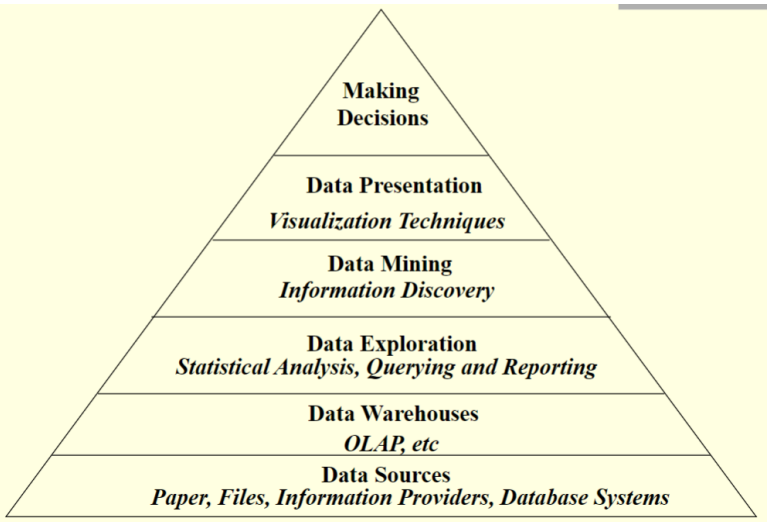
CS3481 notes

Introduction

Fundamentals of Data Science

- Rapid advances in data collection and storage enable large-scale data accumulation.
- Traditional tools may fail with massive datasets; data science principles are needed to discover patterns, rules, and knowledge.

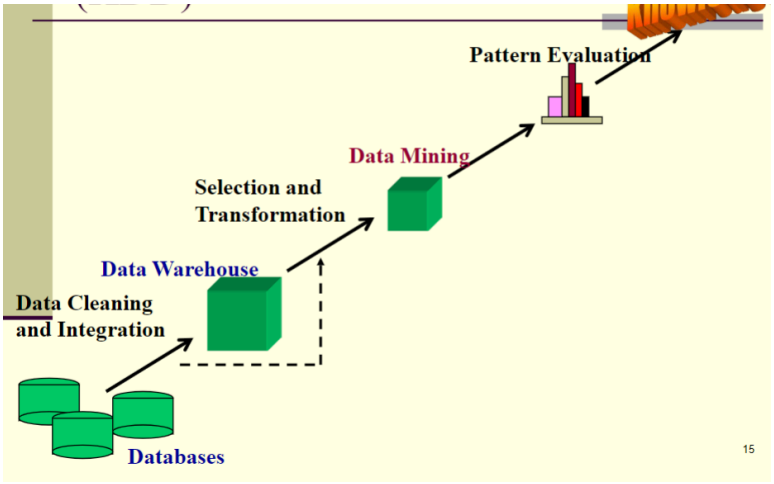
Core Components of Data Science



Data Mining

- Automatic discovery of useful information in large repositories.
- Searches for novel patterns and predicts future outcomes.
- **Not considered data mining**: Simple queries or web searches.

Knowledge Discovery in Databases (KDD)



1. **Data cleaning**: Remove noise and inconsistencies.
2. **Data integration**: Combine multiple sources.
3. **Data selection**: Retrieve relevant data.
4. **Data transformation**: Convert to mining-suitable forms.
5. **Data mining**: Apply intelligent methods to extract patterns.
6. **Pattern evaluation**: Identify interesting patterns.
7. **Knowledge presentation**: Visualize and present results.

Challenges in Data Science

- **Scalability**: Handling gigabytes to petabytes efficiently.
- **High dimensionality**: Hundreds/thousands of attributes (e.g., gene data).
- **Heterogeneous data**: Complex objects (e.g., web pages, DNA sequences).

