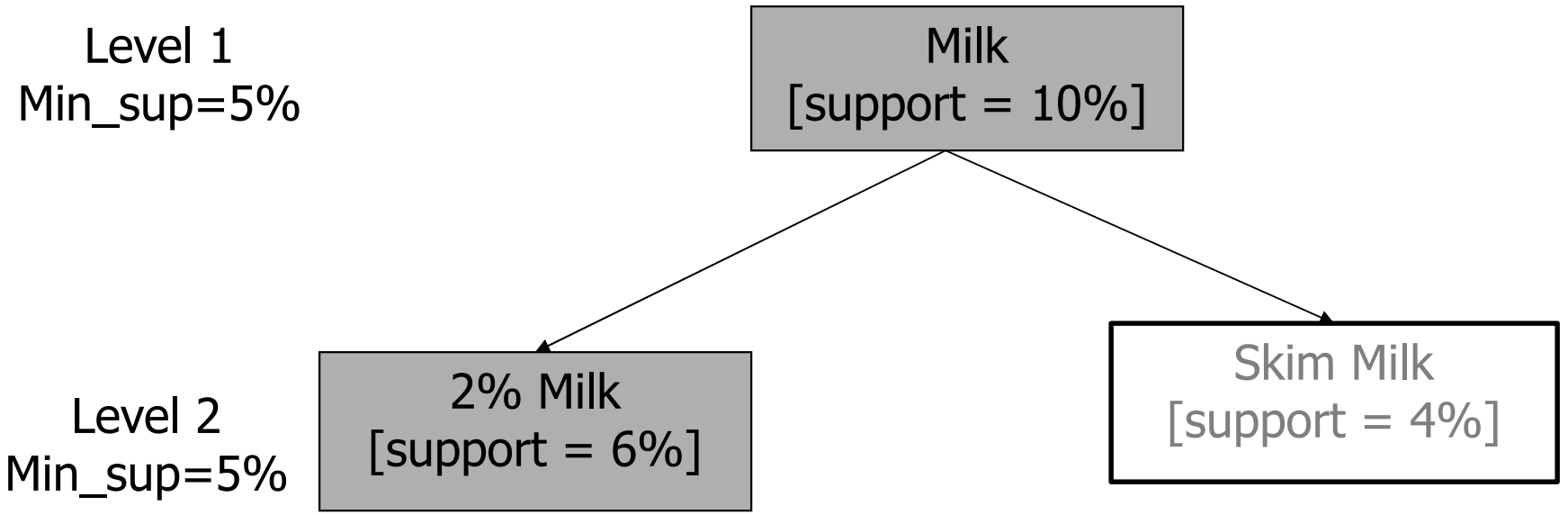
- Mining multilevel associations can be of the following:
- A top_down, progressive deepening approach:
  - First find high-level strong rules:
    - milk -> bread [20%,60%].
  - Then find their lower-level “weaker rules:
    - 2% milk -> wheat bread[6%,50%].

Multilevel Associations

- Mining multilevel associations can be of the following:
- Variations at mining multiple-level association rules:
  - Level –crossed association rules:
    - 2% milk -> Wonder wheat bread
  - Association rules with multiple, alternative hierarchies:
    - 2% milk -> Wonder bread.

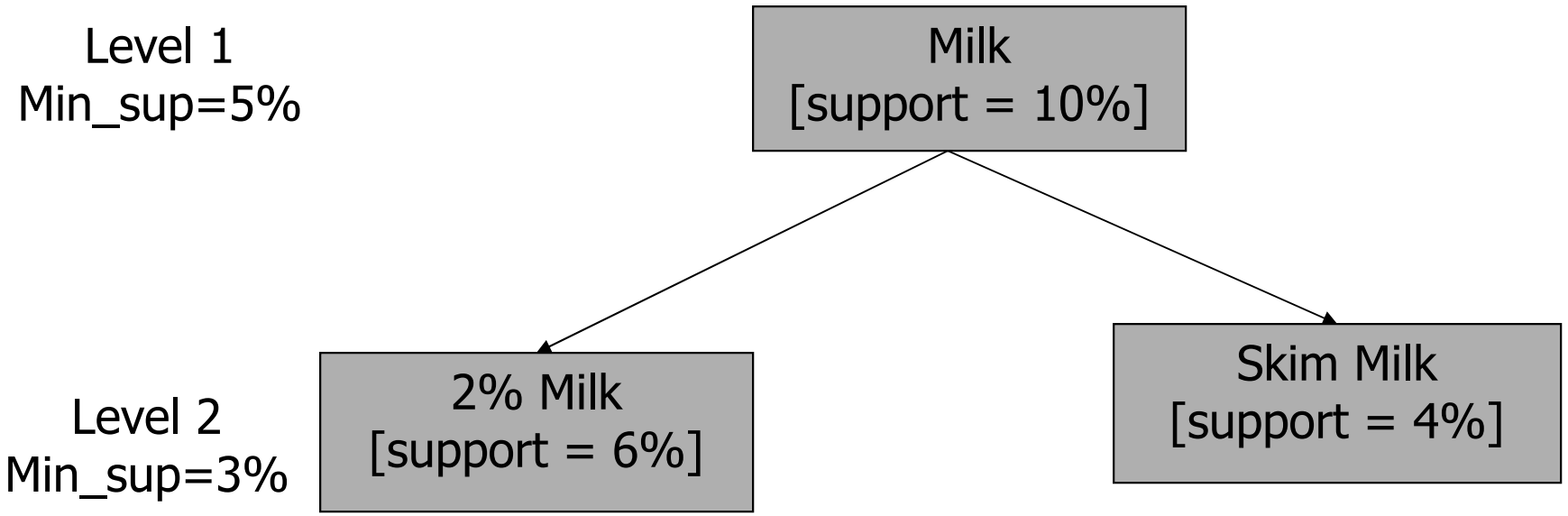
Multilevel Association: Uniform Support Vs Reduced Support

- Uniform Support: This means the same minimul support for all levels.
- There will be only one minimum support threshold.
- No need to examine itemsets containing any item whose ancestors do not have minimum support.



Multilevel Association: Uniform Support Vs Reduced Support

- Reduced Support: This means the minimum support will be reduced minimum support at lower levels



Multilevel Association: Redundancy Filtering

- Some rules may be redundant due to "ancestor" relationships between items.
- Consider the following example

milk -> wheat bread[support = 8%, confidence = 70%]

2%milk-> wheat bread [support = 2%, confidence = 72%]

Multilevel Association: Progressive Deepening

- A top-down , progressive deepening approach can be considered as follows:

First mine high-level frequent items:

milk(15%), bread(10%)

Then mine their lower-level "weaker" frequent itemsets:

2% milk(5%), wheat bread(4%)

2. Mining Multi-Dimensional Association

- Single-dimensional rules:

  buys(X, “milk”) $\Rightarrow$ buys(X, “bread”)

- Multi-dimensional rules: $\geq$ 2 dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    age(X,”19-25”) $\wedge$ occupation(X,“student”) $\Rightarrow$ buys(X, “coke”)

  - hybrid-dimension assoc. rules (*repeated predicates*)

    age(X,”19-25”) $\wedge$ buys(X, “popcorn”) $\Rightarrow$ buys(X, “coke”)

- Categorical Attributes: finite number of possible values, no ordering among values – ie, Occupation, Brand, Colour
- Quantitative Attributes: numeric values, ie, age , salary , ht, wt etc….

