Chapter 11: Data Analytics
Chapter 11: Data Analytics

- Overview
- Data Warehousing
- Online Analytical Processing
- Data Mining
Overview

- **Data analytics**: the processing of data to infer patterns, correlations, or models for prediction

- Primarily used to make business decisions
  - Per individual customer
    - E.g. what product to suggest for purchase
  - Across all customers
    - E.g. what products to manufacture/stock, in what quantity

- Critical for businesses today
Overview (Cont.)

- Common steps in data analytics
  - Gather data from multiple sources into one location
    - Data warehouses also integrated data into common schema
    - Data often needs to be \textit{extracted} from source formats, \textit{transformed} to common schema, and \textit{loaded} into the data warehouse
      - Can be done as \textit{ETL (extract-transform-load)}, or \textit{ELT (extract-load-transform)}
  - Generate aggregates and reports summarizing data
    - Dashboards showing graphical charts/reports
    - \textbf{Online analytical processing (OLAP) systems} allow interactive querying
    - Statistical analysis using tools such as R/SAS/SPSS
      - Including extensions for parallel processing of big data
  - Build \textbf{predictive models} and use the models for decision making
Predictive models are widely used today

- E.g. use customer profile features (e.g. income, age, gender, education, employment) and past history of a customer to predict likelihood of default on loan
  - and use prediction to make loan decision
- E.g. use past history of sales (by season) to predict future sales
  - And use it to decide what/how much to produce/stock
  - And to target customers

Other examples of business decisions:

- What items to stock?
- What insurance premium to change?
- To whom to send advertisements?
Overview (Cont.)

- **Machine learning** techniques are key to finding patterns in data and making predictions.
- **Data mining** extends techniques developed by machine-learning communities to run them on very large datasets.
- The term **business intelligence (BI)** is synonym for data analytics.
- The term **decision support** focuses on reporting and aggregation.
DATA WAREHOUSING
Data Warehousing

- Data sources often store only current data, not historical data
- Corporate decision making requires a unified view of all organizational data, including historical data
- A data warehouse is a repository (archive) of information gathered from multiple sources, stored under a unified schema, at a single site
  - Greatly simplifies querying, permits study of historical trends
  - Shifts decision support query load away from transaction processing systems
Data Warehousing

- Data source 1
- Data source 2
- ... (ellipsis)
- Data source n

Data loaders

DBMS

Data warehouse

Query and analysis tools
Design Issues

- **When and how to gather data**
  - **Source driven architecture**: data sources transmit new information to warehouse
    - either continuously or periodically (e.g. at night)
  - **Destination driven architecture**: warehouse periodically requests new information from data sources
  - **Synchronous vs asynchronous replication**
    - Keeping warehouse exactly synchronized with data sources (e.g. using two-phase commit) is often too expensive
    - Usually OK to have slightly out-of-date data at warehouse
    - Data/updates are periodically downloaded form online transaction processing (OLTP) systems.

- **What schema to use**
  - Schema integration
More Warehouse Design Issues

- **Data transformation and data cleansing**
  - E.g. correct mistakes in addresses (misspellings, zip code errors)
  - Merge address lists from different sources and purge duplicates

- **How to propagate updates**
  - Warehouse schema may be a (materialized) view of schema from data sources
    - View maintenance

- **What data to summarize**
  - Raw data may be too large to store on-line
  - Aggregate values (totals/subtotals) often suffice
  - Queries on raw data can often be transformed by query optimizer to use aggregate values
Multidimensional Data and Warehouse Schemas

- Data in warehouses can usually be divided into
  - **Fact tables**, which are large
    - E.g. `sales(item_id, store_id, customer_id, date, number, price)`
  - **Dimension tables**, which are relatively small
    - Store extra information about stores, items, etc

Attributes of fact tables can be usually viewed as

- **Measure attributes**
  - measure some value, and can be aggregated upon
  - e.g., the attributes `number` or `price` of the `sales` relation

- **Dimension attributes**
  - dimensions on which measure attributes are viewed
  - e.g., attributes `item_id`, `color`, and `size` of the `sales` relation
  - Usually small ids that are foreign keys to dimension tables
Data Warehouse Schema

item_info
- item_id
- itemname
- color
- size
- category

store
- store_id
- city
- state
- country

date_info
- date
- month
- quarter
- year

sales
- item_id
- store_id
- customer_id
- date
- number
- price

customer
- customer_id
- name
- street
- city
- state
- zipcode
- country
Multidimensional Data and Warehouse Schemas

