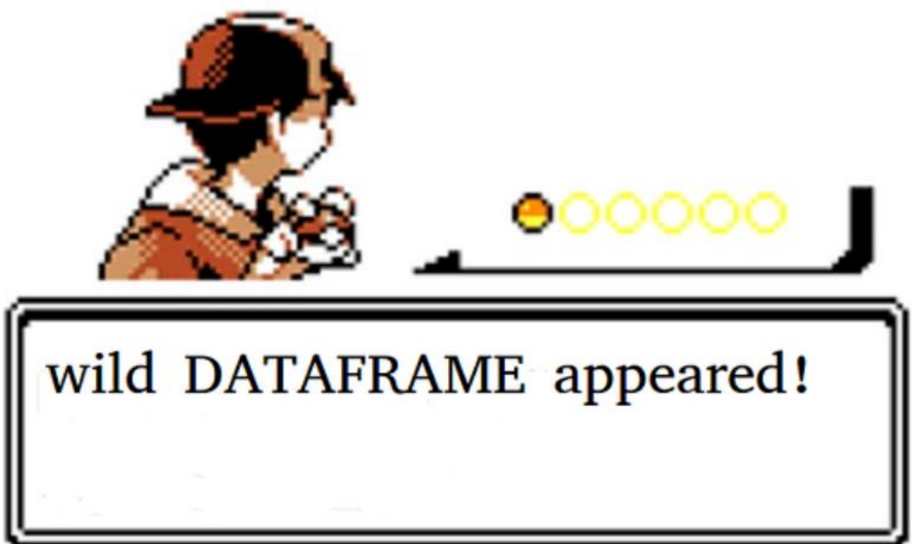
CISC 372
Tuning Methods

	name	age	state	num_children	num_pets
0	john	23	iowa	2	0
1	mary	78	dc	2	4
2	peter	22	california	0	0
3	jeff	19	texas	1	5
4	bill	45	washington	2	0
5	lisa	33	dc	1	0



Last Week

- Cluster analysis groups objects based on their similarity and has wide applications
- Measure of similarity can be computed for various types of data
- Clustering algorithms can be categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- There are still lots of research issues on cluster analysis

Tuning Hyperparameters

- Using the dataset for tuning hyperparameters
  - Recall – The data science workflow
  - With a training set and a testing set
    - Cross-validation on training set for tuning
    - .623 bootstrapping for tuning
  - With a training set, a validation set, and a testing set
    - Training set is for training
    - Validation used for error estimation
    - Based on the estimated error, adjust hyperparameters
    - Testing set used for final testing (like the leaderboard)

Hypermeter Search Algorithm

- The space of hyperparameter is large:
  - Number of trees: (can be any positive number)
  - Regularization weight: (can be any number)
  - Different configurations: (kernels options, discrete choice)
- Educated guess
  - Guess the value to start with
  - Or guess the range of the values to start with
- Automated search

Hypermeter Search Algorithm

- Automated search
  - Grid search
    - Try out every combination of the parameters:
    - Computationally expensive
    - Global optimal (within the given range)
    - Sklearn: *model_selection.GridSearchCV*
  - Random search
    - Try out a random subset
    - `good enough`
    - Local optimal (within the given range)
    - Efficient (less trials)
    - Sklearn: *model_selection.RandomizedSearchCV*
  - Bayesian Optimization
    - As an optimization problem
    - Trial -> estimated error -> Bayesian model estimates the next parameter to try -> trial -> repeat..
    - *pip install bayesian-optimization*