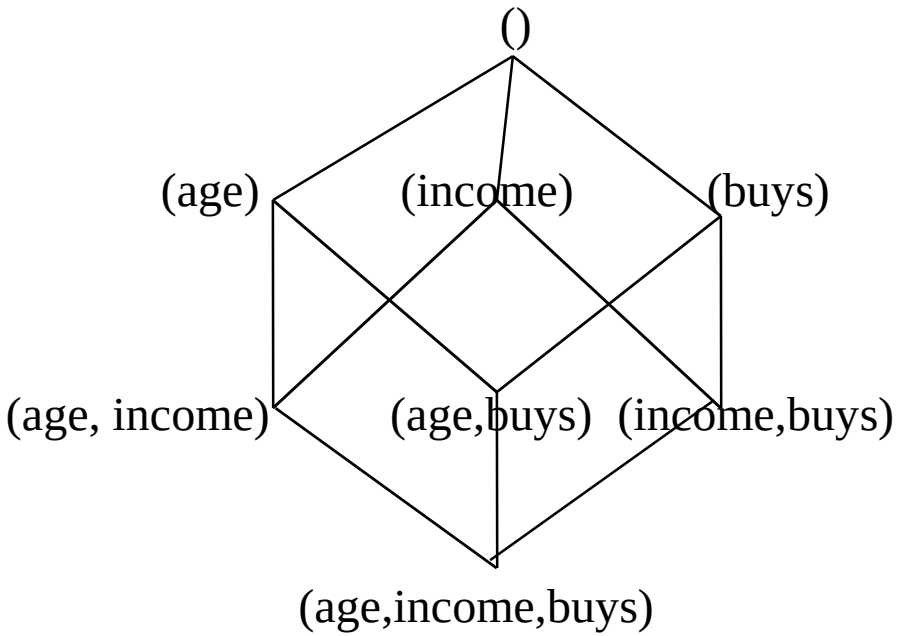
3.Mining Quantitative Associations

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules)
3. Clustering

Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- Data cube is well suited for mining.
- The cells of an n-dimensional

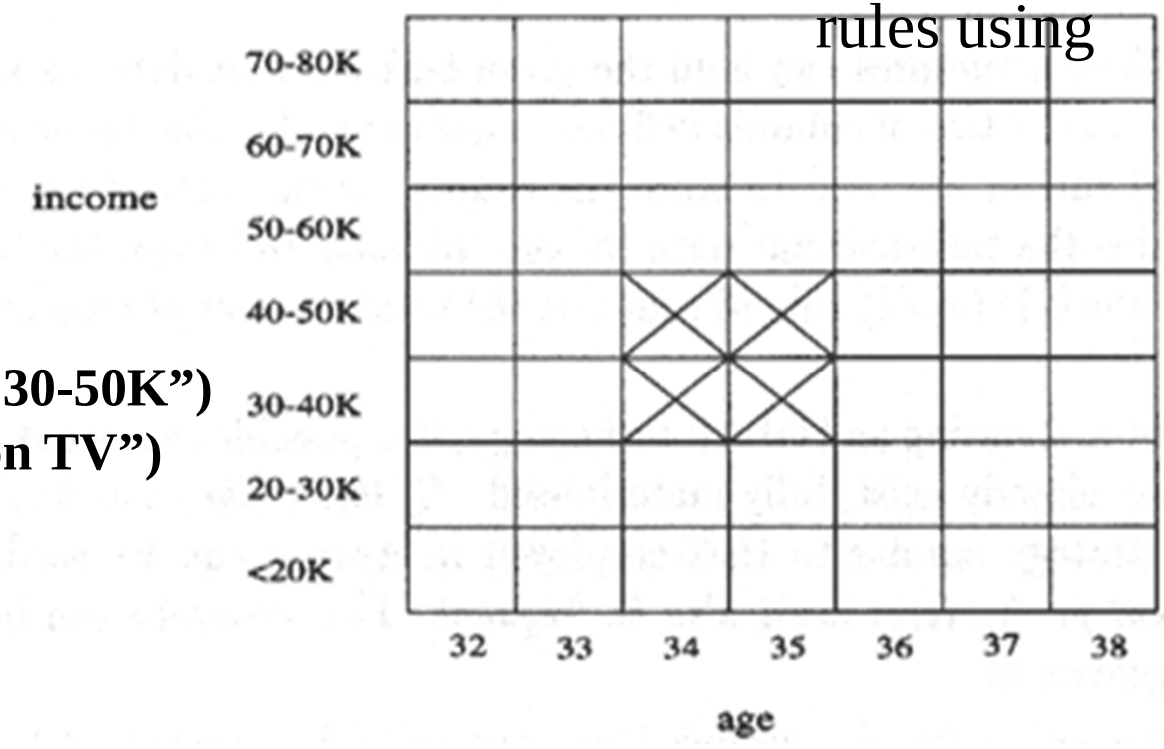
cuboid correspond to the predicate sets.



Quantitative Association Rules

- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Example

age(X,"34-35") $\wedge$ income(X,"30-50K") $\Rightarrow$ buys(X,"high resolution TV")



Association Rule Mining to Correlation Analysis

- Interestingness Measure: Correlations (Lift)
- Consider following association rule

Buys(X,"Computer games") ===> Buys(X,"videos") [supp = 40%,conf=66%]

- This rule is misleading because computer games and videos are negatively associated.

- Correlation measure used to augment the sup-conf for association rules

A===> B [support, confidence ,correlation]

$$lift(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

Classification And Prediction

Classification

- Predicts categorical class labels.
- A data analysis task where a model or classifier is constructed to predict categorical labels
- Ex: a bank loan officer needs analysis of her data in order to learn which loan applicants are "safe" and which are "risky" for bank.
- A medical researcher wants to analyze bone cancer data in order to predict which one of three specific treatment a patient should receive.

Prediction

- Data analysis , predicts a continuous-valued functions, or ordered values. .
- Predicts unknown or missing values.
- Regression analysis is a statistical methodology that is most often used for numeric prediction.