- Resultant schema is called a **star schema**
  - More complicated schema structures
    - **Snowflake schema**: multiple levels of dimension tables
    - May have multiple fact tables

- Typically
  - fact table joined with dimension tables and then
  - group-by on dimension table attributes, and then
  - aggregation on measure attributes of fact table

- Some applications do not find it worthwhile to bring data to a common schema
  - **Data lakes** are repositories which allow data to be stored in multiple formats, without schema integration
  - Less upfront effort, but more effort during querying
Database Support for Data Warehouses

- Data in warehouses usually append only, not updated
  - Can avoid concurrency control overheads

- Data warehouses often use **column-oriented storage**
  - E.g. a sequence of sales tuples is stored as follows
    - Values of item_id attribute are stored as an array
    - Values of store_id attribute are stored as an array,
      - And so on
  - Arrays are compressed, reducing storage, IO and memory costs significantly
  - Queries can fetch only attributes that they care about, reducing IO and memory cost
  - More details in Section 13.6

- Data warehouses often use parallel storage and query processing infrastructure
  - Distributed file systems, Map-Reduce, Hive, …
OLAP
Data Analysis and OLAP

- Online Analytical Processing (OLAP)
  - Interactive analysis of data, allowing data to be summarized and viewed in different ways in an online fashion (with negligible delay)

- We use the following relation to illustrate OLAP concepts
  - sales (item_name, color, clothes_size, quantity)

  This is a simplified version of the sales fact table joined with the dimension tables, and many attributes removed (and some renamed)
**Example sales relation**

<table>
<thead>
<tr>
<th>item_name</th>
<th>color</th>
<th>clothes_size</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>dress</td>
<td>dark</td>
<td>small</td>
<td>2</td>
</tr>
<tr>
<td>dress</td>
<td>dark</td>
<td>medium</td>
<td>6</td>
</tr>
<tr>
<td>dress</td>
<td>dark</td>
<td>large</td>
<td>12</td>
</tr>
<tr>
<td>dress</td>
<td>pastel</td>
<td>small</td>
<td>4</td>
</tr>
<tr>
<td>dress</td>
<td>pastel</td>
<td>medium</td>
<td>3</td>
</tr>
<tr>
<td>dress</td>
<td>pastel</td>
<td>large</td>
<td>3</td>
</tr>
<tr>
<td>dress</td>
<td>white</td>
<td>small</td>
<td>2</td>
</tr>
<tr>
<td>dress</td>
<td>white</td>
<td>medium</td>
<td>3</td>
</tr>
<tr>
<td>dress</td>
<td>white</td>
<td>large</td>
<td>0</td>
</tr>
<tr>
<td>pants</td>
<td>dark</td>
<td>small</td>
<td>14</td>
</tr>
<tr>
<td>pants</td>
<td>dark</td>
<td>medium</td>
<td>6</td>
</tr>
<tr>
<td>pants</td>
<td>dark</td>
<td>large</td>
<td>0</td>
</tr>
<tr>
<td>pants</td>
<td>pastel</td>
<td>small</td>
<td>1</td>
</tr>
<tr>
<td>pants</td>
<td>pastel</td>
<td>medium</td>
<td>0</td>
</tr>
<tr>
<td>pants</td>
<td>pastel</td>
<td>large</td>
<td>1</td>
</tr>
<tr>
<td>pants</td>
<td>white</td>
<td>small</td>
<td>3</td>
</tr>
<tr>
<td>pants</td>
<td>white</td>
<td>medium</td>
<td>0</td>
</tr>
<tr>
<td>pants</td>
<td>white</td>
<td>large</td>
<td>2</td>
</tr>
<tr>
<td>shirt</td>
<td>dark</td>
<td>small</td>
<td>2</td>
</tr>
<tr>
<td>shirt</td>
<td>dark</td>
<td>medium</td>
<td>6</td>
</tr>
<tr>
<td>shirt</td>
<td>dark</td>
<td>large</td>
<td>6</td>
</tr>
<tr>
<td>shirt</td>
<td>pastel</td>
<td>small</td>
<td>4</td>
</tr>
<tr>
<td>shirt</td>
<td>pastel</td>
<td>medium</td>
<td>1</td>
</tr>
<tr>
<td>shirt</td>
<td>pastel</td>
<td>large</td>
<td>2</td>
</tr>
<tr>
<td>shirt</td>
<td>white</td>
<td>small</td>
<td>17</td>
</tr>
<tr>
<td>shirt</td>
<td>white</td>
<td>medium</td>
<td>1</td>
</tr>
<tr>
<td>shirt</td>
<td>white</td>
<td>large</td>
<td>10</td>
</tr>
<tr>
<td>skirt</td>
<td>dark</td>
<td>small</td>
<td>2</td>
</tr>
<tr>
<td>skirt</td>
<td>dark</td>
<td>medium</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The table above is an example of a **cross-tabulation (cross-tab)**, also referred to as a **pivot-table**.

