OLAP and Data Mining

- OLTP (On Line Transaction Processing)
- OLAP (On Line Analytic Processing)
- Data Ming
- Data Warehouse

1 Introduction

- OLTP: On Line Transaction Processing
  - Maintains a database that is an accurate model of some real-world enterprise.
  - Supports day-to-day operations.
  - Characteristics:
    * Short simple transactions
    * Relatively frequent updates
    * Transactions access only a small fraction of the database Processing
  - TPC-benchmark

- OLAP: On Line Analytic Processing
  - Uses information in database to guide strategic decisions.
  - Characteristics:
    - Complex queries
    - Infrequent updates
    - Transactions access a large fraction of the database
    - Data need not be up-to-date

The Internet Grocer

- OLTP-style transaction:

  John Smith, from Schenectady, N.Y., just bought a box of tomatoes; charge his account; deliver the tomatoes from our Schenectady warehouse; decrease our inventory of tomatoes from that warehouse

- OLAP-style transaction: How many cases of tomatoes were sold in all northeast warehouses in the years 2000 and 2001?

OLAP: Traditional Compared with Newer Applications

- Traditional OLAP queries
  - Uses data the enterprise gathers in its usual activities, perhaps in its OLTP system.
  - Queries are ad hoc, perhaps designed and carried out by non-professionals (managers)
- Newer Applications (e.g., Internet companies)
  - Enterprise actively gathers data it wants, perhaps purchasing it
  - Queries are sophisticated, designed by professionals, and used in more sophisticated ways

The Internet Grocer

- Traditional
  - How many cases of tomatoes were sold in all northeast warehouses in the years 2000 and 2001?
- Newer
  - Prepare a profile of the grocery purchases of John Smith for the years 2000 and 2001 (so that we can customize our marketing to him and get more of his business)

Data Mining

- Data Mining (knowledge discovery) is to extract knowledge from a database
- Comparison between OLAP and Data Mining
  - OLAP

    What percentage of people who make over $50,000 defaulted on their mortgage in the year 2000?
  - Data Mining: How can information about salary, net worth, and other historical data be used to predict who will default on their mortgage?

Data Warehouses

- A data warehouse is the repository of the organization's historical data.
- OLAP and data mining databases are frequently stored on the data organization's warehouses:
  - Can accommodate the huge amount of data generated by OLTP systems
  - Allow OLAP queries and data mining to be run off-line so as not to impact the performance of OLTP

OLAP and Data Mining

- Our main interest is in the database aspects of these fields, not the sophisticated analysis techniques
- Many OLAP and Data Mining applications involve sophisticated analysis methods from the fields of mathematics, statistical analysis, and artificial intelligence

2 OLAP: On Line Analytic Process

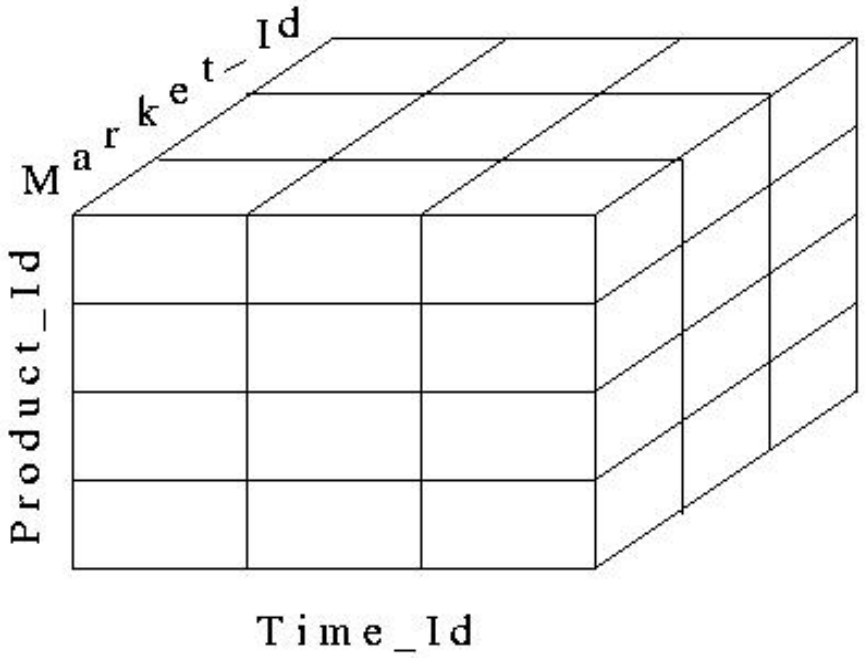
OLAP and Fact Tables

- Many OLAP applications are based on a fact table
- For example, a supermarket application might be based on a table