Typical applications

- Credit approval
- Target marketing
- Medical diagnosis
- Fraud detection

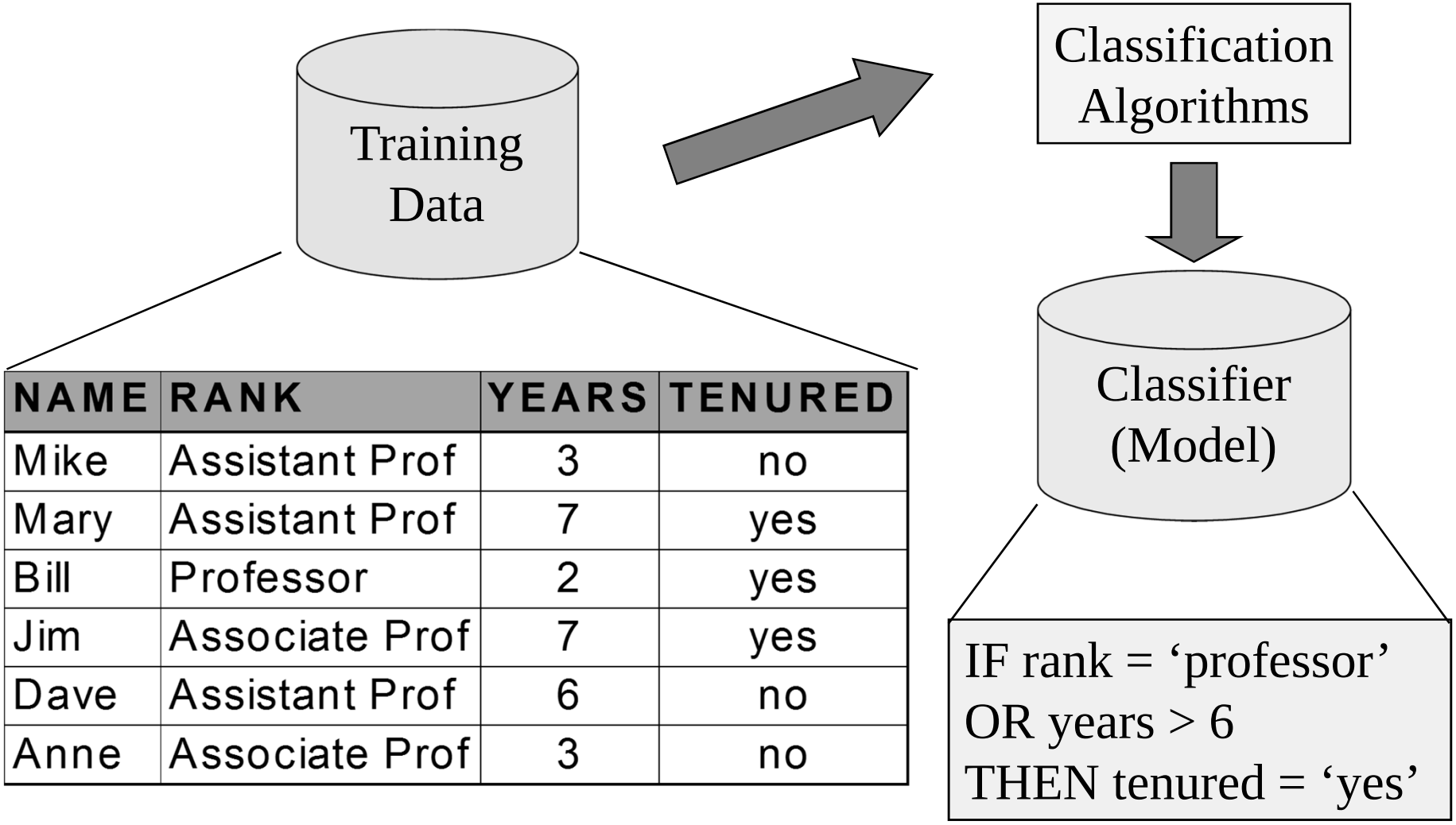
Classification—A Two-Step Process

- Learning step:
- A classification algorithm built the classifier by analyzing or ”learning from” a training set made up of database tuple and their associated class label.
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

- Model usage: In second step a model is used for classifying future or unknown objects
  - Estimate accuracy of the model.
  - The known label of test sample is compared with the classified result from the model.
  - Accuracy rate is the percentage of test set samples that are correctly classified by the model.
  - Test set is independent of training set, otherwise over-fitting will occur

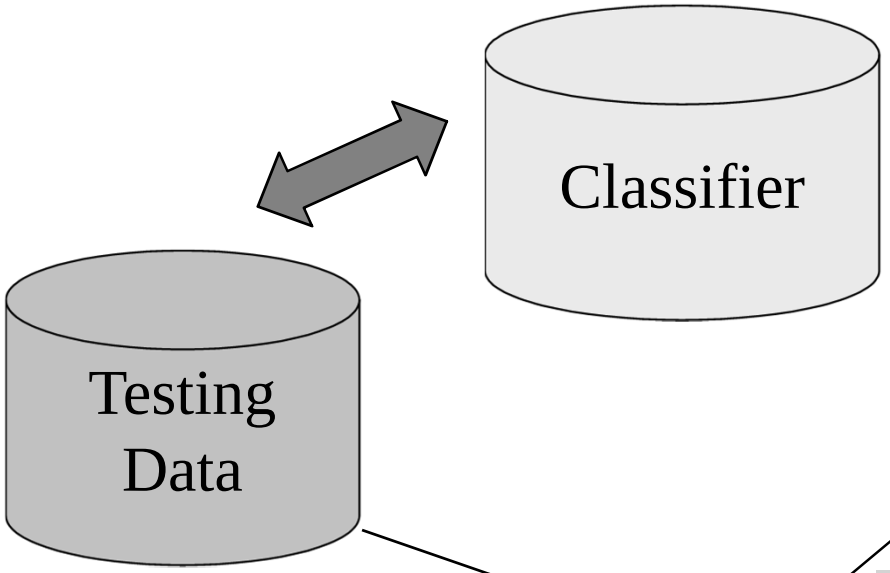
- If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

(Jeff, Professor, 4)

Tenured?

Yes

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Issues: Regarding classification and prediction

- Preparing the data for classification and prediction
  - The following preprocessing steps may be applied to the data to help improve the accuracy, efficiency, and scalability of the classification or prediction process.
- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values.
  - Eg. By replacing a missing value with the most commonly occurring value for that attribute.

Issues: Regarding classification and prediction

- Relevance analysis
  - Many of the attribute in the data may be redundant.
  - Co relation analysis can be used to identify whether any two given attributes are statistically related.
  - Remove the irrelevant or redundant attributes

Issues: Regarding classification and prediction

- Data transformation
  - Data may be transformed by normalization, particularly when neural network or methods involving distance measurements are used in the learning step.
  - Data may be transformed by generalizing it to higher level concepts. Concept hierarchy may used for that.
  - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Classification and prediction can be compared and evaluated according to the following criteria.
- Accuracy :

  The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data.

