# HAIMLC701 AI & ML in Healthcare

| | | | |
|---|---|---|---|
| **2.0** | | **AI, ML, Deep Learning and Data Mining Methods for Healthcare** | 10 |
| | 2.1 | Knowledge discovery and Data Mining, ML, Multi classifier Decision Fusion, Ensemble Learning, Meta-Learning and other Abstract Methods. | |
| | 2.2 | Evolutionary Algorithms, Illustrative Medical Application-Multiagent Infectious Disease Propagation and Outbreak Prediction, Automated Amblyopia Screening System etc. | |
| | 2.3 | Computational Intelligence Techniques, Deep Learning, Unsupervised learning, dimensionality reduction algorithms. | |

# Unsupervised Learning

- refers to the process of learning a model from unlabeled data means that input data (x) is supplied without output (y)
- Semi-supervised learning occurs when some output labels (y) are supplied
- For example, learning would be semi-supervised in a model learning to predict diabetic retinopathy from patient eye scans with partially labeled data
- Unsupervised learning can be resource consuming concerning time, money, and expertise
- Unsupervised learning is composed of two main problem concepts:
  - clustering and association

# Clustering

- Clustering refers to the process of discovering relationships within the data
- Clustering is used for a variety of healthcare uses including the following
  - Grouping patients of similar profiles together for monitoring
  - Detecting anomalies or outliers in claims or transactions
  - Defining treatment groups based on medication or condition

# What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

# Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

# Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

# Considerations for Cluster Analysis

- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one class)
- Similarity measure
  - Distance-based (e.g., Euclidian, road network, vector)  vs. connectivity-based (e.g., density or contiguity)
- Clustering space
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

# Requirements and Challenges

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality

# Major Clustering Approaches

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS, FCM
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

# Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where $c_i$ is the centroid or medoid of cluster $C_i$)

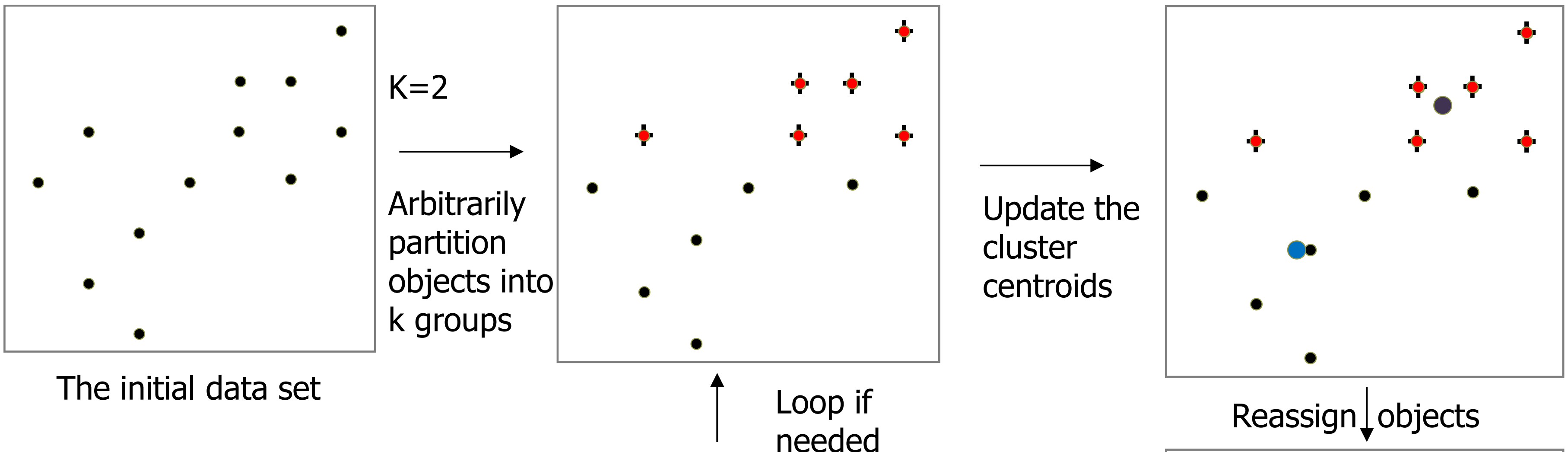
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