  Sales (Market_Id, Product_Id, Time_Id, Sales_Amt)
- The table can be viewed as multidimensional
  - Market_Id, Product_Id, Time_Id are the dimensions that represent specific supermarkets, products, and time intervals
  - Sales_Amt is a function of the other three

A Data Cube

- Fact tables can be viewed as an N-dimensional data cube (3-dimensional in our example)
  - The entries in the cube are the values for Sales_Amts



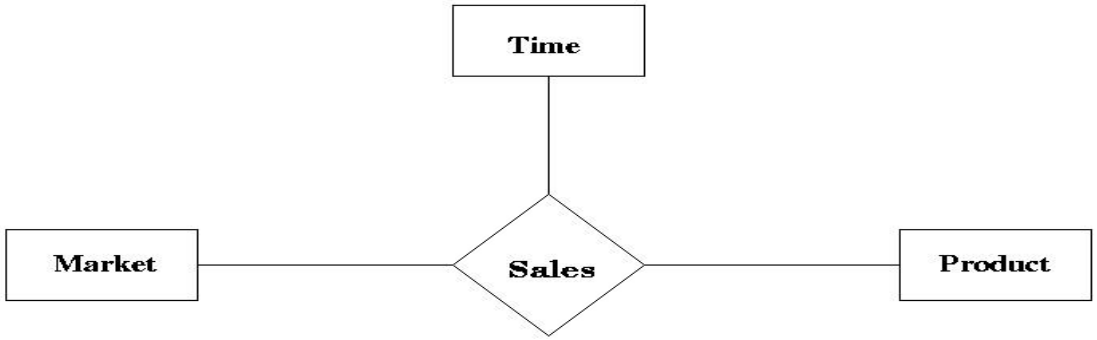
Dimension Tables

- The dimensions of the fact table are further described with dimension tables
- Fact table:

  Sales (Market_id, Product_Id, Time_Id, Sales_Amt)
- Dimension Tables:
  - Market (Market_Id, City, State, Region)
  - Product (Product_Id, Name, Category, Price)
  - Time (Time_Id, Week, Month, Quarter)

Star Schema

- The fact and dimension relations can be displayed in an E-R diagram, which looks like a star and is called a star schema



Aggregation

- Many OLAP queries involve aggregation of the data in the fact table
- For example, to find the total sales (over time) of each product in each market, we might use

```
SELECT    S.Market_Id, S.Product_Id, SUM (S.Sales_Amt)
FROM      Sales  S
GROUP BY  S.Market_Id, S.Product_Id
```

- The aggregation is over the entire time dimension and thus produces a two-dimensional view of the data. (Note: aggregation here is over time, not supermarkets or products.)

Aggregation over Time

- The output of the previous query

Market_Id

Product_Id	**SUM(Sales_Amt)**	M1	M2	M3	M4
	P1	3003	1503	...	...
	P2	6003	2402	...	...
	P3	4503	3	...	...
	P4	7503	7000	...	...
	P5	...	...	...	...