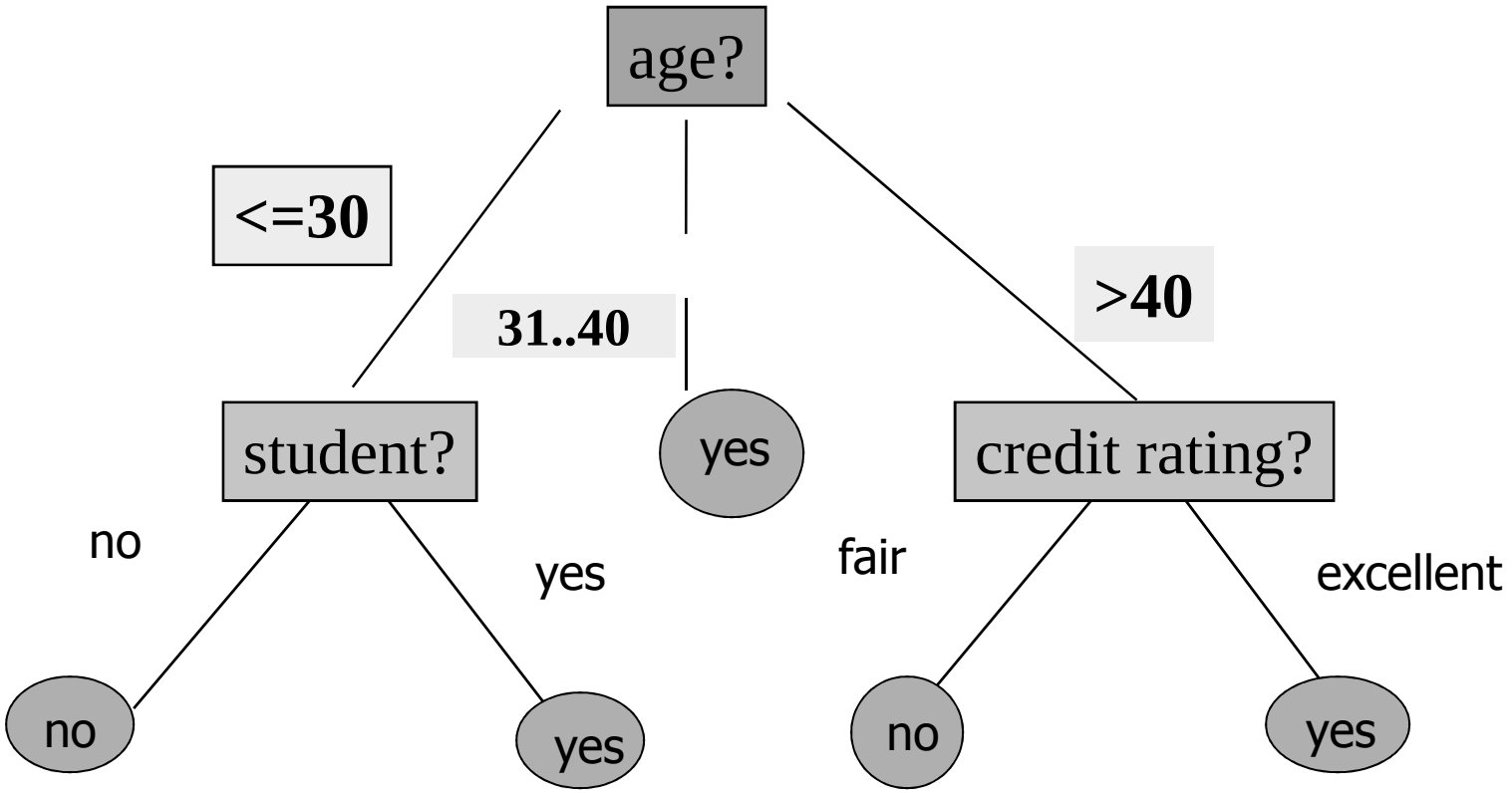
  The accuracy of predictor : refers to how well a given predictor can guess value of predicted attribute for new or previously unseen data.

- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness:** is the ability of the classifier or predictor to make correct predictions , on handling noise and missing values
- **Scalability:** refers to the ability to construct the classifier or predictor efficiently given large amount of data.
- **Interpretability :** refers to the level of understanding and insight that is provided by the classifier or predictor.

Decision Tree Induction: Training Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



Classification by decision tree

- Decision tree induction is the learning of decision from class –labeled training tuples.
- A decision tree is a tree like structure where each internal node(non leaf node) denotes a test on an attribute , each branch represents an outcome of the test, and each leaf node a class label .
- The top most node in a tree is the root node.
- Internal node- rectangle
- Leaf node –oval

How decision tree used for classification:

- Given a tuple X for which the associated class label is unknown , the attribute value of the tuple are tested against the decision tree.
- A path is traced from the root to a leaf node , which hold the class prediction for that tuple. Decision tree can easily be converted to classification rules.
- The decision tree induction is so popular because the construction of decision tree classifiers does not require any domain knowledge or parameter setting.

- The benefits of having a decision are as follows:
  - It does not require any domain knowledge
  - It is easy to comprehend
  - The learning and classification steps of a decision tree are simple and fast.

- Decision tree generation consists of two phases
- **Tree construction** – At start all the training examples are at the root. Partition examples recursively based on selected attributes.
- **Tree pruning**- Identify and remove branches that reflect noise or outliers.
- **Use of decision tree**- Classifying an unknown sample. Also test the attribute values of the sample against decision tree.

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Computing Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_A(D) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$
  - gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini index (CART, IBM IntelligentMiner)

- If a data set *D* contains examples from *n* classes, gini index, *gini*(*D*) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  where p_j is the relative frequency of class *j* in *D*
- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Overfitting and Tree Pruning

- **Overfitting:** An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - **Prepruning:** Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - **Postpruning:** Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Bayesian Classification: Why?

- **A statistical classifier:** performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- **Foundation:** Based on Bayes' Theorem.
- **Performance:** A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- **Incremental:** Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- **Standard:** Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (*posteriori probability*), the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

NAIVE BAYES CLASSIFIER

- Naive Bayes is a kind of classifier which uses the Bayes Theorem.
- It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class.
- The class with the highest probability is considered as the most likely class.
- This is also known as Maximum A Posteriori (P(Ci|X).

This can be derived from Bayes' theorem

$$P(C_i|X) = P(X|C_i)P(C_i)/P(X)$$

Assumption

Naive Bayes classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature.

- Let's consider a training dataset with 1500 records and 3 classes. We presume that there are no missing values in our data. We have We have 3 classes associated with Animal Types:
  - Parrot,
  - Dog,
  - Fish.

The Predictor features set consists of 4 features as

●Swim ●Wings ●GreenColor ●Dangerous Teeth.

All the features are categorical variables with either of the 2 values: T(True) or F(False).