- Values for one of the dimension attributes form the row headers.
- Values for another dimension attribute form the column headers.
- Other dimension attributes are listed on top.
- Values in individual cells are (aggregates of) the values of the dimension attributes that specify the cell.
A **data cube** is a multidimensional generalization of a cross-tab.

- Can have $n$ dimensions; we show 3 below.
- Cross-tabs can be used as views on a data cube.

### Example Data Cube

<table>
<thead>
<tr>
<th>Item Name</th>
<th>Skirt</th>
<th>Dress</th>
<th>Shirt</th>
<th>Pants</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark</td>
<td>8</td>
<td>20</td>
<td>14</td>
<td>20</td>
<td>62</td>
</tr>
<tr>
<td>Pastel</td>
<td>35</td>
<td>10</td>
<td>7</td>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>White</td>
<td>10</td>
<td>5</td>
<td>28</td>
<td>5</td>
<td>48</td>
</tr>
<tr>
<td>All</td>
<td>53</td>
<td>35</td>
<td>49</td>
<td>27</td>
<td>164</td>
</tr>
</tbody>
</table>

---

### Data Cube Dimensions

<table>
<thead>
<tr>
<th>Clothes Size</th>
<th>Skirt</th>
<th>Dress</th>
<th>Shirt</th>
<th>Pants</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>29</td>
</tr>
<tr>
<td>Medium</td>
<td>34</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td>Large</td>
<td>34</td>
<td>16</td>
<td>18</td>
<td>45</td>
<td>111</td>
</tr>
</tbody>
</table>
Online Analytical Processing Operations

- **Pivoting**: changing the dimensions used in a cross-tab
  - E.g. moving colors to column names

- **Slicing**: creating a cross-tab for fixed values only
  - E.g. fixing color to white and size to small
  - Sometimes called **dicing**, particularly when values for multiple dimensions are fixed.

- **Rollup**: moving from finer-granularity data to a coarser granularity
  - E.g. aggregating away an attribute
  - E.g. moving from aggregates by day to aggregates by month or year

- **Drill down**: The opposite operation - that of moving from coarser-granularity data to finer-granularity data
Hierarchies on Dimensions

- **Hierarchy** on dimension attributes: lets dimensions be viewed at different levels of detail
- E.g., the dimension *datetime* can be used to aggregate by hour of day, date, day of week, month, quarter or year

(a) time hierarchy
(b) location hierarchy
Cross Tabulation With Hierarchy

- Cross-tabs can be easily extended to deal with hierarchies
- Can drill down or roll up on a hierarchy
- E.g. hierarchy: item_name → category

clothes_size: all

<table>
<thead>
<tr>
<th>category</th>
<th>item_name</th>
<th>color</th>
<th></th>
<th></th>
<th></th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>dark</td>
<td>pastel</td>
<td>white</td>
<td></td>
<td></td>
</tr>
<tr>
<td>womenswear</td>
<td>skirt</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dress</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td>subtotal</td>
<td>28</td>
<td>28</td>
<td>15</td>
<td></td>
<td>88</td>
</tr>
<tr>
<td>menswear</td>
<td>pants</td>
<td>14</td>
<td>14</td>
<td>28</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>shirt</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>subtotal</td>
<td>34</td>
<td>34</td>
<td>33</td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>62</td>
<td>62</td>
<td>48</td>
<td></td>
<td>164</td>
</tr>
</tbody>
</table>
Relational Representation of Cross-tabs

- Cross-tabs can be represented as relations.
- We use the value \textit{all} to represent aggregates.
- The SQL standard actually uses \textit{null} values in place of \textit{all}.
  - Works with any data type.
  - But can cause confusion with regular null values.