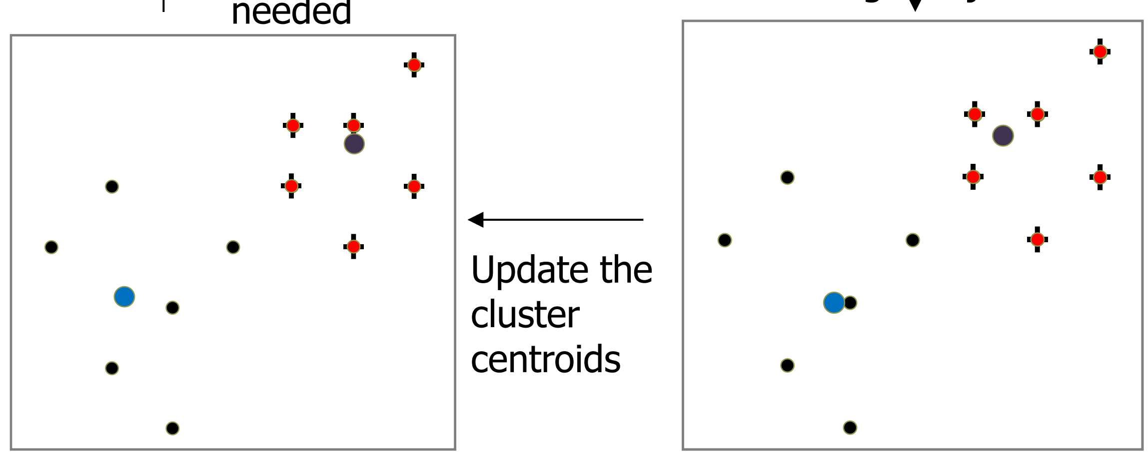
- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* : Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids): Each cluster is represented by one of the objects in the cluster

# The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

# An Example of *K-Means* Clustering




- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

# Example K- Means

Initial Centroids:
A1: (2, 10)
B1: (5, 8)
C1: (1, 2)

New Centroids:
A1: (2, 10)
B1: (6, 6)
C1: (1.5, 3.5)

| Data Points | | | Distance to | | | Cluster | New Cluster |
|---|---|---|---|---|---|---|---|
| | | | 2, 10 | 5, 8 | 1, 2 | | |
| A1 | 2 | 10 | 0.00 | 3.61 | 8.06 | 1 | |
| A2 | 2 | 5 | 5.00 | 4.24 | 3.16 | 3 | |
| A3 | 8 | 4 | 8.49 | 5.00 | 7.28 | 2 | |
| B1 | 5 | 8 | 3.61 | 0.00 | 7.21 | 2 | |
| B2 | 7 | 5 | 7.07 | 3.61 | 6.71 | 2 | |
| B3 | 6 | 4 | 7.21 | 4.12 | 5.39 | 2 | |
| C1 | 1 | 2 | 8.06 | 7.21 | 0.00 | 3 | |
| C2 | 4 | 9 | 2.24 | 1.41 | 7.62 | 2 | |

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

| Data Points | | | Distance to | | | | | | Cluster | New Cluster |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 3.67 | 9 | 7 | 4.33 | 1.5 | 3.5 | | |
| A1 | 2 | 10 | 1.94 | | 7.56 | | 6.52 | | 1 | 1 |
| A2 | 2 | 5 | 4.33 | | 5.04 | | 1.58 | | 3 | 3 |
| A3 | 8 | 4 | 6.62 | | 1.05 | | 6.52 | | 2 | 2 |
| B1 | 5 | 8 | 1.67 | | 4.18 | | 5.70 | | 1 | 1 |
| B2 | 7 | 5 | 5.21 | | 0.67 | | 5.70 | | 2 | 2 |
| B3 | 6 | 4 | 5.52 | | 1.05 | | 4.53 | | 2 | 2 |
| C1 | 1 | 2 | 7.49 | | 6.44 | | 1.58 | | 3 | 3 |
| C2 | 4 | 9 | 0.33 | | 5.55 | | 6.04 | | 1 | 1 |