Swim	Wings	Green Color	Dangerous Teeth	Animal Type
50	500/500	400/500	0	Parrot
450/500	0	0	500/500	Dog
500/500	0	100/500	50/500	Fish

The above table shows a frequency table of our data. In our training data:

- Parrots have 50(10%) value for Swim, i.e., 10% parrot can swim according to our data, 500 out of 500(100%) parrots have wings, 400 out of 500(80%) parrots are Green and 0(0%) parrots have Dangerous Teeth.
- Classes with Animal type Dogs shows that 450 out of 500(90%) can swim, 0(0%) dogs have wings, 0(0%) dogs are of Green color and 500 out of 500(100%) dogs have Dangerous Teeth.
- Classes with Animal type Fishes shows that 500 out of 500(100%) can swim, 0(0%) fishes have wings, 100(20%) fishes are of Green color and 50 out of 500(10%) Fishes have Dangerous Teeth.

- Now, it's time to work on predict classes using the Naive Bayes model. We have taken 2 records that have values in their feature set, but the target variable needs to predicted

	Swim	Wings	Green	Teeth
1.	True	False	True	False
2.	True	False	True	True

We have to predict animal type using the feature values. We have to predict whether the animal is a Dog, a Parrot or a Fish

Let's consider the first record.

The Evidence here is Swim & Green. The Hypothesis can be an animal type to be Dog, Parrot, Fish.

For Hypothesis testing for the animal to be a Dog:

P(Dog | Swim, Green) = P(Swim|Dog) * P(Green|Dog) * P(Dog) / P(Swim, Green)

= 0.9 * 0 * 0.333 / P(Swim, Green)

For Hypothesis testing for the animal to be a Parrot:

P(Parrot| Swim, Green) = P(Swim|Parrot) * P(Green|Parrot) * P(Parrot) / P(Swim, Green)

= 0.1 * 0.80 * 0.333 / P(Swim, Green)

= 0.0264/ P(Swim, Green)

For Hypothesis testing for the animal to be a Fish:

P(Fish| Swim, Green) = P(Swim|Fish) * P(Green|Fish) * P(Fish) / P(Swim, Green)

= 1 * 0.2 * 0.333 / P(Swim, Green)

= 0.0666/ P(Swim, Green)

The denominator of all the above calculations is same i.e, P(Swim, Green).
The value of P(Fish| Swim, Green) is greater that P(Parrot| Swim, Green).

Using Naive Bayes, we can predict that the class of this record is Fish.

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

  R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

  - Rule antecedent/precondition vs. rule consequent

- Assessment of a rule: *coverage* and *accuracy*

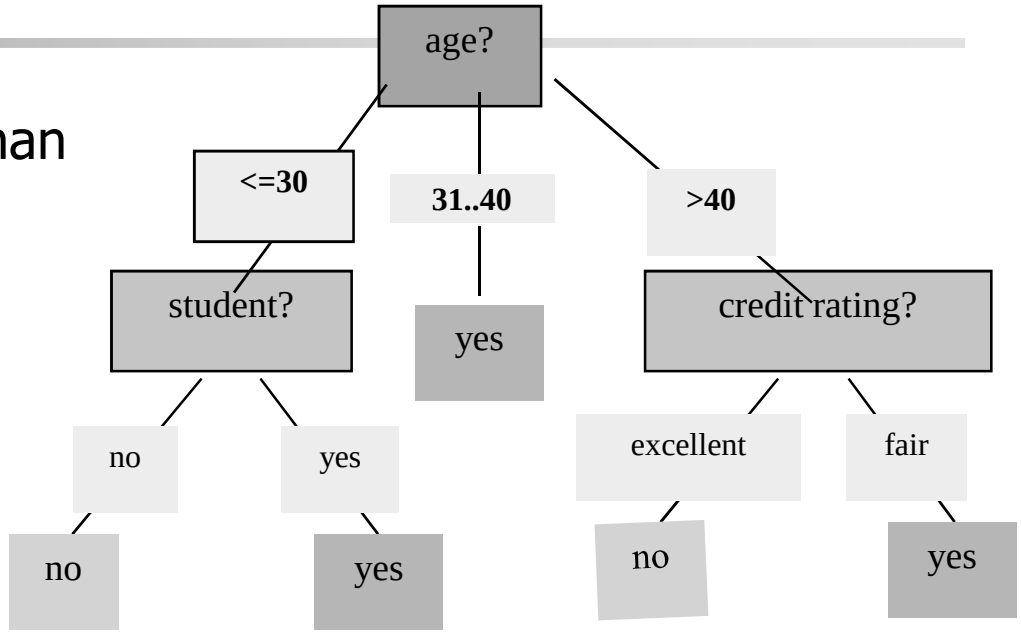
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

  coverage(R) = n_{covers} / |D| /* D: training data set */

  accuracy(R) = $n_{correct}$ / n_{covers}

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Rules are mutually exclusive and exhaustive



- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

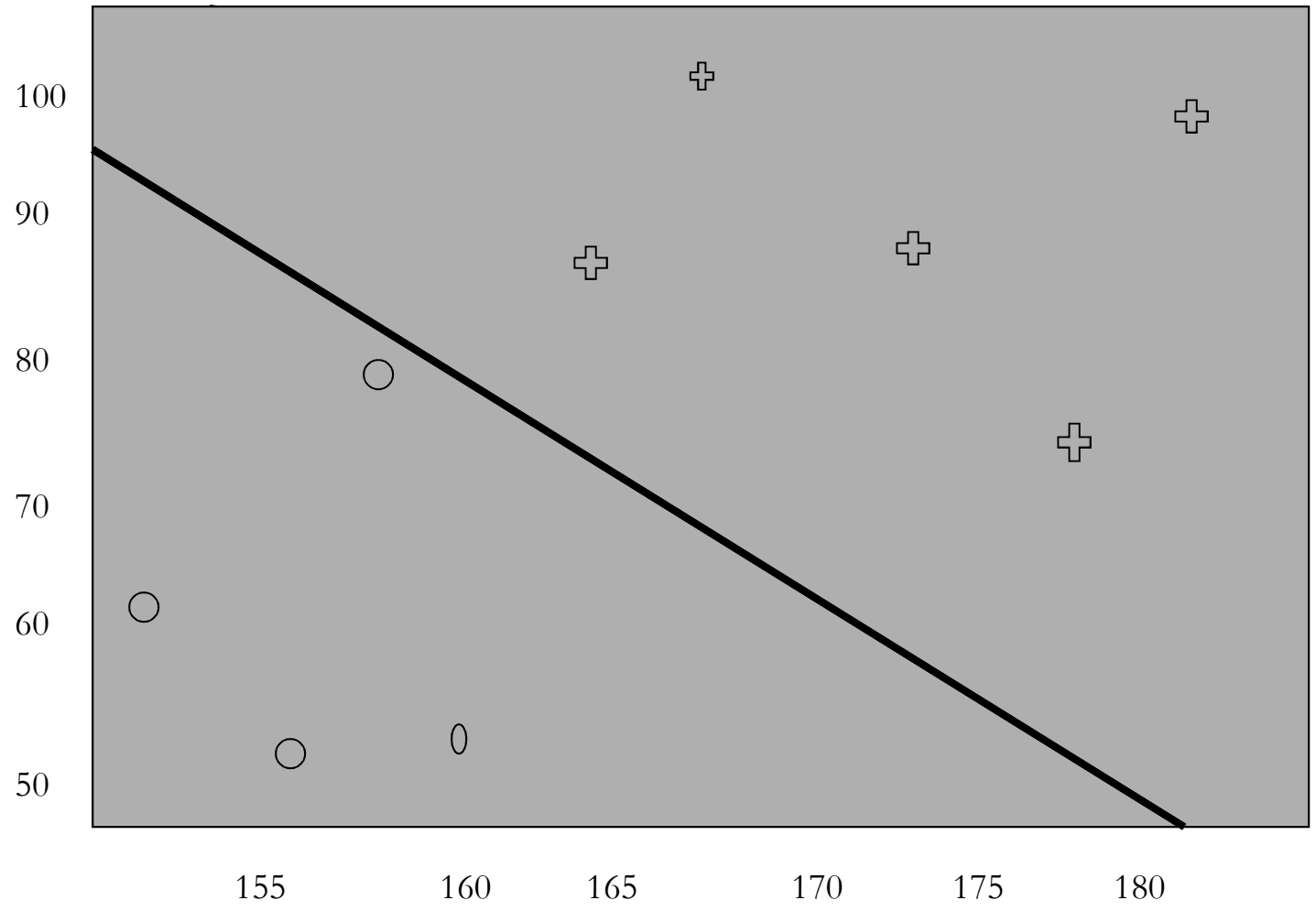
IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

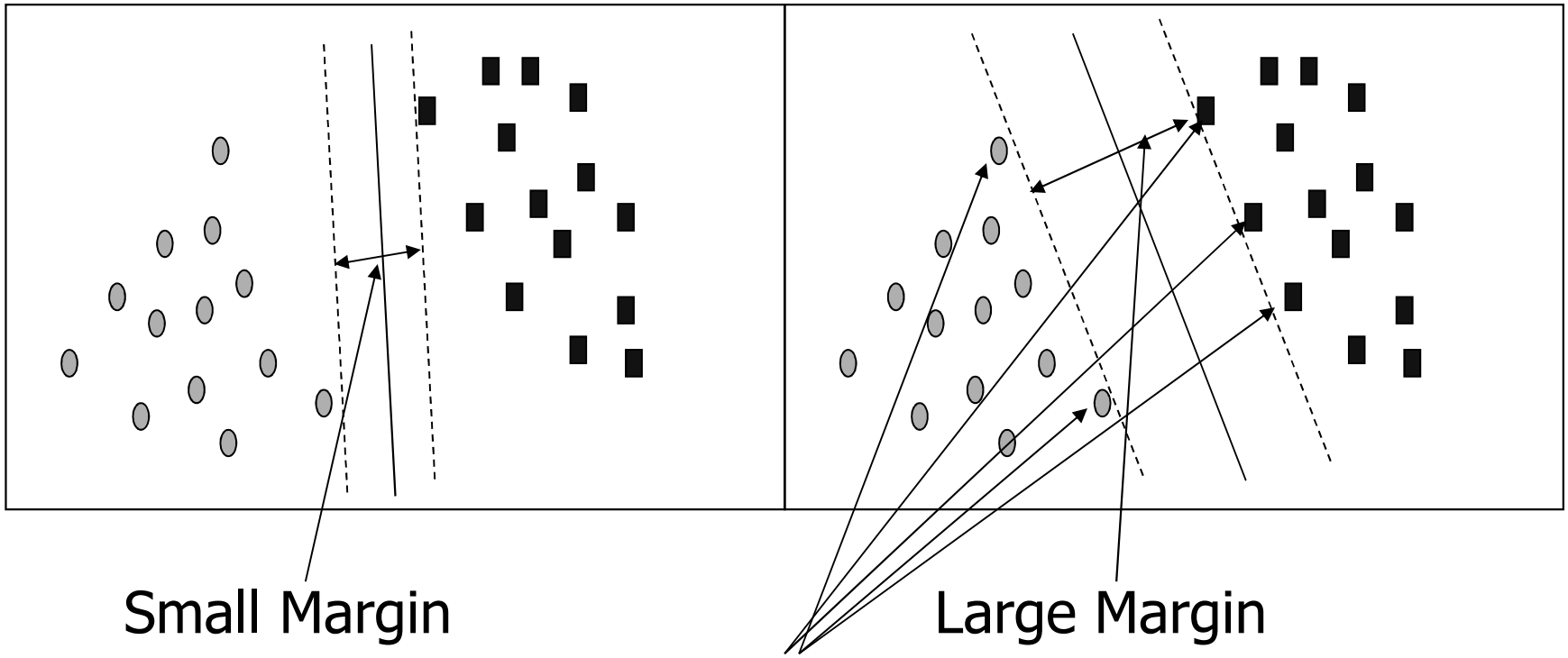
IF *age* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*