<table>
<thead>
<tr>
<th>item_name</th>
<th>color</th>
<th>clothes_size</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>skirt</td>
<td>dark</td>
<td>all</td>
<td>8</td>
</tr>
<tr>
<td>skirt</td>
<td>pastel</td>
<td>all</td>
<td>35</td>
</tr>
<tr>
<td>skirt</td>
<td>white</td>
<td>all</td>
<td>10</td>
</tr>
<tr>
<td>skirt</td>
<td>all</td>
<td>all</td>
<td>53</td>
</tr>
<tr>
<td>dress</td>
<td>dark</td>
<td>all</td>
<td>20</td>
</tr>
<tr>
<td>dress</td>
<td>pastel</td>
<td>all</td>
<td>10</td>
</tr>
<tr>
<td>dress</td>
<td>white</td>
<td>all</td>
<td>5</td>
</tr>
<tr>
<td>dress</td>
<td>all</td>
<td>all</td>
<td>35</td>
</tr>
<tr>
<td>shirt</td>
<td>dark</td>
<td>all</td>
<td>14</td>
</tr>
<tr>
<td>shirt</td>
<td>pastel</td>
<td>all</td>
<td>7</td>
</tr>
<tr>
<td>shirt</td>
<td>white</td>
<td>all</td>
<td>28</td>
</tr>
<tr>
<td>shirt</td>
<td>all</td>
<td>all</td>
<td>49</td>
</tr>
<tr>
<td>pants</td>
<td>dark</td>
<td>all</td>
<td>20</td>
</tr>
<tr>
<td>pants</td>
<td>pastel</td>
<td>all</td>
<td>2</td>
</tr>
<tr>
<td>pants</td>
<td>white</td>
<td>all</td>
<td>5</td>
</tr>
<tr>
<td>pants</td>
<td>all</td>
<td>all</td>
<td>27</td>
</tr>
<tr>
<td>all</td>
<td>dark</td>
<td>all</td>
<td>62</td>
</tr>
<tr>
<td>all</td>
<td>pastel</td>
<td>all</td>
<td>54</td>
</tr>
<tr>
<td>all</td>
<td>white</td>
<td>all</td>
<td>48</td>
</tr>
<tr>
<td>all</td>
<td>all</td>
<td>all</td>
<td>164</td>
</tr>
</tbody>
</table>
OLAP IN SQL
Pivot Operation

- `select * from sales pivot (sum(quantity) for color in ('dark','pastel','white')) order by item name;`

<table>
<thead>
<tr>
<th>item_name</th>
<th>clothes_size</th>
<th>dark</th>
<th>pastel</th>
<th>white</th>
</tr>
</thead>
<tbody>
<tr>
<td>dress</td>
<td>small</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>dress</td>
<td>medium</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>dress</td>
<td>large</td>
<td>12</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>pants</td>
<td>small</td>
<td>14</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>pants</td>
<td>medium</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pants</td>
<td>large</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>shirt</td>
<td>small</td>
<td>2</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>shirt</td>
<td>medium</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>shirt</td>
<td>large</td>
<td>6</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>skirt</td>
<td>small</td>
<td>2</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>skirt</td>
<td>medium</td>
<td>5</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>skirt</td>
<td>large</td>
<td>1</td>
<td>15</td>
<td>3</td>
</tr>
</tbody>
</table>
Cube Operation

- The **cube** operation computes union of **group by**’s on every subset of the specified attributes.
- E.g. consider the query

  ```
  select item_name, color, size, sum(number)
  from sales
  group by cube(item_name, color, size)
  ```

  This computes the union of eight different groupings of the `sales` relation:

  ```
  { (item_name, color, size), (item_name, color),
  (item_name, size), (color, size),
  (item_name), (color),
  (size), ( ) } 
  ```

  where ( ) denotes an empty **group by** list.
- For each grouping, the result contains the null value for attributes not present in the grouping.
Online Analytical Processing Operations

- Relational representation of cross-tab that we saw earlier, but with *null* in place of *all*, can be computed by

  ```sql
  select item_name, color, sum(number)
  from sales
  group by cube(item_name, color)
  ```

- The function *grouping()* can be applied on an attribute
  - Returns 1 if the value is a null value representing all, and returns 0 in all other cases.

  ```sql
  select case when grouping(item_name) = 1 then 'all'
    else item_name end as item_name,
  case when grouping(color) = 1 then 'all'
    else color end as color,
  'all' as clothes size, sum(quantity) as quantity
  from sales
  group by cube(item name, color);
  ```
Online Analytical Processing Operations

- Can use the function `decode()` in the `select` clause to replace such nulls by a value such as `all`
  - E.g., replace `item_name` in first query by `decode(grouping(item_name), 1, 'all', item_name)`
Extended Aggregation (Cont.)