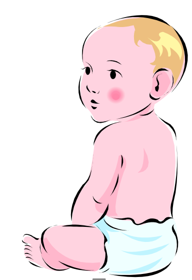
CISC 372
Transaction data

Association and Correlation Analysis

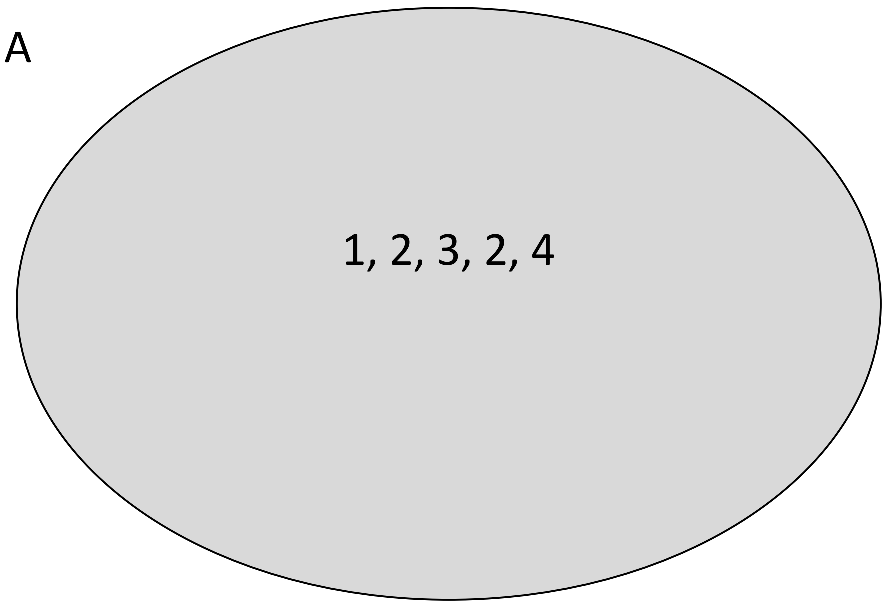
- Frequent patterns (or frequent itemsets)
  - What items are frequently purchased together in Walmart?
- Association, correlation vs. causality
  - A typical association rule
    - Diaper → Beer

    [Support=40%, Confidence=67%]
- How to mine such patterns and rules efficiently in large datasets?
- How to use such patterns for classification, clustering, and other applications?

Transaction database	
TID	**Items bought**
100	bread, butter, diaper
200	bread, butter, diaper, beer
300	bread, butter, pencil
400	orange, pencil, beer
500	diaper, beer, pencil, bread