SVM—Support Vector Machines

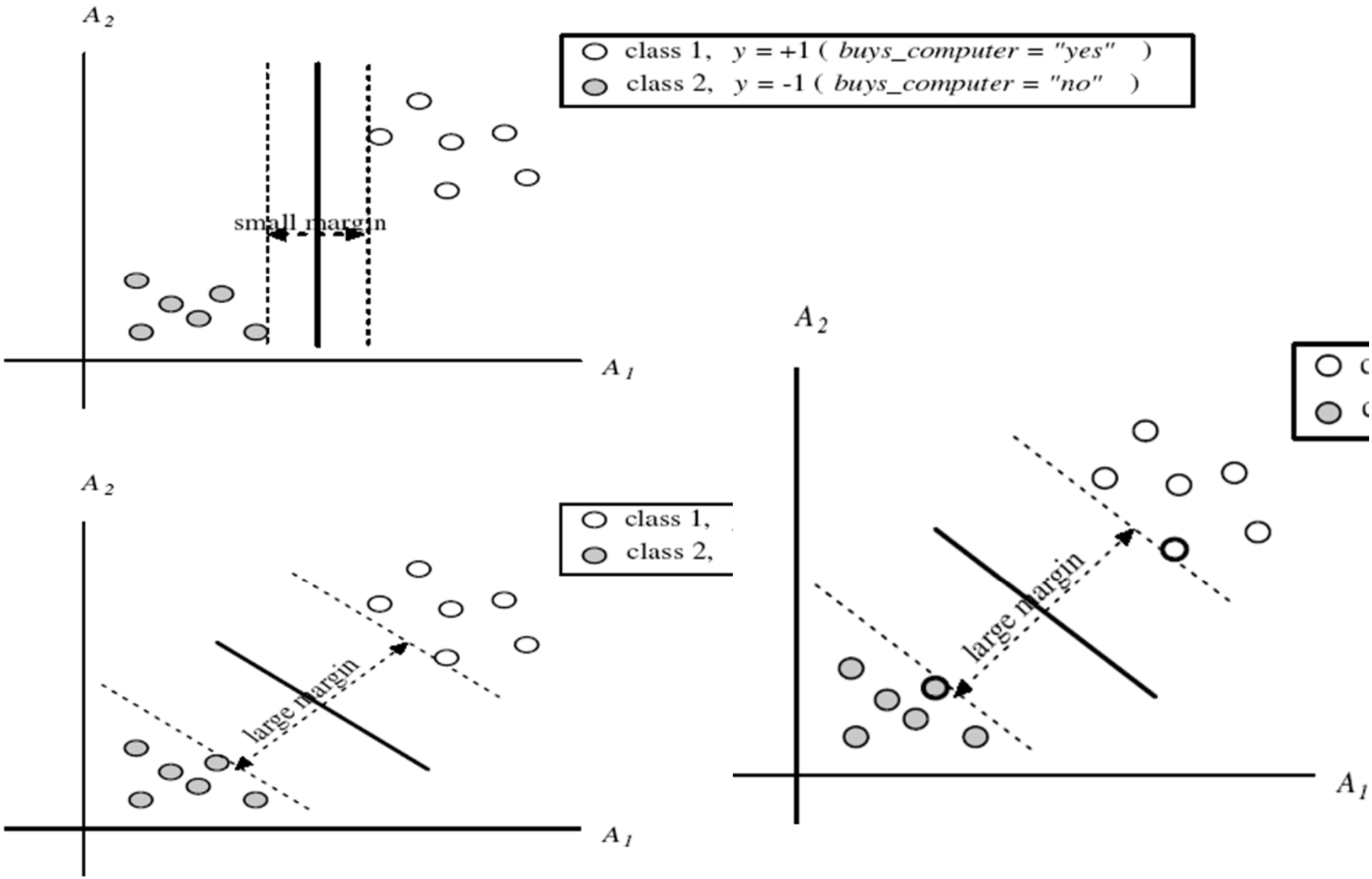
- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)


100
90
80
70
60
50
155
160
165
170
175
180

SVM—General Philosophy



SVM—Margins and Support Vectors



SVM—Linearly Separable

- SVM find the optimal separating hyperplane which maximizes the margin of the training data.
- A separating hyperplane can be written as

  $$\mathbf{W} \bullet \mathbf{X} + b = 0$$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

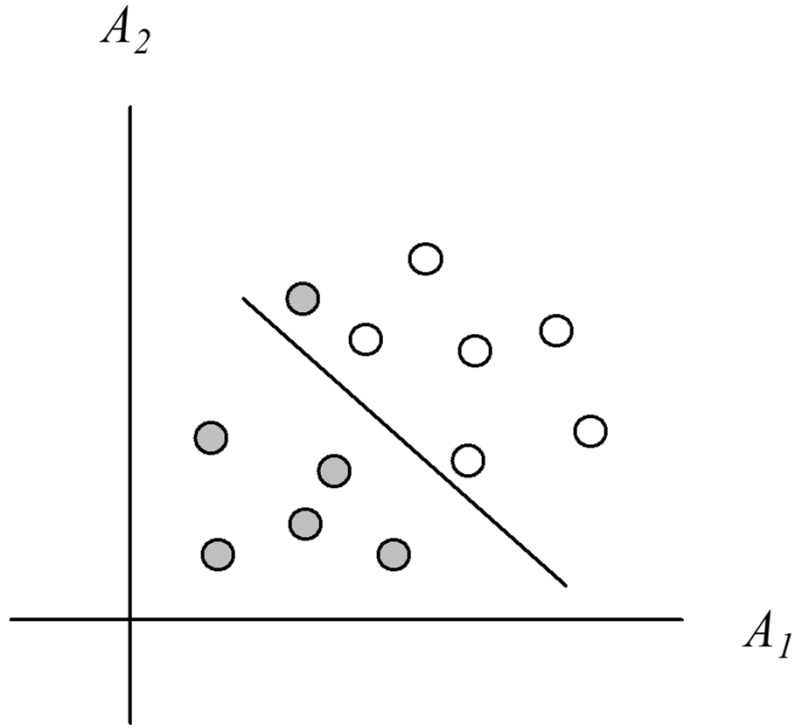
  $$w_0 + w_1 x_1 + w_2 x_2 = 0$$
- The hyperplane defining the sides of the margin:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**

SVM—Linearly Inseparable

- Transform the original input data into a higher dimensional space



- Search for a linear separating hyperplane in the new space

Lazy vs. Eager Learning

- Lazy vs. eager learning
  - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

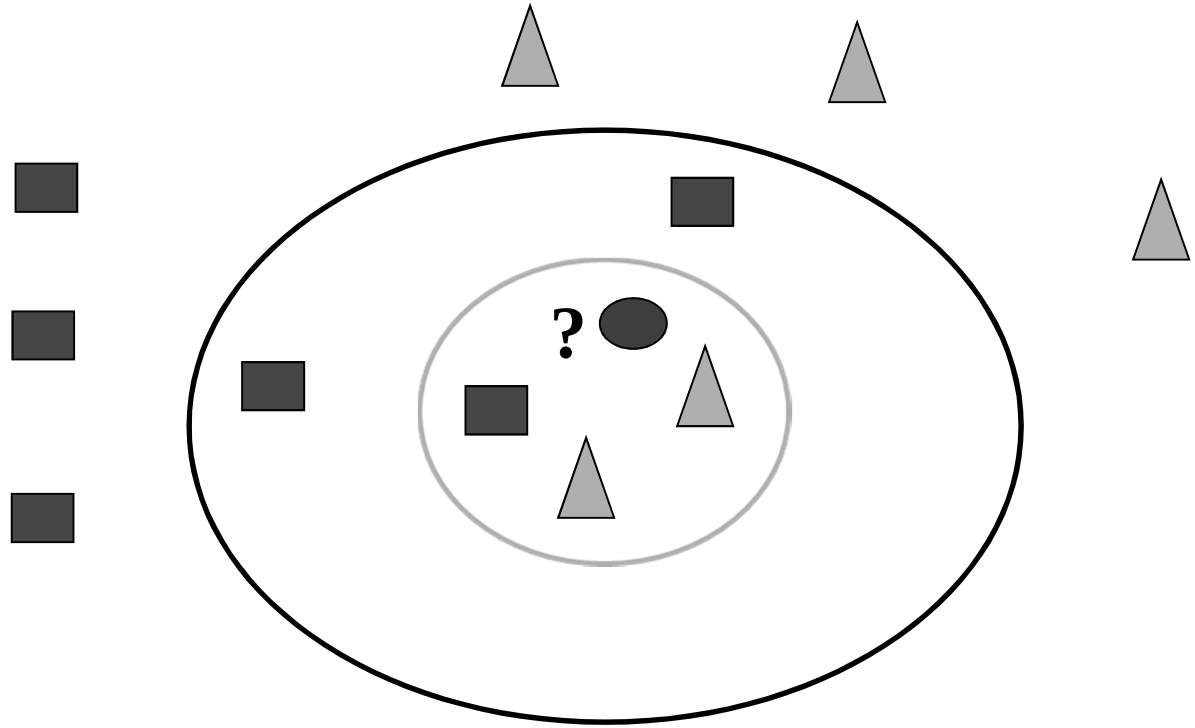
The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X}_1$, $\mathbf{X}_2$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q