Drilling Down and Rolling Up

- Some dimension tables form an aggregation hierarchy

  Market_Id $->$ City $->$ State $->$ Region

- Executing a series of queries that moves down a hierarchy (e.g., from aggregation over regions to that over states) is called drilling down
  - Requires the use of the fact table or information more specific than the requested aggregation (e.g., cities)
- Executing a series of queries that moves up the hierarchy (e.g., from states to regions) is called rolling up
  - Note: In a rollup, coarser aggregations can be computed using prior queries for finer aggregations

Drilling Down

1. Drilling down on market: from Region to State

   Sales (Market_Id, Product_Id, Time_Id, Sales_Amt)

   Market (Market_Id, City, State, Region)

2.
```
SELECT   S.Product_Id, M.Region, SUM (S.Sales_Amt)
FROM     Sales  S,  Market  M
WHERE    M.Market_Id = S.Market_Id
GROUP BY S.Product_Id,  M.Region
```

3.
```
SELECT   S.Product_Id, M.State, SUM (S.Sales_Amt)
FROM     Sales  S,  Market  M
WHERE    M.Market_Id = S.Market_Id
GROUP BY S.Product_Id,  M.State,
```

Rolling Up

1 Rolling up on market, from State to Region

If we have already created a table, State_Sales, using

```
2. SELECT   S.Product_Id,  M.State, SUM (S.Sales_Amt)
   FROM     Sales  S,   Market  M
   WHERE    M.Market_Id = S.Market_Id
   GROUP BY S.Product_Id,  M.State
```

then we can roll up from there to:

```
3. SELECT        T.Product_Id,  M.Region, SUM (T.Sales_Amt)
   FROM         State_Sales  T,  Market  M
   WHERE        M.State = T.State
   GROUP BY T.Product_Id,  M.Region
```

Can reuse the results of query 1.

Pivoting

When we view the data as a multi-dimensional cube and group on a subset of the axes, we are said to be performing a pivot on those axes

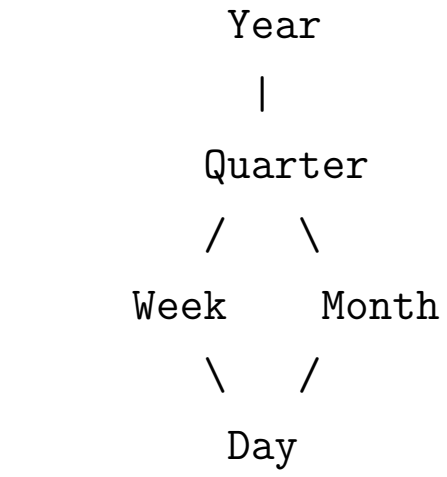
Pivoting on dimensions D1,..., Dk in a data cube
D1, ... ,Dk,Dk+1, ...,Dn means that we use
GROUP BY A1,...,Ak and aggregate over Ak+1,...An,
where Ai is an attribute of the dimension Di

Example: Pivoting on Product and Time corresponds to grouping on Product_id and Quarter and aggregating Sales_Amt over Market_id:

```
SELECT   S.Product_Id,  T.Quarter,  SUM (S.Sales_Amt)
FROM     Sales S,  Time T
WHERE    T.Time_Id = S.Time_Id
GROUP BY S.Product_Id,  T.Quarter
```

Time Hierarchy as a Lattice

- Not all aggregation hierarchies are liner
  - The time hierarchy is no lattice
    - Weeks are not contained in months
    - We can roll up days into weeks or months, but we can only roll up weeks into quarters



Slicing-and-Dicing

When we use WHERE to specify a particular value
for an axis (or several axes), we are performing a slice

Slicing the data cube in the Time dimension
(choosing sales only in week 12) then
pivoting to Product_id (aggregating over Market_id)

```
SELECT   S.Product_Id,  SUM (Sales_Amt)
FROM     Sales S, Time T
WHERE    T.Time_Id = S.Time_Id  AND  T.Week = Wk-12
GROUP BY S. Product_Id
```

Slicing-and-Dicing

Typically slicing and dicing involves several queries
to find the right slice.