- The **rollup** construct generates union on every prefix of specified list of attributes.

\[
\text{select } \text{item}_\text{name}, \text{color}, \text{size}, \text{sum}(\text{number}) \\
\text{from } \text{sales} \\
\text{group by rollup(} \text{item}_\text{name}, \text{color}, \text{size})
\]

Generates union of four groupings:
\[
\{ (\text{item}_\text{name}, \text{color}, \text{size}), (\text{item}_\text{name}, \text{color}), (\text{item}_\text{name}), ( ) \}
\]

- Rollup can be used to generate aggregates at multiple levels of a hierarchy.

- E.g., suppose table \text{itemcategory(} \text{item}_\text{name}, \text{category}) gives the category of each item. Then

\[
\text{select } \text{category}, \text{item}_\text{name}, \text{sum}(\text{number}) \\
\text{from } \text{sales, itemcategory} \\
\text{where } \text{sales.item}_\text{name} = \text{itemcategory.item}_\text{name} \\
\text{group by rollup(} \text{category}, \text{item}_\text{name})
\]

would give a hierarchical summary by \text{item}_\text{name} and by \text{category}. 
Extended Aggregation (Cont.)

- Multiple rollups and cubes can be used in a single group by clause
  - Each generates set of group by lists, cross product of sets gives overall set of group by lists
- E.g.,
  ```sql
  select item_name, color, size, sum(number)
  from sales
  group by rollup(item_name), rollup(color, size)
  ```
  generates the groupings
  \[ \{item\_name, ()\} \times \{(color, size), (color), ()\} \]
  \[ = \{(item\_name, color, size), (item\_name, color), (item\_name),
  (color, size), (color), ()\} \]
- ```sql
  select item_name, color, clothes\_size, sum(quantity)
  from sales
  group by grouping sets ([(color, clothes\_size),
  (clothes\_size, item\_name)]);
  ```
OLAP Implementation

- The earliest OLAP systems used multidimensional arrays in memory to store data cubes, and are referred to as **multidimensional OLAP (MOLAP)** systems.
- OLAP implementations using only relational database features are called **relational OLAP (ROLAP)** systems.
- Hybrid systems, which store some summaries in memory and store the base data and other summaries in a relational database, are called **hybrid OLAP (HOLAP)** systems.
OLAP Implementation (Cont.)

- Early OLAP systems precomputed *all* possible aggregates in order to provide online response
  - Space and time requirements for doing so can be very high
    - \(2^n\) combinations of *group by*
  - It suffices to precompute some aggregates, and compute others on demand from one of the precomputed aggregates
    - Can compute aggregate on \((\text{item\_name, color})\) from an aggregate on \((\text{item\_name, color, size})\)
      - For all but a few “non-decomposable” aggregates such as *median*
      - is cheaper than computing it from scratch
- Several optimizations available for computing multiple aggregates
  - Can compute aggregate on \((\text{item\_name, color})\) from an aggregate on \((\text{item\_name, color, size})\)
  - Can compute aggregates on \((\text{item\_name, color, size}), (\text{item\_name, color})\) and \((\text{item\_name})\) using a single sorting of the base data
Reporting and Visualization

- **Reporting tools** help create formatted reports with tabular/graphical representation of data
  - E.g. SQL Server reporting services, Crystal Reports
- **Data visualization** tools help create interactive visualization of data
  - E.g. Tableau, FusionChart, plotly, Datawrapper, Google Charts, etc
  - Frontend typically based on HTML+JavaScript

<table>
<thead>
<tr>
<th>Region</th>
<th>Category</th>
<th>Sales</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>Computer Hardware</td>
<td>1,000,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Computer Software</td>
<td>500,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All categories</td>
<td></td>
<td>1,500,000</td>
</tr>
<tr>
<td>South</td>
<td>Computer Hardware</td>
<td>200,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Computer Software</td>
<td>400,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All categories</td>
<td></td>
<td>600,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2,100,000</td>
</tr>
</tbody>
</table>
DATA MINING
Data Mining

- **Data mining** is the process of semi-automatically analyzing large databases to find useful patterns
  - Similar goals to machine learning, but on very large volumes of data
- Part of the larger area of **knowledge discovery in databases (KDD)**
- Some types of knowledge can be represented as rules
- More generally, knowledge is discovered by applying machine learning techniques on past instances of data, to form a model
  - Model is then used to make predictions for new instances
Types of Data Mining Tasks