# Comments on the *K-Means* Method

- Strength: *Efficient*: $O(tkn)$, where $n$ is # objects, $k$ is # clusters, and $t$ is # iterations. Normally, $k$, $t << n$.
    - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
    - Applicable only to objects in a continuous n-dimensional space
        - Using the k-modes method for categorical data
        - In comparison, k-medoids can be applied to a wide range of data
    - Need to specify *k, the number* of clusters, in advance
    - Sensitive to noisy data and *outliers*
    - Not suitable to discover clusters with *non-convex shapes*

# Association

- These methods extract rules that best explain perceived relationships between variables in data
- Association rule learning is historically best applied to online shopping checkout basket datasets gathered on users' purchasing habits
- Through analyzing transactional datasets, the probability of associations can be predicted
- In healthcare, in particular, associative symptoms can be understood to predict better and diagnose disease and adverse events
- Potential adverse effects based on medication and associative patient comorbidity pathways could lead to improved care and treatment pathways
- Three important metrics
    - Support is the value of absolute frequency
    - An association rule holds with support sup in dataset T if the sup % of transactions contain X U Y
    - This represents how popular an itemset is, as measured by the proportion of transactions in which an itemset appears
- sup = Pr (XUY) =count(X U Y)/total transaction count

# Association

- **Confidence**
  - The confidence measure represents correlative frequency
  - An association rule holds in dataset T with confidence conf if the conf % of transactions that contain X also contain Y
  - This estimates how likely item Y is to occur or be present in the transactional dataset when item X occurs
  - This is expressed as {X → Y} and measures the proportion of transactions with item X in which item Y also appears.
  - conf = Pr(Y|X) = count(X U Y)/count(X)
- **Lift**
  - Lift determines how likely item Y is given that X occurs while accommodating for Y's popularity.
- Lift = Support (X U Y)/Support (X) * Support (Y)

Association rule learning is a data mining technique that identifies frequent patterns, connections and dependencies among different groups of items called itemsets in data.

# Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

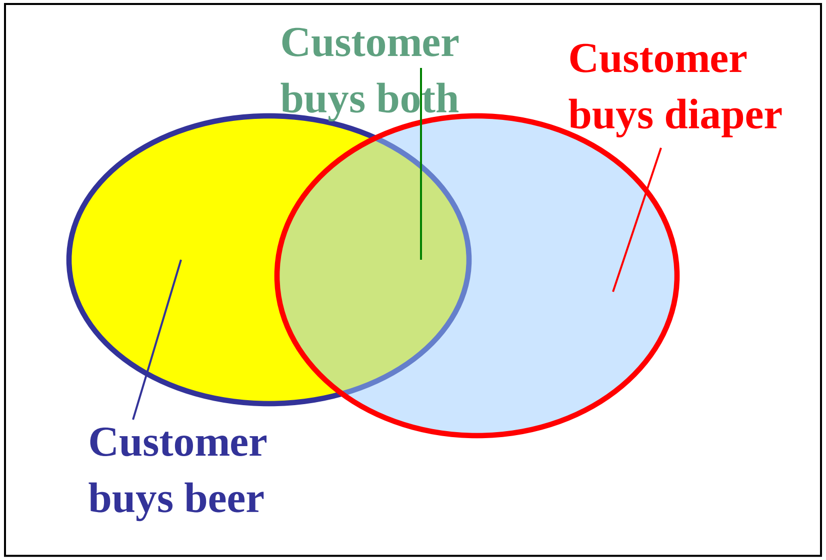
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

# Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

# Basic Concepts: Frequent Patterns

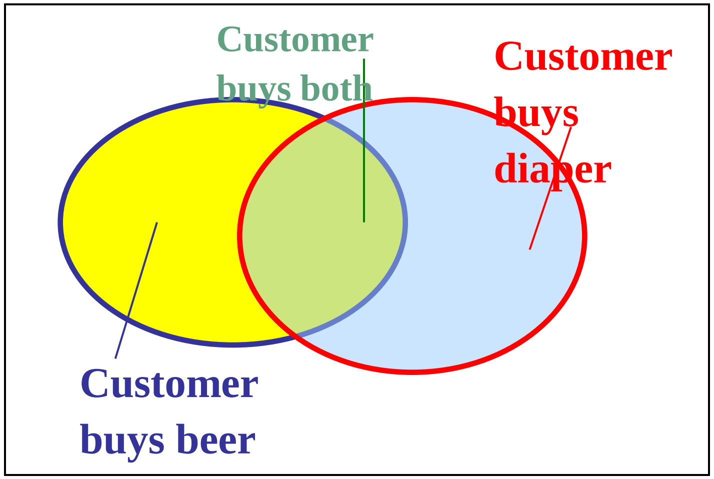
| Tid | Items bought |
|---|---|
| 10 | Beer, Nuts, Diaper |
| 20 | Beer, Coffee, Diaper |
| 30 | Beer, Diaper, Eggs |
| 40 | Nuts, Eggs, Milk |
| 50 | Nuts, Coffee, Diaper, Eggs, Milk |