For example, change the slice & the axes(from the prev. example)

* Slicing on Time and Market dimensions
* pivoting to Product_id and Week (in the time dimension)

```
SELECT   S.Product_Id,  T.Quarter,  SUM (Sales_Amt)
FROM     Sales S,  Time T
WHERE    T.Time_Id = S.Time_Id AND
         T.Quarter =  4  AND
         S.Market_id = 12345
GROUP BY S.Product_Id,  T.Week
```

The CUBE Operator

To construct the following table, would take 4 queries (next slide)

SUM(Sales_Amt)		Market_ID			Total
		M1	M2	M3	
	P1	3003	1503	15003	19509
Product_ID	P2	6003	2402	24003	32408
	P3	4503	3	33	4539
	P4	7503	7000	9903	24406
	Total	21012	10908	48942	80862

The Four Queries

For the table entries, without the totals (aggregation on time)

```
   SELECT    S.Market_Id,  S.Product_Id,  SUM (S.Sales_Amt)
   FROM      Sales S
   GROUP BY  S.Market_Id, S.Product_Id
```

For the row totals (aggregation on time and markets)

```
   SELECT    S.Product_Id,  SUM (S.Sales_Amt)
   FROM      Sales S
   GROUP BY  S.Product_Id
```

For the column totals (aggregation on time and products)

```
  SELECT     S.Market_Id,  SUM (S.Sales)
  FROM       Sales S
  GROUP BY   S.Market_Id
```

For the grand total (aggregation on time, markets, and products)

```
  SELECT     SUM (S.Sales)
  FROM       Sales S
```

Definition of the CUBE Operator

- Doing these three queries is wasteful
  - The first does much of the work of the other two: if we could save that result and aggregate over Market_Id and Product_Id, we could compute the other queries more efficiently
- The CUBE clause is part of SQL:1999

  GROUP BY CUBE (v1, v2, , vn)

  Equivalent to a collection of GROUP BYs, one for each of the 2^n subsets of v1, v2, ..., vn

Example of CUBE Operator

The following query returns all the information needed to make the previous products/markets table:

```
SELECT   S.Market_Id, S.Product_Id, SUM(S.Sales_Amt)
FROM     Sales S
GROUP BY CUBE (S.Market_Id, S.Product_Id)
```

ROLLUP

- ROLLUP is similar to CUBE except that instead of aggregating over all subsets of the arguments, it creates subsets moving from right to left
  - GROUP BY ROLLUP (A1,A2,,An) is a series of these aggregations:
  - GROUP BY A1 ,, An-1 ,An
  - GROUP BY A1 ,, An-1
  - ...
  - GROUP BY A1, A2
  - GROUP BY A1
  - No GROUP BY
- ROLLUP is also in SQL:1999

Example of ROLLUP Operator

```
SELECT   S.Market_Id,S.Product_Id,SUM(S.Sales_Amt)
FROM     Sales S
GROUP BY ROLLUP(S.Market_Id,S.Product_Id)

first aggregates with the finest granularity:
    GROUP BY S.Market_Id,S.Product_Id
then with the next level of granularity:
    GROUP BY S.Market_Id
then the grand total is computed with no GROUP BY clause
```

ROLLUP vs. CUBE

- The same query with CUBE:
  - first aggregates with the finest granularity:

    ```
    GROUP BY S.Market_Id,S.Product_Id
    ```

  - then with the next level of granularity:

    ```
    GROUP BY S.Market_Id
    ```

    and

    ```
    GROUP BY S.Product_Id
    ```

  - then the grand total with no GROUP BY

- Can you write a single SQL query, without using GROUP BY CUBE, that is equivalent to the following query?

```
SELECT   S.Market_Id, S.Product_Id, SUM(S.Sales_Amt)
FROM     Sales S
GROUP BY CUBE (S.Market_Id, S.Product_Id)
```

- Can you use SQL query facilities, without using GROUP BY CUBE, to retrieve the same result table as the following one with comparable evaluation performance?

```
SELECT   S.Market_Id, S.Product_Id, SUM(S.Sales_Amt)
FROM     Sales S
GROUP BY CUBE (S.Market_Id, S.Product_Id)
```

Materialized Views

The CUBE operator is often used to PRE-compute aggregations on all dimensions of a fact table and then save them as a materialized views to speed up future queries

Implementation Issues

- OLAP applications are characterized by a very large amount of data that is relatively static, with infrequent updates
  - Thus, various aggregations can be PRE-computed and stored in the database
  - Star joins, join indices, and bitmap indices can be used to improve efficiency (recall the methods to compute star joins in Chapter 14)
  - Since updates are infrequent, the inefficiencies associated with updates are minimized

3 Classification of Data Mining Techniques

Data Mining, also known as knowledge discovery, is a process of nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases.