- **Prediction** based on past history
  - Predict if a credit card applicant poses a good credit risk, based on some attributes (income, job type, age, ..) and past history
  - Predict if a pattern of phone calling card usage is likely to be fraudulent

- Some examples of prediction mechanisms:
  - **Classification**
    - Items (with associated attributes) belong to one of several classes
    - **Training instances** have attribute values and classes provided
    - Given a new item whose class is unknown, predict to which class it belongs based on its attribute values
  - **Regression** formulae
    - Given a set of mappings for an unknown function, predict the function result for a new parameter value
Data Mining (Cont.)

- **Descriptive Patterns**
  - **Associations**
    - Find books that are often bought by “similar” customers. If a new such customer buys one such book, suggest the others too.
    - Associations may be used as a first step in detecting **causation**
      - E.g. association between exposure to chemical X and cancer,
  - **Clusters**
    - E.g. typhoid cases were clustered in an area surrounding a contaminated well
    - Detection of clusters remains important in detecting epidemics
Decision Trees

- Each internal node of the tree partitions the data into groups based on a **partitioning attribute**, and a **partitioning condition** for the node.

- Leaf node:
  - all (or most) of the items at the node belong to the same class, or
  - all attributes have been considered, and no further partitioning is possible.

- Traverse tree from top to make a prediction.

- Number of techniques for constructing decision tree classifiers
  - We omit details.
Bayesian Classifiers

- Bayesian classifiers use **Bayes theorem**, which says

\[
p (c_j | d ) = \frac{p (d | c_j ) p(c_j )}{p (d )}
\]

where

- \( p (c_j | d ) = \) probability of instance \( d \) being in class \( c_j \),
- \( p (d | c_j ) = \) probability of generating instance \( d \) given class \( c_j \),
- \( p (c_j ) = \) probability of occurrence of class \( c_j \), and
- \( p (d ) = \) probability of instance \( d \) occurring
Naïve Bayesian Classifiers

- Bayesian classifiers require
  - computation of $p(d | c_j)$
  - precomputation of $p(c_j)$
  - $p(d)$ can be ignored since it is the same for all classes
- To simplify the task, naïve Bayesian classifiers assume attributes have independent distributions, and thereby estimate
  $$p(d | c_j) = p(d_1 | c_j) \times p(d_2 | c_j) \times \ldots \times p(d_n | c_j)$$
  - Each of the $p(d_i | c_j)$ can be estimated from a histogram on $d_i$ values for each class $c_j$
    - the histogram is computed from the training instances
  - Histograms on multiple attributes are more expensive to compute and store
Support Vector Machine Classifiers

- Simple 2-dimensional example:
  - Points are in two classes
  - Find a line (maximum margin line) s.t. line divides two classes, and distance from nearest point in either class is maximum
Support Vector Machine

- In n-dimensions points are divided by a plane, instead of a line.
- SVMs can be used separators that are curve, not necessarily linear, by transforming points before classification.
  - Transformation functions may be non-linear and are called kernel functions.
  - Separator is a plane in the transformed space, but maps to curve in original space.
- There may not be an exact planar separator for a given set of points.
  - Choose plane that best separates points.
- N-ary classification can be done by N binary classifications.
  - In class $i$ vs. not in class $i$. 
Neural Network Classifiers

- Neural network has multiple layers
  - Each layer acts as input to next later
- First layer has input nodes, which are assigned values from input attributes
- Each node combines values of its inputs using some weight function to compute its value
  - Weights are associated with edges
- For classification, each output value indicates likelihood of the input instance belonging to that class
  - Pick class with maximum likelihood
- Weights of edges are key to classification
- Edge weights are learnt during training phase
Neural Network Classifiers

- Value of a node may be a linear combination of inputs, or may be a non-linear function
  - E.g. sigmoid function
- **Backpropagation algorithm** works as follows
  - Weights are set randomly initially
  - Training instances are processed one at a time
    - Output is computed using current weights
    - If classification is wrong, weights are tweaked to get a higher score for the correct class
Neural Networks (Cont.)

- **Deep neural networks** have a large number of layers with large number of nodes in each layer
- **Deep learning** refers to training of deep neural network on very large numbers of training instances
- Each layer may be connected to previous layers in different ways
  - Convolutional networks used for image processing
  - More complex architectures used for text processing, and machine translation, speech recognition, etc.