k-NN Algorithm

1. Calculate "d(x,xi)" i=1,2,….,n; where d denotes the Euclidean distance between the points.
2. Arrange the calculated n Euclidean distances in non decreasing order.
3. Let k be a +ve integer, take the first k distances from this sorted list.
4. Find those k-points corresponding to these k-distances.
5. Let ki denotes the number of points belonging to the ith class among k points i.e.k>=0
6. If ki>kj for all i!=j, put x in class i.

Example



If k=3(green inner circle) it is assigned to triangles .

If k=5(yellow circle) it is assigned to squares

Choosing the right value for k

- Run KNN algorithm several times with different values of K and choose the K that reduces the number of errors .
- As we decrease the value of K to 1, our predictions become less stable.
- Inversely, as we increase the value of K, our predictions become more stable due to majority voting /averaging.
- An increase in errors occur when the value of K is too high.
- In cases to take a majority vote among labels make k an odd number to have a tiebreaker.

- Advantages
  - The algorithm is simple and easy to implement.
  - There's no need to build a model, tune several parameters, or make additional assumptions.
  - The algorithm is versatile. It can be used for classification, regression and search.
- Disadvantages
  - The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

What Is Prediction?

- (Numerical) prediction is similar to classification
  - construct a model using training set.
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear regression
  - Multiple regression
  - Non-linear regression

Linear Regression

- Linear regression: involves a response variable y and a single predictor variable x

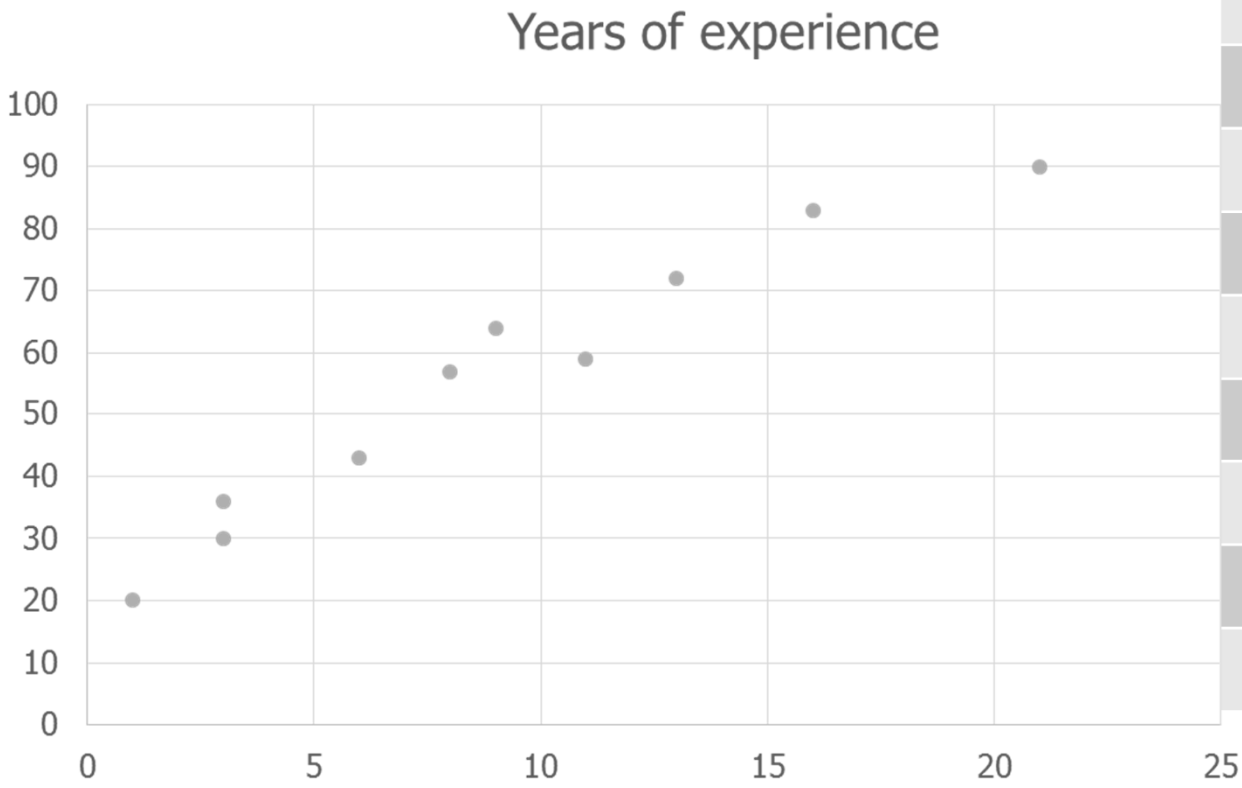
  $y = w_0 + w_1 x$

  where w_0 (y-intercept) and w_1 (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line as the one that minimizes the error between the actual data and estimate of the line.

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- Example
- The given 2 D data can be graphed on a scatter plot as follows:



X years experience	Y salary(in 1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

- The plot suggests a linear relationship between the two variables, x and y. model the relationship of salary with number of years of work with the equation y=w0+w1x
- mean of x=9.1 and mean of y =55.4 in case of above data.
- Substitute in above equation w1=3.5 and w0=23.6
- Equation of the least squares line is estimated by

$$y=23.6+3.5x$$

- We can predict the salary of a person with 10 years of experience is 58,600/- with above eqation

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1)$, $(\mathbf{X_2}, y_2)$,…, $(\mathbf{X_{|D|}}, y_{|D|})$
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method

Nonlinear Regression

- Many nonlinear functions can be transformed into the above equation.
- Some nonlinear models can be modeled by a polynomial function.
- A polynomial regression model can be transformed into linear regression model. For example,

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

convertible to linear with new variables: $x_2 = x^2$, $x_3 = x^3$

$$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

This can be solved using method of least squares