Applications of the discovered knowledge

- information management
- query processing
- decision making
- process control
- ...

Challenges of Data Mining

- Mining information from different types/sources of data
- Efficiency and scalability of data mining algorithms
- Interesting and useful results
- Protection of privacy and data security

Classification of Data Mining Techniques

- Based on data sources
  - relational data miner
  - object-oriented data miner
  - Internet-information miner
  - others: deductive databases, spatial databases, temporal databases, multimedia databases, etc.
- Based on knowledge to be mined
  - association rules
  - characteristic rules
  - classification rules
  - clustering, ...
- Based on techniques utilized

  data-driving, query-driving, statistics-based, mathematics-based, ...

4 Mining Association Rules

To discover important associations among items such that the presence of some items in a transaction will likely imply the presence of other items in the same transaction.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items,
$D = \{T_1, T_2, \ldots, T_m\}$ be a set of transactions such that $T_i \subseteq I$,
$X \subseteq I$ and $Y \subseteq I$ are sets of items

An *association rule* is an implication of the form

$$X \Rightarrow Y$$

where $X \cap Y = \emptyset$.
The rule $X \Rightarrow Y$

- holds in D with *confidence* c if $c\%$ of transactions in D that contain X also contain Y;
- has *support* s in D if $s\%$ of transactions in D contain $X \cup Y$.

Strong Rules:

Association rules with high confidence and strong supports.

The task of mining association rules is essentially to discover strong association rules in large databases:

Methods:

- Discover the large itemsets, i.e. the sets of itemsets that have transaction support above a pre-determined minimum support s.
- Use the large itemsets to generate the association rules for the database.

The second step is rather straightforward.

Iterative algorithms for finding large itemsets

- Apriori

Step 1: finding the set L_1 of all itemsets of size 1 with support s

Step 2: finding the set L_2 of all itemsets of size 2 with support s based on L_1

Step 3: finding the set L_3 of all itemsets of size 3 with support s based on L_2

...

Example 1 Consider database D below and the supporting factor 40%

TID	Item
100	A C D
200	B C E
300	A B C E
400	B E

Thus, $s = 2$.

Step 1: C_1

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

$\Rightarrow$ L_1

Itemset	Sup.
{A}	2
{B}	3
{C}	3
{E}	3

Step 2: C_2

Itemset	Sup.
{A B}	1
{A C}	2
{A E}	1
{B C}	2
{B E}	3
{C E}	2

$\Rightarrow$ L_2

Itemset	Sup.
{A C}	2
{B C}	2
{B E}	3
{C E}	2

Step 3: C_3

Itemset	Sup.
{B C E}	2

$\Rightarrow$ L_3

Itemset	Sup.
{B C E}	2

Note that $C_{k+1} = \{ (I \mid I \in L_k * L_k) \wedge (I' \in L_k \quad \text{for all} \quad I' \subset I)\}$

Interestingness of Discovered Rules

Not all the discovered strong rules are interesting.

Example 2 Consider a school of 5000 students:

- 3000 play basketball (60%)
- 3750 eat cereal (75%)
- 2000 play basketball and eat cereal (40%)

and the rule (Play basketball) $\Rightarrow$ (Eat cereal) with 40% support and 67% confidence.
The rule is strong because the confident is $\frac{2000}{3000} = 67\%$ while the support is 40%.
But the rule is misleading for 75% of students eat cereal while only 67% of those who play basketball eat cereal.

A rule $A \Rightarrow B$ is interesting only if its confidence exceeds a certain measure.

- For some suitable constant d,

$$\frac{P(A \cap B)}{P(A)} - P(B) > d$$

  Notions of interestingness and/or usefulness of discovered rules.

Efficiency of mining association rules

- database scan reduction
- sampling: mining with adjustable accuracy
- incremental updating of discovered association rules
- parallel data mining

5 Data Generalization

Data generalization is a process of abstracting a large set of relevant data in a database from a low concept level to relatively high levels.