- itemset: A set of one or more items
- k-itemset X = $\{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

# Basic Concepts: Association Rules

| Tid | Items bought |
|---|---|
| 10 | Beer, Nuts, Diaper |
| 20 | Beer, Coffee, Diaper |
| 30 | Beer, Diaper, Eggs |
| 40 | Nuts, Eggs, Milk |
| 50 | Nuts, Coffee, Diaper, Eggs, Milk |




- Find all the rules *X* → *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

*Let minsup = 50%, minconf = 50%*

*Freq. Pat.:* Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

# Apriori: A Candidate Generation & 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

$Sup_{min} = 2$

Database TDB

| Tid | Items |
|---|---|
| 10 | A, C, D |
| 20 | B, C, E |
| 30 | A, B, C, E |
| 40 | B, E |

1st scan →

$C_1$

| Itemset | sup |
|---|---|
| {A} | 2 |
| {B} | 3 |
| {C} | 3 |
| {D} | 1 |
| {E} | 3 |

$L_1$

| Itemset | sup |
|---|---|
| {A} | 2 |
| {B} | 3 |
| {C} | 3 |
| {E} | 3 |

$C_2$

| Itemset |
|---|
| {A, B} |
| {A, C} |
| {A, E} |
| {B, C} |
| {B, E} |
| {C, E} |

2nd scan ←

$C_2$

| Itemset | sup |
|---|---|
| {A, B} | 1 |
| {A, C} | 2 |
| {A, E} | 1 |
| {B, C} | 2 |
| {B, E} | 3 |
| {C, E} | 2 |

$L_2$

| Itemset | sup |
|---|---|
| {A, C} | 2 |
| {B, C} | 2 |
| {B, E} | 3 |
| {C, E} | 2 |

$C_3$

| Itemset |
|---|
| {B, C, E} |

3rd scan →

$L_3$

| Itemset | sup |
|---|---|
| {B, C, E} | 2 |

# The Apriori Algorithm (Pseudo-Code)

$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k

$L_1$ = {frequent items};
**for** ($k = 1$; $L_k$ $!= \varnothing$; $k$++) **do begin**
    $C_{k+1}$ = candidates generated from $L_k$;
    **for each** transaction $t$ in database do
        increment the count of all candidates in $C_{k+1}$ that are
         contained in $t$
    $L_{k+1}$ = candidates in $C_{k+1}$ with min_support
    **end**
**return** $\cup_k L_k$;

# Implementation 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*}

$$Rule: \quad X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

- Here are a dozen sales transactions.
- The objective is to use this transaction data to find affinities between products, that is, which products sell together often.
- The support level will be set at 33 percent; the confidence level will be set at 50 percent.

Support is a measure of the number of times an item set appears in a dataset.

Confidence is a measure of the likelihood that an itemset will appear if another itemset appears.

**Transactions List**

| | | | | |
|---|---|---|---|---|
| 1 | Milk | Egg | Bread | Butter |
| 2 | Milk | Butter | Egg | Ketchup |
| 3 | Bread | Butter | Ketchup | |
| 4 | Milk | Bread | Butter | |
| 5 | Bread | Butter | Cookies | |
| 6 | Milk | Bread | Butter | Cookies |
| 7 | Milk | Cookies | | |
| 8 | Milk | Bread | Butter | |
| 9 | Bread | Butter | Egg | Cookies |
| 10 | Milk | Butter | Bread | |
| 11 | Milk | Bread | Butter | |
| 12 | Milk | Bread | Cookies | Ketchup |