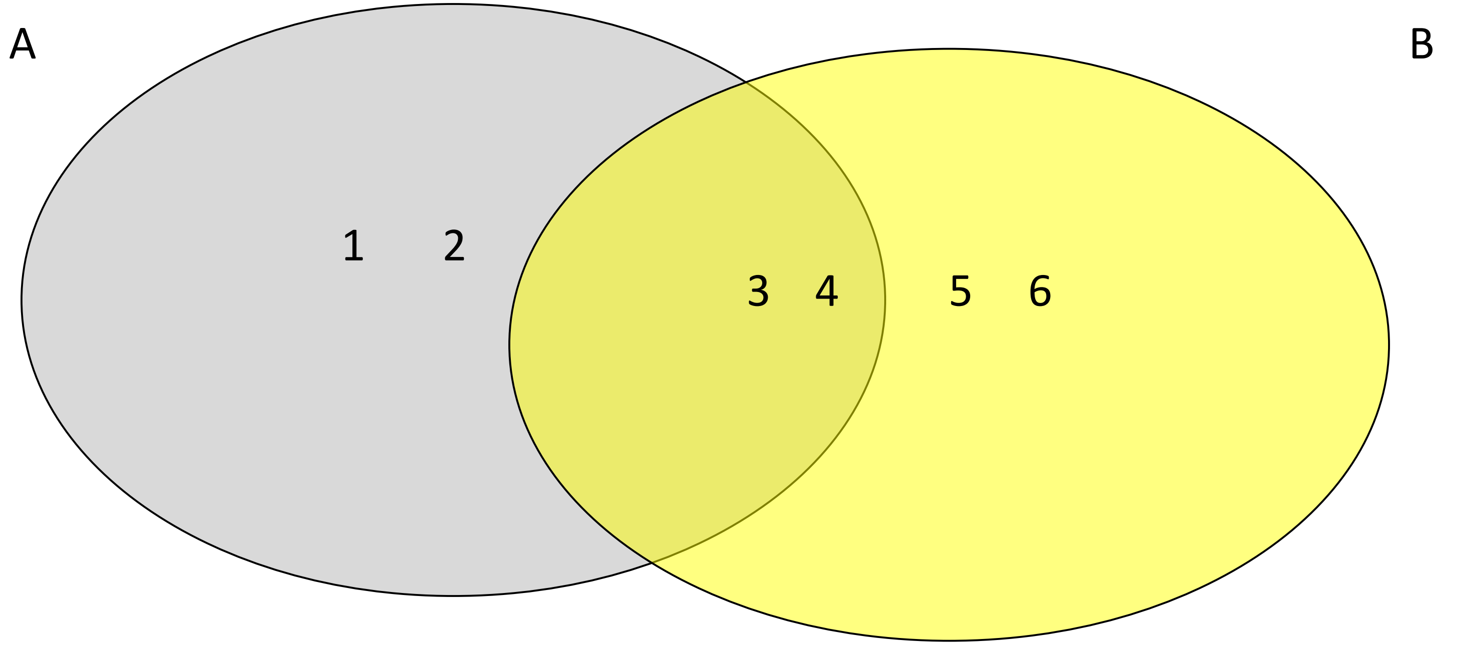


Set notations



- $A = \{1, 2, 3, 2, 4\} = \{1, 2, 2, 3, 4\} = \{1, 2, 3, 4\}$
- $1 \in A$ $\quad 4 \in A$ $\quad 5 \notin A$
- $\{1, 2\} \subset A$ $\quad \{1, 2, 3, 4\} \subseteq A$ $\quad \{3, 5\} \not\subset A$
- $A \supset \{1, 2\}$ $\quad \{1, 2, 3, 4\} \supseteq A$

Set notations



- A = {1, 2, 3, 4} B = {3, 4, 5, 6}
- $A \cup B$ = {1, 2, 3, 4, 5, 6} $A \cap B$ = {3, 4}

Transactions in Real Applications

- A large department store often carries more than 100 thousand different kinds of items
  - Amazon.com carries more than 2M books.
  - Walmart has more than 20 million transactions per day.
  - AT&T produces more than 275 million calls per day

- Mining large transaction databases of many items is a real demand

What Is Frequent Pattern Analysis?

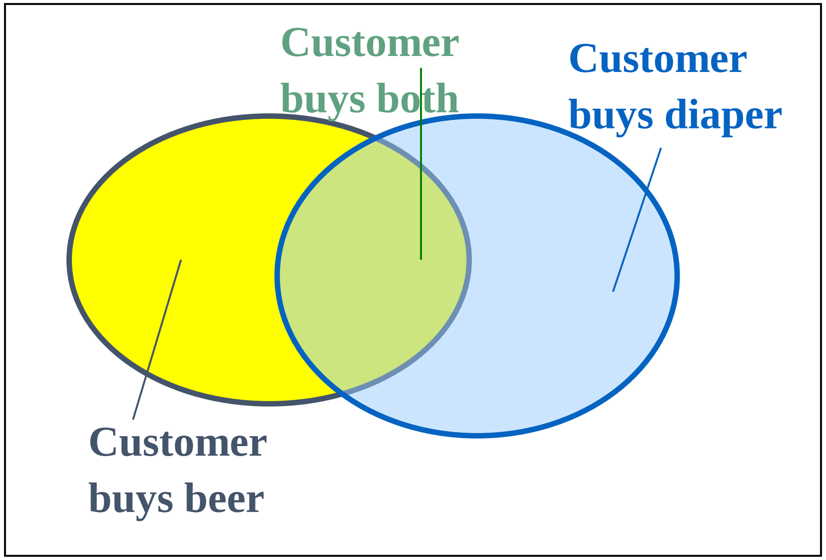
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to a particular new drug?
- 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?

- Discloses an intrinsic and important property of data sets
- Forms the 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: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient

Basic Concepts: Frequent Patterns

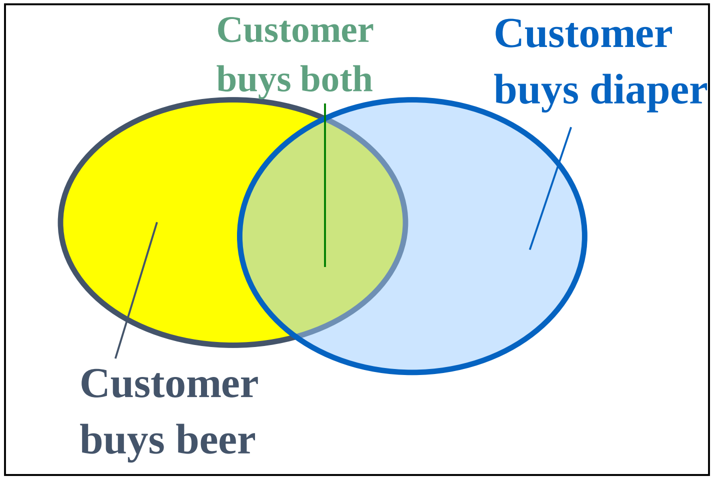
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- 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*, *sup*, 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, *sup*, probability that a transaction contains X ∪ Y
  - confidence, *conf,* conditional probability that a transaction having X also contains *Y*

Let minsup = 50%, minconf = 50%

Frequent patterns: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: *X*→*Y (sup, conf)*
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

A Naïve Attempt

- Generate all possible itemsets, test their supports against the database
- A transaction of length 100 needs to update the support of 2^{100}-1 = 1.27x10^{30} possible itemsets.
- How to hold a large number of itemsets into main memory?
- How to test the supports of a huge number of itemsets against a large database, say containing 100 million transactions?