Data Mining Tasks

1. **Predictive Modeling**
    - Predict a target attribute using explanatory variables.
    - **Classification**: Discrete target (e.g., purchase yes/no).
    - **Regression**: Continuous target (e.g., stock price).
    - Goal: Minimize error between predicted and true values.
2. **Association Analysis**
    - Discover strongly associated items (e.g., {Diapers}→{Milk}).

- Represented as implication rules or item subsets.

3. **Cluster Analysis**
   - Group similar observations (e.g., news articles by topic).
   - Intra-cluster similarity > inter-cluster similarity.
   - Each article is represented as a set of word-frequency pairs (w, c)
   - Example:
     - Each article is represented as a data record
     - **Each attribute**: a unique word in the article
     - Each **value of attribute**: the times occured in the article
     - **Similarity**: calculate distance between two data record
       - Euclidean Distance:

$$d(x,y) = \sqrt{\sum_{u=1}^{n}(x_u - y_u)^2}$$

       - Cosine Similarity:

$$cos(x,y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_{u=1}^{n} x_u y_u}{\sqrt{\sum_{u=1}^{n}(x_u)^2}\sqrt{\sum_{u=1}^{n}(y_u)^2}}$$

Data Representation

Data Objects and Attributes

Student ID	Year	Grade Point Average (GPA)
51034262	1	3.24
51052663	2	3.51
51082246	3	3.62

- **Data Object (row)**: A collection of attributes (also called record, point, vector, etc.).
- **Attribute (column)**: Property of an object (also called variable, feature, dimension).
    - **Types**:
        1. **Nominal**
            - Distinct categories (e.g., gender).
            - Only equality comparisons (e.g., "Is A equal to B?")
        2. **Ordinal**
            - Ordered categories (e.g., grades).
            - Can compare order (e.g., "Is A greater than B?"), but not differences between values.
        3. **Interval**
            - Meaningful differences (e.g., dates).
            - Can measure distance, but cannot measure ratio like "30°C is 1.5 times hotter than 20°C".
        4. **Ratio**
            - Meaningful differences & ratios (e.g., weight).
            - Can measure both distance and ratio.
    - **Discrete**: Finite/countable values (e.g., counts).
    - **Continuous**: Real numbers (e.g., temperature).

Data Sets

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

(a) Record data.

TID	ITEMS
1	Bread, Soda, Milk
2	Beer, Bread
3	Beer, Soda, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Soda, Diaper, Milk

(b) Transaction data.

Projection of x Load	Projection of y Load	Distance	Load	Thickness
10.23	5.27	15.22	27	1.2
12.65	6.25	16.22	22	1.1
13.54	7.23	17.34	23	1.2
14.27	8.43	18.45	25	0.9

(c) Data matrix.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

(d) Document-term matrix.

- **Record Data**: Fixed set of attributes per object (stored in flat files or databases).
- **Transaction Data**: Sets of items (e.g., market baskets).
- **Data Matrix**: Objects as vectors in multi-dimensional space (matrix form).
- **Sparse Data Matrix**: Mostly zero values (e.g., document-term matrices).
  - saving computation time and storage

Data Quality

- **Precision**: Closeness of repeated measurements (variation)
  - often measured by standard deviation
- **Bias**: Systematic deviation from true values
  - measure by taking the difference between
    1. the mean of the set of values
    2. the known value of the quantity being measured
- **Noise**: Random measurement error.
- **Outliers**: Unusual data points.
- **Missing Values**: Strategies:

- Eliminate objects (if too many object have missing value, analysis can not be reliable)
- Estimate values (mode for discrete values, mean for continuous values).