- Data cube approach

  To materialize certain expensive computations that are frequently inquired and to store such materialized views in a multi-dimensional database, called data cube, for decision support, knowledge discovery, and other applications.

- Attribute-oriented induction approach

  Data generalization is performed on the set of relevant data collected using an SQL-like query language.

Data cube approach (See OLTP)

Example 3 Consider a relation with the schema

Sales(part, supplier, customer, sale_price)

Then eight different views can be extracted:

- psc: aggregate function values (such as total_sales) group by three attributes part, supplier, customer.

  select part, supplier, customer, total(sale_price)
  from Sales
  group by part, supplier, customer

- pc: group by part, customer
- sc, ps, p, s, c
- none: total sales_price

Attribute-oriented induction approach

Data generalization is performed on the set of relevant data collected **On-Line** using an SQL-like query language.

- examining the data distribution for each attribute in the set of relevant data
- calculating the corresponding abstract level to be generalized, and
- replacing each data tuple with its generalized value

Background knowledge: Concept hierarchy associated with each attribute, stored implicitly in databases

- For example, a set of attributes in

  address(number, street, city, province, country)

  indicates a hierarchy.

6 Data Classification

Data classification is a process which finds the common properties among a set of objects in a database and classifies them into different classes, according to a *classification model*.

How to construct a classification model ?

- training set: a sample database such that
  - each tuple in the training set consists of the same set of multiple attributes as in a large database, and
  - each tuple is associated with a known class identify (label).
- analyze the training data and develop an accurate description for each class
- classify test data in the large database

[1] Classification based on decision trees

A supervised machine learning method that constructs decisions trees from a set of examples.

Given

- a set of objects that are described in terms of attribute values, and
- a *training set* of objects whose class is known.

The induction task is to develop a classification rule, based on the training set, that can determine the class of any object from its value of attributes.

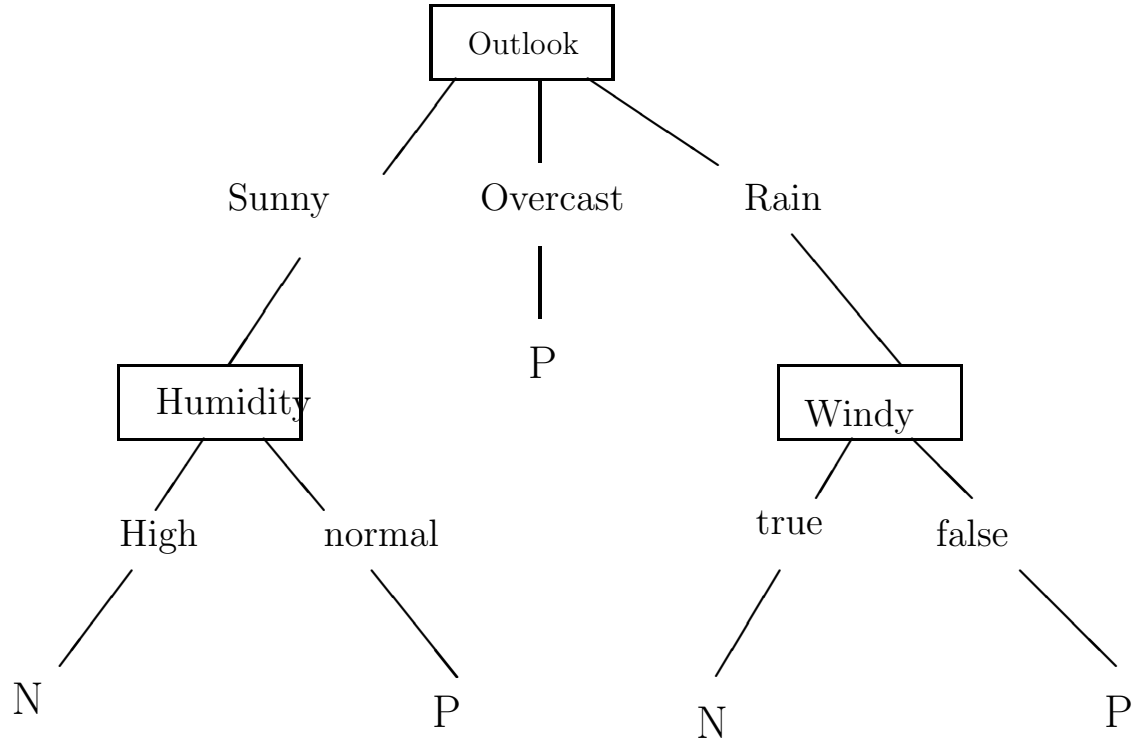
A decision tree is defined/constructed as such a classification rule.