- Neural networks are a large area in themselves
  - Further details beyond scope of this chapter
Regression deals with the prediction of a value, rather than a class.

- Given values for a set of variables, $X_1, X_2, \ldots, X_n$, we wish to predict the value of a variable $Y$.

One way is to infer coefficients $a_0, a_1, a_2, \ldots, a_n$ such that

$$Y = a_0 + a_1 \times X_1 + a_2 \times X_2 + \ldots + a_n \times X_n$$

Finding such a linear polynomial is called linear regression.

- In general, the process of finding a curve that fits the data is also called curve fitting.

The fit may only be approximate

- because of noise in the data, or
- because the relationship is not exactly a polynomial

Regression aims to find coefficients that give the best possible fit.
Retail shops are often interested in associations between different items that people buy.

- Someone who buys bread is quite likely also to buy milk
- A person who bought the book *Database System Concepts* is quite likely also to buy the book *Operating System Concepts*.

Associations information can be used in several ways.

- E.g. when a customer buys a particular book, an online shop may suggest associated books.

**Association rules:**

\[ \text{bread} \Rightarrow \text{milk} \quad \text{DB-Concepts, OS-Concepts} \Rightarrow \text{Networks} \]

- Left hand side: **antecedent**, right hand side: **consequent**
- An association rule must have an associated **population**; the population consists of a set of **instances**
  - E.g. each transaction (sale) at a shop is an instance, and the set of all transactions is the population
Association Rules (Cont.)

- Rules have an associated support, as well as an associated confidence.

- **Support** is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule.
  - E.g. suppose only 0.001 percent of all purchases include milk and screwdrivers. The support for the rule is $milk \Rightarrow screwdrivers$ is low.

- **Confidence** is a measure of how often the consequent is true when the antecedent is true.
  - E.g. the rule $bread \Rightarrow milk$ has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk.

- We omit further details, such as how to efficiently infer association rules.
Clustering: Intuitively, finding clusters of points in the given data such that similar points lie in the same cluster.

Can be formalized using distance metrics in several ways:
- Group points into $k$ sets (for a given $k$) such that the average distance of points from the centroid of their assigned group is minimized.
  - Centroid: point defined by taking average of coordinates in each dimension.
- Another metric: minimize average distance between every pair of points in a cluster.

Hierarchical clustering: example from biological classification.
- (the word classification here does not mean a prediction mechanism)

```
chordata
  mammalia
    leopards  humans
  reptilia
    snakes  crocodiles
```
Clustering and Collaborative Filtering

- Goal: predict what movies/books/… a person may be interested in, on the basis of
  - Past preferences of the person
  - Preferences of other people

- One approach based on repeated clustering
  - Cluster people based on their preferences for movies
  - Then cluster movies on the basis of being liked by the same clusters of people
  - Again cluster people based on their preferences for (the newly created clusters of) movies
  - Repeat above till equilibrium
  - Given new user
    - Find most similar cluster of existing users and
    - Predict movies in movie clusters popular with that user cluster

- Above problem is an instance of collaborative filtering
Other Types of Mining

- **Text mining**: application of data mining to textual documents
- **Sentiment analysis**
  - E.g. learn to predict if a user review is positive or negative about a product
- **Information extraction**
  - Create structured information from unstructured textual description or semi-structured data such as tabular displays
- **Entity recognition and disambiguation**
  - E.g. given text with name “Michael Jordan” does the name refer to the famous basketball player or the famous ML expert
- **Knowledge graph** (see Section 8.4)
  - Can be constructed by information extraction from different sources, such as Wikipedia
End of Chapter
Best Splits

- Pick best attributes and conditions on which to partition
- The purity of a set S of training instances can be measured quantitatively in several ways.
  - Notation: number of classes = $k$, number of instances = $|S|$, fraction of instances in class $i = p_i$.
- The **Gini** measure of purity is defined as
  \[
  \text{Gini} (S) = 1 - \sum_{i=1}^{k} p_i^2
  \]
  - When all instances are in a single class, the Gini value is 0
  - It reaches its maximum (of $1 - 1/k$) if each class the same number of instances.
Another measure of purity is the entropy measure, which is defined as

\[ \text{entropy} (S) = - \sum_{i=1}^{k} p_i \log_2 p_i \]

When a set \( S \) is split into multiple sets \( S_i, \) \( i=1, 2, \ldots, r \), we can measure the purity of the resultant set of sets as:

\[ \text{purity}(S_1, S_2, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} \text{purity} (S_i) \]

The information gain due to particular split of \( S \) into \( S_i, i = 1, 2, \ldots, r \)

**Information-gain** \( (S, \{S_1, S_2, \ldots, S_r\}) = \text{purity}(S) - \text{purity} (S_1, S_2, \ldots S_r) \)
Best Splits (Cont.)