Data Preprocessing

1. **Aggregation**
   - Combine objects to reduce size/variability
   - potential loss of interesting details
2. **Sampling**:
   - **Random Sampling**: Uniform, with/without replacement.
   - Larger sample size: representativeness ↑, advantage of sampling ↓
   - Smaller sample size: important information ↓, advantage of sampling ↑
3. **Dimensionality Reduction**:
   - Advantage:
     - reduce 'curse of dimensionality', which:
       - As the number of dimensions increases, the set of data objects becomes too sparse → harder to find meaningful patterns, lower reliability
     - eliminate irrelevant features and reduce noise
     - allow the data to be more easily visualized
     - reduce time and memory required for processing the data
   - **Feature Transformation**: PCA (linear combinations capturing max variance in the data).
   - **Feature Subset Selection**
     - impractical to try all subset, since the number of subsets involving n attributes is 2^n
     - Standard approach:
       1. Embedded approaches: itself decides which attributes to use or ignore

2. Filter approaches: evaluate features individually (e.g. scoring of features)
3. Wrapper approaches: evaluate feature subsets using model performance

4. **Discretization**: Convert continuous to discrete (binning).
   - first sorting, then map values in the same interval into same discrete value
5. **Normalization**:
   - avoid high valued variable dominates the result of the calculation
   - **Z-score**: $x' = \frac{x - \overline{x}}{\sigma_x}$
   - **Min-Max**: $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$ (range [0,1]).

Similarity and Distance

- **Distance Metrics**:
  - **Minkowski**:
    - $h = 1$: Manhattan distance (L_1)
    - $h = 2$: Euclidean distance (L_2)
    - $h = \infty$: Supremum (L_∞).

$$d(x, y) = (\sum_{u=1}^{n} |x_u - y_u|^h)^{1/h}$$

  - Weighted:

$$d(x, y) = (\sum_{u=1}^{n} w_u |x_u - y_u|^h)^{1/h}$$

Summary Statistics

$$var(x) = \sigma_x^2 = \frac{1}{m-1} \sum_{i=1}^{m} (x_i - \overline{x})^2$$

- u and v refer to u-th and v-th attributes of the data

$$cov(x_u, x_v) = \frac{1}{m-1}\sum_{i=1}^{m}(x_{ui} - \overline{x}_u)(x_{vi} - \overline{x}_v)$$

- Covariance matrix:

$$C = \begin{bmatrix} var(x_u) & cov(x_u, x_v) \\ cov(x_v, x_u) & var(x_v) \end{bmatrix}$$

- r_{uv} range -1 to 1

$$r_{uv} = \frac{cov(x_u, x_v)}{\sigma_u \sigma_v}$$

Model Evaluation

- **Confusion Matrix**:
  - Tabulates correct/incorrect predictions:
    - Correct: $a_{11} + a_{00}$
    - Incorrect: $a_{10} + a_{01}$
  - **Accuracy:** $\frac{a_{11}+a_{00}}{a_{11}+a_{10}+a_{01}+a_{00}}$
  - **Error Rate**: $\frac{a_{10}+a_{01}}{a_{11}+a_{10}+a_{01}+a_{00}}$

Decision Trees

- **Structure**:
  - **Root Node**: No incoming edges.
  - **Internal Nodes**: Attribute test conditions (one incoming, multiple outgoing edges).
  - **Leaf Nodes**: Class labels (one incoming, no outgoing edges).
- **Construction**:
  - **Recursive Partitioning**: Split training records into purer subsets.
  - **Termination**: Leaf node created if all records belong to one class.

Attribute Test Conditions

- **Types**:
  - **Binary**: Two outcomes (e.g., Yes/No).
  - **Nominal**: Multi-way or binary splits ($2^{S-1}-1$ possible binary splits for S values).
  - **Ordinal**: Splits must preserve order (e.g., Small/Medium vs. Large).
  - **Continuous**: Binary split at threshold T (e.g., $x \leq T$).