Formally, a decision tree is either

- a leaf node that contains a class name, or
- a non-leaf node (or, decision node) that contains an attribute test with a branch to another decision tree for each possible value of the attribute.

Example 4 Consider the following training set:

No	Outlook	Temperature	Humidity	Windy	Class
1	sunny	hot	high	false	N
2	sunny	hot	high	false	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N



Now consider a Saturday morning with the following attribute values:

Outlook: overcast, Temperature: cool Humidity: normal Windy: false

What is the class of this Saturday?

- How many decision trees are there for a given training set ?

  Finite but very large.

- The simpler, the better ?

- How to find the simplest one ?

  Generate all decision trees and then choose the one you like.

- Accuracy?

- ID3 Algorithm

ID3 Algorithm

An iterative approach

- randomly select a subset of training set: window
- generate a decision tree from the window
  - if the tree correctly classify all the objects in the training set, then output the tree
  - otherwise, expand the window by adding incorrectly classified objects into the window and continue the process

Challenges

- How to form a decision tree from an arbitrary collection of objects ?

  Not difficult: select an attribute for the root test that provides a non-trivial partition of the given set of objects.

- How to choose the root attribute such that the decision tree is simple ?

  It is difficult but interesting

Information-based approach

Let C contain p objects of class P and n objects of class N. We assume

- Any correct decision tree for C will classify objects in the same proportion as their representation in C.

  Thus, an arbitrary object will belong to P with probability $\frac{p}{p+n}$ and N $\frac{n}{p+n}$.

- A decision tree can be regarded as a course of a message P or N, with the expected information needed to generate this message given by

$$I(p,n) = -\frac{p}{p+n}log_2\frac{p}{p+n} - \frac{n}{p+n}log_2\frac{n}{p+n}$$

If attribute A with values $\{A_1, A_2, \ldots, A_v\}$ is used for the root, it will partition C into $\{C_1, C_2, \ldots, C_v\}$ where C_i contains objects that have value A_i. The the expected information required for the subtree C_i is $I(p_i, n_i)$.
Therefore, the information required for the tree with A as the root is then

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

The information gained by branching on A is therefore

$$gain(A) = I(p,n) - E(A)$$

How to choose the root attribute ?

Choose the attribute to branch on which gains the most information.

Example 5 (Example **??** continued)