Tid	Items bought
1	A, B, C
2	B, C
3	A, C
…	C
100000000	B, C

How to Get an Efficient Method?

- Reduce the number of itemsets that need to be checked
- Check the supports of selected itemsets efficiently
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Frequent pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM'02)

The Downward Closure Property

- Any subset of a frequent itemset must be also frequent – downward closure (apriori) property
  - If **{beer, diaper, nuts}** is frequent, then **{beer, diaper}** must also be frequent.
  - A transaction containing {beer, diaper, nuts} also contains {beer, diaper}.

- In other words, any superset of an infrequent itemset must also be infrequent
  - No superset of any infrequent itemset should be generated or tested.
    - Suppose sup(eggs)=2, sup(beer)=3.
    - Then sup(egg,beer) <= 2
  - Many item combinations can be pruned!

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

Apriori: A Candidate Generation-and-Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)

- 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

minsup = 2

Database

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—Another Example

minsup = 2

Database

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

1st scan

C_1

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

L_1

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

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}	2
{C, E}	1

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	2

C_3

Itemset
{B, C, E}

Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- How to count supports of candidates?
- 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*}

To generate C_k, we join an itemset in L_{k-1} only if the first k-2 items are identical.

Then, we check whether or not this itemset has support > min_sup.

Generating Association Rules from Frequent Itemsets

- Once the frequent itemsets have been found, generating strong association rules from them is straight forward.
- An association rule A $\Rightarrow$ B is *strong* if it satisfies both minimum support *min_sup* and minimum confidence *min_conf*.

$$sup(A \Rightarrow B) = \frac{\#\ of\ records\ containing\ A\ and\ B}{total\ number\ of\ records}$$

$$confidence(A \Rightarrow B) = \frac{\#\ of\ records\ containing\ A\ and\ B}{number\ of\ records\ containting\ A}$$

Generating Association Rules from Frequent Itemsets

- Methods:

1. For each frequent itemset X, generate all non-empty subsets of X.

2. For every non-empty subset s of X, output the rule $s \Rightarrow (X-s)$ if $conf(s \Rightarrow (X-s)) \geq$ *min_conf*.

Generating Association Rules from Frequent Itemsets (example)

- Suppose X = {a,b,c} and min_conf = 60%
  - $a \Rightarrow b$ (conf=3/3=100%) ✓
  - $b \Rightarrow a$ (conf=3/5=60%) ✓
  - $a \Rightarrow c$ (conf=2/3=67%) ✓
  - $c \Rightarrow a$ (conf=2/4=50%) ✗
  - $b \Rightarrow c$ (conf=4/5=80%) ✓
  - $c \Rightarrow b$ (conf=4/4=100%) ✓
  - $a \wedge b \Rightarrow c$ (conf=2/3=67%) ✓
  - $a \wedge c \Rightarrow b$ (conf=2/2=100%) ✓
  - $b \wedge c \Rightarrow a$ (conf=2/4=50%) ✗
  - $a \Rightarrow b \wedge c$ (conf=2/3=67%) ✓
  - $b \Rightarrow a \wedge c$ (conf=2/5=40%) ✗
  - $c \Rightarrow a \wedge b$ (conf=2/4=55%) ✗

Tid	Items bought
10	a, b, c, d
20	b, c, f
30	a, b, c, d
40	a, b, e
50	b, c, g, h

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
- The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower

support and confidence

- Measure of dependent/correlated events: lift
- **<1 means negatively correlated; >1 means positively correlated.**

$$lift(X,Y) = \frac{P(X \cup Y)}{P(X)P(Y)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(B,C) = \frac{2000/5000}{3000/5000*3750/5000} = 0.89$$

$$lift(B,\neg C) = \frac{1000/5000}{3000/5000*1250/5000} = 1.33$$

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FP-growth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Generating association rules from frequent patterns.