Impurity Measures

- **Entropy**: Measures uncertainty in class distribution:

$$I(U) = -\sum_{k=1}^{K} p(c_k) log_2 p(c_k)$$

→ **Information Gain**: Reduction in entropy after splitting:

$$Gain(P) = I(U) - \sum_{s=1}^{S} \frac{|U_s|}{|U|} I(U_s)$$

- **Gini Index:**

$$G = 1 - \sum_{k=1}^{K} p(c_k)^2$$

- **Classification Error:**

$$E = 1 - max\ p(c_k)$$

- **Gain Ratio**: Adjusts for multi-valued attributes:

$$\text{Gain Ratio} = \frac{\text{Gain}(P)}{\text{Split Info}}, \ \ \text{Split Info} = -\sum_{s=1}^{S} \frac{|U_s|}{|U|} log_2 \frac{|U_s|}{|U|}$$

Handling Continuous Attributes

- **Threshold Selection**:

- Sort values $\{x_{(1)}, \ldots, x_{(m)}\}$
- Test thresholds $T_r = (x_{(r)} + x_{(r+1)})/2$
- Choose T_r maximizing $\mathrm{Gain}(P, T_r)$

Advanced Topics on Decision Tree

- **Oblique Decision Trees**:
  - Test conditions combine multiple attributes (e.g., x+y<1).
  - Computationally expensive but more expressive.
- **Tree Pruning**: Reduces overfitting by trimming branches.

Classifier Evaluation

Classification Errors

- **Training Error**: Mistakes on the training data (also called resubstitution error).
- **Generalization Error**: Expected mistakes on new, unseen data.

  Goal: A good model should have low errors on both training and unseen data.

Classification Errors

- **Training Error (Resubstitution Error)**: Misclassification errors on the training data.
- **Generalization Error**: Expected error on unseen data.

Model Performance

- A good model should have:
  - Low training error (fits training data well).
  - Low generalization error (performs well on unseen data).

Overfitting vs. Underfitting

- **Underfitting**:

    - Model is too simple.
    - High training and test errors.
    - Occurs when the model cannot capture the true data structure.
- **Overfitting**:
    - Model is too complex.
    - Low training error but high test error.
    - Occurs when the model fits noise in the training data.

Generalization Error Estimation Methods

1. **Resubstitution Estimate**
    - Uses training error as an estimate (optimistic).
2. **Estimates Incorporating Model Complexity**
    - Adjusts training error with a **penalty** for complexity.
    - Formula:

$$e_c(T) = \frac{\text{Training errors} + \text{Penalty} * \text{Number of leaf nodes}}{\text{Total training records}}$$

3. **Validation Set Approach**
    - Splits training data into training and validation sets.
    - Uses validation set (test set) to estimate generalization error.
    - Advantage: used to estimate the generalization error for different parameter settings for the classifier → choosing best parameter to minimize the estimated generalization error.
    - Disadvantage: less examples are available for training the classifier

Handling Overfitting in Decision Trees

- **Pre-pruning**:
    - Stops tree growth early (e.g., change in impurity measure falls below a certain threshold).
- **Post-pruning**:
    - Grows the full tree first, then trims sub-trees.

- Replaces sub-trees with leaf nodes based on majority class (e.g. class 1 if [1, 14, 0]).
- Disadvantage: the additional computations for constructing those sub-trees which are later pruned are wasted

Classifier Evaluation Methods

1. **Hold-Out Method**
   - Splits data into training and test sets.
2. **Cross-Validation**
   - **k-Fold Cross-Validation**:
     - Divides data into *k* equal partitions.
     - Each partition is used as a test set exactly once.
     - Final error is the average of all test errors.

Classification Model

Nearest Neighbor Classifier

- Classifies a test record based on the class labels of its closest training examples.
- **k-Nearest Neighbors (k-NN)**:
  - Uses the **k** closest training points to determine the class.
  - Majority voting resolves ties (random choice if equal).
- **Key Considerations**:
  - Small **k**: Sensitive to noise.
  - Large **k**: May include irrelevant distant points.
- **Computational Cost**: High for large datasets (requires distance calculations to all training points).