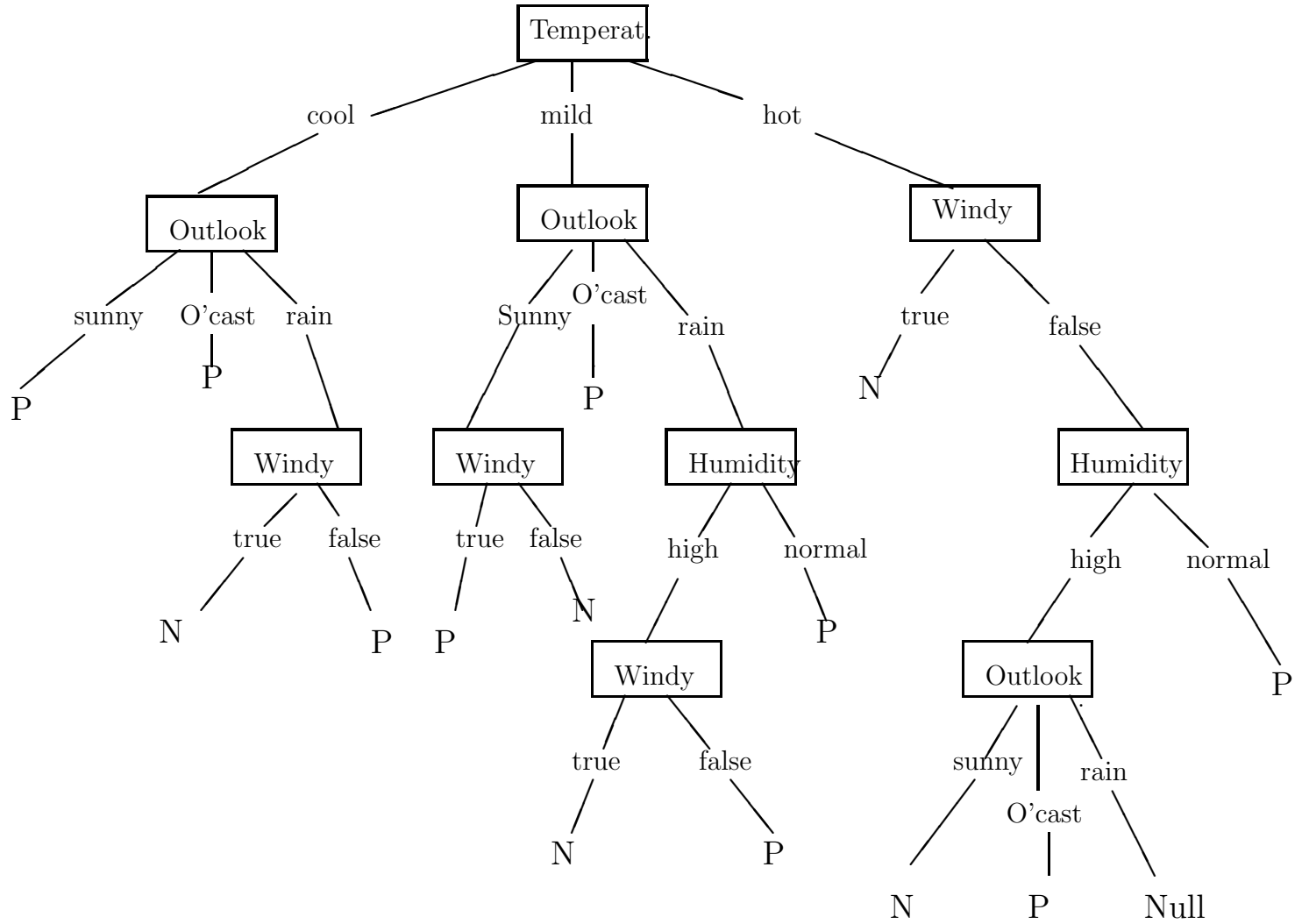
Of the 14 objects, 9 are in P and 5 in N, so the information required for classification is

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

The following table shows the expected information required and gain for each attribute to be the root.

Attribute	**Information required**	**Information gain**
Outlook	0.694	0.246
Temperature	0.911	0.029
Humidity	0.789	0.151
Windy	0.892	0.048

Obviously, by the information gain, Overlook is the best choice for the root.



A complex decision tree

Accuracy of decision trees

The accuracy of an algorithm for decision trees is usually determined

- use only part of the given set of objects as a training set, and
- check the resulting decision on the remainder.

Some experiments

- 1.4 million chess positions described in terms of 49 binary-valued attributes gave rise to 715 distinct objects divided 65%:35% between the classes. A decision tree contains about 150 nodes.

  The decision tree based on a training set that contains randomly selected 20% of these 715 objects correctly classified 84% of unseen objects.

- Another similar example: the decision tree based on a training set of 20% objects correctly classified 98% unseen objects.

[2] Other methods for data classification

- Statistical approaches
  - linear regression
  - liner discriminant analysis
- rough sets approach
- interval classifier
- neural network approach

[3] Methods for performance improvement

? Scaling up

7 Clustering Analysis

Clustering analysis, also called *unsupervised classification*, is a process of grouping physical or abstract objects into classes of similar objects.
It has been studied in

- statistics
- machine learning
- data mining

  identify clusters, i.e., densely populated regions in a large, multidimensional database, according to some distance measurement.