| **1-item Sets** | **Frequency** |
|---|---|
| Milk | 9 |
| Bread | 10 |
| Butter | 10 |
| Egg | 3 |
| Ketchup | 3 |
| Cookies | 5 |

| Frequent 1-item Sets | Frequency |
|---|---|
| Milk | 9 |
| Bread | 10 |
| Butter | 10 |
| Cookies | 5 |

| | | | | |
|---|---|---|---|---|
| 1 | Milk | Egg | Bread | Butter |
| 2 | Milk | Butter | Egg | Ketchup |
| 3 | Bread | Butter | Ketchup | |
| 4 | Milk | Bread | Butter | |
| 5 | Bread | Butter | Cookies | |
| 6 | Milk | Bread | Butter | Cookies |
| 7 | Milk | Cookies | | |
| 8 | Milk | Bread | Butter | |
| 9 | Bread | Butter | Egg | Cookies |
| 10 | Milk | Butter | Bread | |
| 11 | Milk | Bread | Butter | |
| 12 | Milk | Bread | Cookies | Ketchup |

| **2-item Sets** | **Frequency** |
|---|---|
| Milk, Bread | 7 |
| Milk, Butter | 7 |
| Milk, Cookies | 3 |
| Bread, Butter | 9 |
| Butter, Cookies | 3 |
| Bread, Cookies | 4 |

| Frequent 2-item Sets | Frequency |
|---|---|
| Milk, Bread | 7 |
| Milk, Butter | 7 |
| Bread, Butter | 9 |
| Bread, Cookies | 4 |

| **3-item Sets** | **Frequency** |
|---|---|
| Milk, Bread, Butter | 6 |
| Milk, Bread, Cookies | 1 |
| Bread, Butter, Cookies | 3 |
| Milk, Butter, Cookies | 2 |

| Frequent 3-item Sets | Frequency |
|---|---|
| Milk, Bread, Butter | 6 |

Frequent 3-Item Set = I => {Milk, Bread, Butter}

Non-Empty subset are

– {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}

How to form Association Rule…?

– For every non-empty subset S of I, the association rule is,
  - **S → (I-S)**
  - **If support(I) / support(S) >= min_confidence**

Non-Empty subset are

– {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}

– Min_Support = 30% and Min_Confidence = 60%

Rule 1: {Milk} → {Bread, Butter} {S=50%, C=66.67%}

– Support = 6/12 = 50%

– Confidence = Support (Milk, Bread, Butter)/Support(Milk) = $\frac{6/12}{9/12}$ = 6/9 = 66.67% > 60%

– Valid

# Data Reduction Strategies

- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.
- Data reduction strategies
  - Dimensionality reduction, e.g., remove unimportant attributes
    - Wavelet transforms
    - Principal Components Analysis (PCA) –project original data in smaller space
    - Attribute subset selection – irrelevant/weakly relevant/ redundant attributes are detected and removed
  - Numerosity reduction - replace the original data volume by alternative, smaller forms of data representation.
    - Parametric -Regression and log-linear models
    - Non-parametric -histograms, clustering, sampling, and data cube aggregation
  - Data compression

# Data Reduction 1: Dimensionality Reduction

- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Wavelet transforms
  - Principal Component Analysis
  - Supervised and nonlinear techniques (e.g., feature selection)

# Principal Component Analysis (PCA)

## Definition

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

The number of principal components is less than or equal to the number of original variables.

## Goals

- The main goal of a PCA analysis is to identify patterns in data
- PCA aims to detect the correlation between variables.
- It attempts to reduce the dimensionality.

# Dimensionality Reduction

It reduces the dimensions of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d) in order to increase the computational efficiency while retaining most of the information.

# Transformation

This transformation is defined in such a way that the first principal component has the largest possible variance and each succeeding component in turn has the next highest possible variance.

# PCA Approach

- Standardize the data.
- Perform Singular Vector Decomposition to get the Eigenvectors and Eigenvalues.
- Sort eigenvalues in descending order and choose the k- eigenvectors
- Construct the projection matrix from the selected k- eigenvectors.
- Transform the original dataset via projection matrix to obtain a k-dimensional feature subspace.

# Applications of PCA :

- Interest Rate Derivatives Portfolios
- Neuroscience

## PCA Approach

It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.