- Measure of “cost” of a split:
  \[
  \text{Information-content } (S, \{S_1, S_2, \ldots, S_r\}) = - \sum_{i=1}^{r} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
  \]

- Information-gain ratio:
  \[
  \text{Information-gain ratio} = \frac{\text{Information-gain } (S, \{S_1, S_2, \ldots, S_r\})}{\text{Information-content } (S, \{S_1, S_2, \ldots, S_r\})}
  \]

- The best split is the one that gives the maximum information gain ratio.
Finding Best Splits

- Categorical attributes (with no meaningful order):
  - Multi-way split, one child for each value
  - Binary split: try all possible breakup of values into two sets, and pick the best

- Continuous-valued attributes (can be sorted in a meaningful order)
  - Binary split:
    - Sort values, try each as a split point
    - E.g. if values are 1, 10, 15, 25, split at \( \leq 1, \leq 10, \leq 15 \)
    - Pick the value that gives best split
  - Multi-way split:
    - A series of binary splits on the same attribute has roughly equivalent effect
Decision-Tree Construction Algorithm

**Procedure** GrowTree (S )
Partition (S );

**Procedure** Partition (S )
if ( purity (S ) > δ_p or |S| < δ_s ) then
    return;
for each attribute A
    evaluate splits on attribute A;
Use best split found (across all attributes) to partition S into S_1, S_2, ...., S_r,
for i = 1, 2, ......, r
    Partition (S_i );
Finding Association Rules

- We are generally only interested in association rules with reasonably high support (e.g. support of 2% or greater)

- Naïve algorithm
  1. Consider all possible sets of relevant items.
  2. For each set find its support (i.e. count how many transactions purchase all items in the set).
    - **Large itemsets**: sets with sufficiently high support
  3. Use large itemsets to generate association rules.
    - From itemset $A$ generate the rule $A - \{b\} \Rightarrow b$ for each $b \in A$.
      - Support of rule = support ($A$).
      - Confidence of rule = support ($A$) / support ($A - \{b\}$)
Finding Support

- Determine support of itemsets via a single pass on set of transactions
  - Large itemsets: sets with a high count at the end of the pass
- If memory not enough to hold all counts for all itemsets use multiple passes, considering only some itemsets in each pass.
- Optimization: Once an itemset is eliminated because its count (support) is too small none of its supersets needs to be considered.
- The **a priori** technique to find large itemsets:
  - Pass 1: count support of all sets with just 1 item. Eliminate those items with low support
  - Pass $i$: **candidates**: every set of $i$ items such that all its $i$-1 item subsets are large
    - Count support of all candidates
    - Stop if there are no candidates
Other Types of Associations

- Basic association rules have several limitations
- Deviations from the expected probability are more interesting
  - E.g. if many people purchase bread, and many people purchase cereal, quite a few would be expected to purchase both
  - We are interested in positive as well as negative correlations between sets of items
    - Positive correlation: co-occurrence is higher than predicted
    - Negative correlation: co-occurrence is lower than predicted
- Sequence associations / correlations
  - E.g. whenever bonds go up, stock prices go down in 2 days
- Deviations from temporal patterns
  - E.g. deviation from a steady growth
  - E.g. sales of winter wear go down in summer
    - Not surprising, part of a known pattern.
    - Look for deviation from value predicted using past patterns
Hierarchical Clustering

- **Agglomerative clustering algorithms**
  - Build small clusters, then cluster small clusters into bigger clusters, and so on

- **Divisive clustering algorithms**
  - Start with all items in a single cluster, repeatedly refine (break) clusters into smaller ones
Clustering Algorithms

- Clustering algorithms have been designed to handle very large datasets
- E.g. the Birch algorithm
  - Main idea: use an in-memory R-tree to store points that are being clustered
  - Insert points one at a time into the R-tree, merging a new point with an existing cluster if is less than some $\delta$ distance away
  - If there are more leaf nodes than fit in memory, merge existing clusters that are close to each other
  - At the end of first pass we get a large number of clusters at the leaves of the R-tree
    - Merge clusters to reduce the number of clusters