Bayes Theorem for Classification

- Estimates probabilistic relationships between attributes and class

- **Posterior Probability**: $P(Y \mid X)$ (probability of a class Y given attributes X).
- **Prior Probability**: $P(Y)$ (baseline probability of class Y).
- **Class-Conditional Probability**: $P(X \mid Y)$ (probability of attributes X given class Y).
- **Evidence**: $P(X)$ (constant, can be ignored for comparison).
- **Classification Rule**: Choose Y that maximizes $P(Y \mid X)$.

Naive Bayes Classifier

- Assumes **conditional independence** of attributes given the class:

$$P(X|Y = c_k) = \prod_{u=1}^{n} P(X_u|Y = c_k)$$

- **We have Posterior Probability:**

$$P(Y = c \mid X) = \frac{P(X|Y = c) * P(Y = c)}{P(X)}$$

- **Advantages**: Efficient (avoids estimating joint probabilities for all attribute combinations).

- **Handling Continuous Attributes:**
  - **Discretization**: Convert to categorical intervals
  - **Gaussian Assumption**: Model $P(X_u = x_u \mid Y = c_k)$ as a normal distribution:

$$P(X_u = x_u \mid Y = c_k) \approx \frac{\epsilon}{\sqrt{2\pi}\sigma_{u,k}} \exp[-\frac{(x_u - \overline{x}_{u,k})^2}{2\sigma_{u,k}^2}]$$

    - where:
      - $\overline{x}_{u,k}$: mean of X_u for class c_k
      - $\sigma_{u,k}^2$: Variance of X_u for class c_k

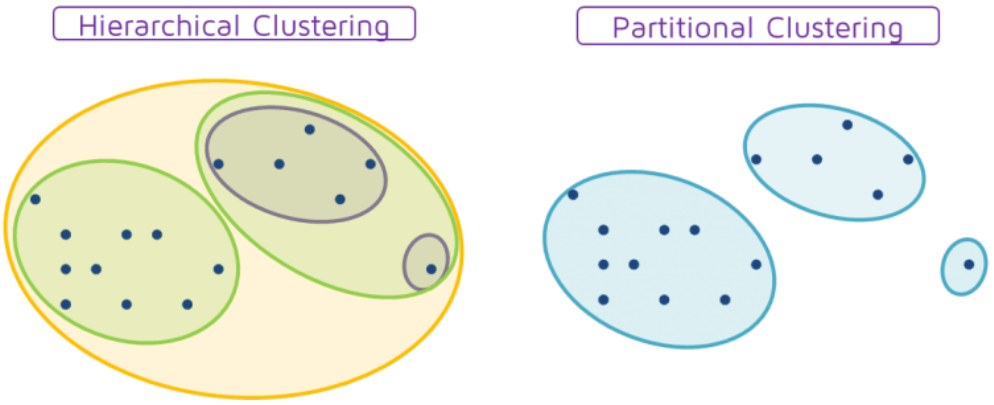
Cluster Analysis

Cluster Analysis Overview

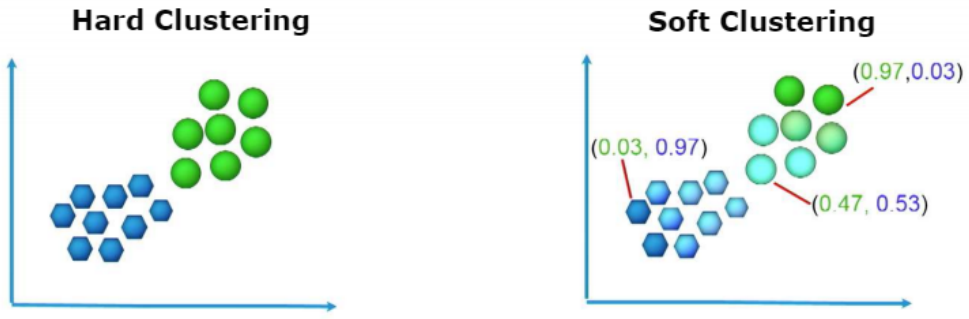
- Groups data objects based on their attributes.
- **Objective**:
  - Objects within a group should be similar.
  - Objects in different groups should be dissimilar.
- **Applications**: Biology, Medicine, Information Retrieval, Business.

Types of Clusterings

- **Partitional vs. Hierarchical**:



  - **Partitional**: Divides data into non-overlapping subsets (clusters).
  - **Hierarchical**: Nested clusters organized as a tree (e.g., dendrogram).
- **Exclusive vs. Fuzzy**:



  - **Exclusive**: Each object belongs to only one cluster.

    - **Fuzzy**: Each object belongs to every cluster with a membership weight (0 to 1).
- **Complete vs. Partial**:
    - **Complete**: Every object is assigned to a cluster.
    - **Partial**: Some objects (e.g., noise/outliers) are not assigned.

K-means Clustering

- **Prototype-based**: Uses centroids as cluster prototypes.
- **Algorithm Steps**:
    1. Select K initial centroids.
    2. Repeat:

        a. Assign each point to the closest centroid.

        b. Recompute centroids.
    3. Until centroids do not change.
- **Key Parameter**: K (number of clusters).

Distance Measure

- **Euclidean (L_2) Distance**: Used to assign points to the nearest centroid.
- **Sum of Squared Error (SSE) (越細越好)**:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} d(x, c_i)^2$$

    - x: Data object.
    - C_i: i-th cluster.
    - c_i: Centroid of C_i.
    - d: Euclidean distance.
- **Centroid Calculation**:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x$$

- m_i: Number of objects in C_i.

Initial Centroids Selection

- Random selection is common but may lead to poor results.
- **Solution**: Perform multiple runs and choose the clustering with the lowest SSE.

Outliers

- Can distort cluster centroids.
- **Identification**: Track each point's contribution to SSE; eliminate high-contribution points.

Post-processing Strategies

- **Decrease SSE (Increase Clusters)**:
  - Split a cluster (choose the split that reduces SSE the most).
  - Introduce a new centroid (e.g., farthest point from current centroids).
- **Increase SSE (Reduce Clusters)**:
  - Disperse a cluster (remove centroid, reassign points).
  - Merge two clusters (choose the pair that minimizes SSE increase).

Limitations of K-means

- Struggles with:
  - Non-spherical cluster shapes.
  - Clusters of widely different sizes or densities.
  - Poorly separated clusters.
- **Examples of Failure**:
  - Large clusters overshadow smaller ones.
  - Dense clusters merge with sparse ones.
  - Non-globular shapes are incorrectly partitioned.

Hierarchical Clustering

- **Definition**: A set of nested clusters organized as a tree (dendrogram).

- **Approaches**:
  - **Agglomerative**:
    - Starts with each point as a separate cluster.
    - Iteratively merges the closest pair of clusters.
  - **Divisive**:
    - Starts with one all-inclusive cluster.
    - Iteratively splits clusters until singleton clusters remain.
- **Representation**:
  - **Dendrogram**: Displays cluster-subcluster relationships and merge/split order.
  - **Nested Cluster Diagram**: For 2-D data.

Cluster Distance Measures (Agglomerative)

- **Single Link (MIN)**:
  - Distance between clusters = minimum distance between any two points (one from each cluster).
  - Formula:

$$d(C_i, C_j) = \min\{d(x, y) \mid x \in C_i, y \in C_j\}$$

  - Handles non-elliptical shapes well.
- **Complete Link (MAX)**:
  - Distance between clusters = maximum distance between any two points.
  - Formula:

$$d(C_i, C_j) = \max\{d(x, y) \mid x \in C_i, y \in C_j\}$$

  - Produces globular clusters.
- **Group Average**:
  - Distance between clusters = average pairwise distance of all points.
  - Formula:

$$d(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} d(x, y)}{m_i \times m_j}$$

        - m_i, m_j: Sizes of clusters C_i and C_j.
    - Intermediate between single and complete link.

Hierarchical Clustering Algorithm Steps

1. Compute the distance matrix for all points.
2. **Repeat**:
    - Merge the two closest clusters (agglomerative) or split a cluster (divisive).
    - Update the distance matrix.
3. **Until** only one cluster remains (agglomerative) or all clusters are singletons (divisive).

Key Issues in Hierarchical Clustering

- **Strengths**:
    - Suitable for multi-level structure requirements.
- **Limitations**:
    - High computational and storage costs ($O(n^2)$ for agglomerative).
    - Irreversible merge/split decisions.

Density-Based Clustering (DBSCAN)

- **Core Idea**: Identifies clusters as regions of high density separated by low-density areas.
- **Key Parameters**:
    - **Eps**: Radius for neighborhood search.
    - **MinPts**: Minimum points required to form a dense region.
- **Point Types**:
    - **Core Point**:
        - Has ≥ MinPts within its Eps-neighborhood.

- **Border Point**:
  - Not a core point but falls within a core point's neighborhood.
- **Noise Point**:
  - Neither core nor border.

DBSCAN Algorithm Steps

1. **Label Points**:
   - For each unprocessed point:
     - If it has ≥ MinPts in its Eps-neighborhood:
       - Mark as core, form a new cluster, and expand by adding all reachable points (core/border).
     - Else: Mark as noise (may be reclassified later).
2. **Expand Clusters**:
   - For each core point, recursively add density-reachable points to the cluster.

Advantages of DBSCAN

- **Robustness**: Resistant to noise and outliers.
- **Flexibility**: Detects arbitrarily shaped clusters (unlike K-means).
- **No Predefined K**: Does not require specifying the number of clusters.

Comparative Strength (Hierarchical vs DBSCAN)

- Hierarchical: Multi-level insights but computationally heavy.
- DBSCAN: Handles noise and complex shapes but sensitive to parameter tuning.

Association Analysis

Introduction to Association Analysis

- **Objective**: Discover interesting relationships (association rules or frequent itemsets) hidden in large transactional datasets (e.g., market baskets).

- **Applications**: Marketing promotions, inventory management, customer relationship management.
- **Data Representation**:
  - **Transactional Format**: Each row (TID) contains a set of items.
  - **Binary Matrix**: Items as columns, transactions as rows (1 = present, 0 = absent).

Key Definitions

- **Itemset**: A collection of items (e.g., {Bread, Milk}).
  - **k-itemset**: Itemset with k items.
- **Support Count ($\sigma(X)$)**: Number of transactions containing itemset X:

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$$

- **Association Rule**: Implication $X \to Z$, where X (antecedent) and Z (consequent) are disjoint itemsets.

Rule Evaluation Metrics

- **Support**: Fraction of transactions containing both X and Z:

$$s(X \to Z) = \frac{\sigma(X \cup Z)}{N}$$

  - N: Total number of transactions.
- **Confidence**: Conditional probability of Z given X:

$$c(X \to Z) = \frac{\sigma(X \cup Z)}{\sigma(X)}$$

- **Thresholds**:
  - **minsup**: Minimum support to prune uninteresting rules.
  - **minconf**: Minimum confidence to ensure reliability.

Frequent Itemset Generation

- **Goal**: Find all itemsets with support ≥ minsup.

- **Challenges**: Exponential search space (2^k possible itemsets for k items).
- **Lattice Structure**: Enumerates all possible itemsets (e.g., null → 1-itemsets → 2-itemsets → ...).

Apriori Principle

- **Core Idea**:
  - If an itemset is frequent, all its subsets must be frequent (anti-monotonicity).
  - If an itemset is infrequent, all its supersets are infrequent.
- **Mathematical Property (for anti-monotone)**:

$$\forall X, Y \in J : (X \subseteq Y) \rightarrow \sigma(Y) \leq \sigma(X)$$

  - J: Power set of all items.

Apriori Algorithm

- **Steps**:
  1. **Initial Pass**: Count 1-itemset support, prune infrequent items.
  2. **Iterative Phase**:
     - **Candidate Generation**: Merge frequent (k–1)-itemsets to form k-itemsets.
     - **Pruning**: Remove candidates with infrequent subsets.
     - **Support Counting**: Scan dataset to validate candidates.
  3. **Termination**: No new frequent itemsets found.

Candidate Generation and Pruning

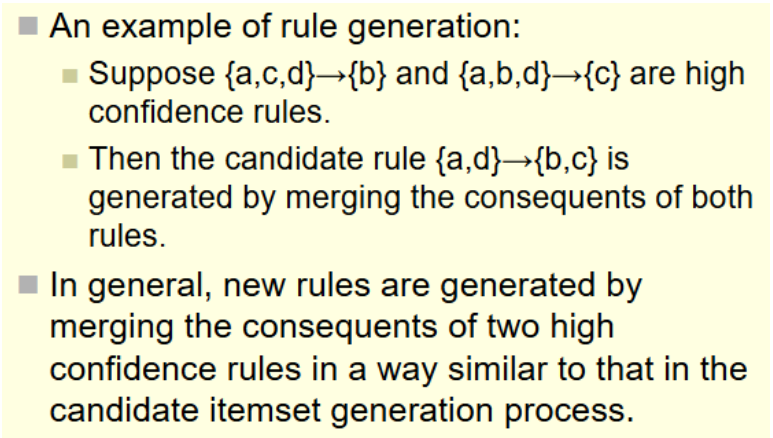
Given a set of frequent (k–1)-itemsets F_{k-1}, generate candidate k-itemsets C_k as follows:

1. **Join Step** (Merging):
   - Take two frequent (k–1)-itemsets that share the first (k–2) items.
   - Merge them to form a candidate k-itemset.
   - Example:

    - Frequent 2-itemsets: {Bread, Milk}, {Bread, Diapers}
    - Merged candidate 3-itemset: {Bread, Milk, Diapers}
2. **Prune Step** (Subset Checking):
    - Remove any candidate k-itemset where at least one (k–1)-subset is **not** frequent.
    - Example:
        - If {Milk, Diapers} is **not** frequent, then {Bread, Milk, Diapers} is pruned.

Rule Generation

- **Process**:
    - For each frequent itemset Y, partition into X and $Y - X$ to form rules $X \rightarrow Y - X$.
    - Compute confidence and prune rules below minconf.
- **Pruning Property**:
    - If $X \rightarrow Y - X$ has low confidence, all rules $X' \rightarrow Y - X'$ (where $X' \subseteq X$) also have low confidence.



Maximal Frequent Itemsets

- Definition: A frequent itemset for which none of its immediate supersets are frequent.

- Properties:
  - Forms the smallest set of itemsets from which all frequent itemsets can be derived.
  - Does not contain the support information of their subsets.
  - An additional pass over the dataset is required to determine the support counts of non-maximal frequent itemsets.

Closed Frequent Itemsets

- Definition: An itemset X is closed if none of its immediate supersets has exactly the same support count as X.
- Properties:
  - An itemset is closed frequent if it is closed and its support is greater than or equal to the minimum support threshold (minsup).
  - Can be used to determine the support counts for non-closed frequent itemsets.
  - The support of a non-closed itemset is equal to the largest support among its immediate supersets.

Remarks

- All maximal frequent itemsets are closed.
- The hierarchy:
  - Frequent Itemsets $\supseteq$ Closed Frequent Itemsets $\supseteq$ Maximal Frequent Itemsets.
- **Closed Itemset Condition:**
  - X is closed if $\nexists Y \supset X$ such that $\text{support}(Y) = \text{support}(X)$
- **Support Determination for Non-Closed Itemsets:**
  - For a non-closed itemset X, $\text{support}(X) = \max(\text{support}(Y))$ where Y is an immediate superset of X.