ML – Unsupervised Learning

CPSC 383: Explorations in Artificial Intelligence and Machine Learning
Fall 2025

Jonathan Hudson, Ph.D
Associate Professor (Teaching)
Department of Computer Science
University of Calgary

August 27, 2025

UNIVERSITY OF CALGARY

Categories

In **unsupervised learning**, the system is given a data set and must find some inherent patterns or structure.

- e.g. Netflix recommendations, identifying high-value customers, detecting unusual bank transactions, topic modeling

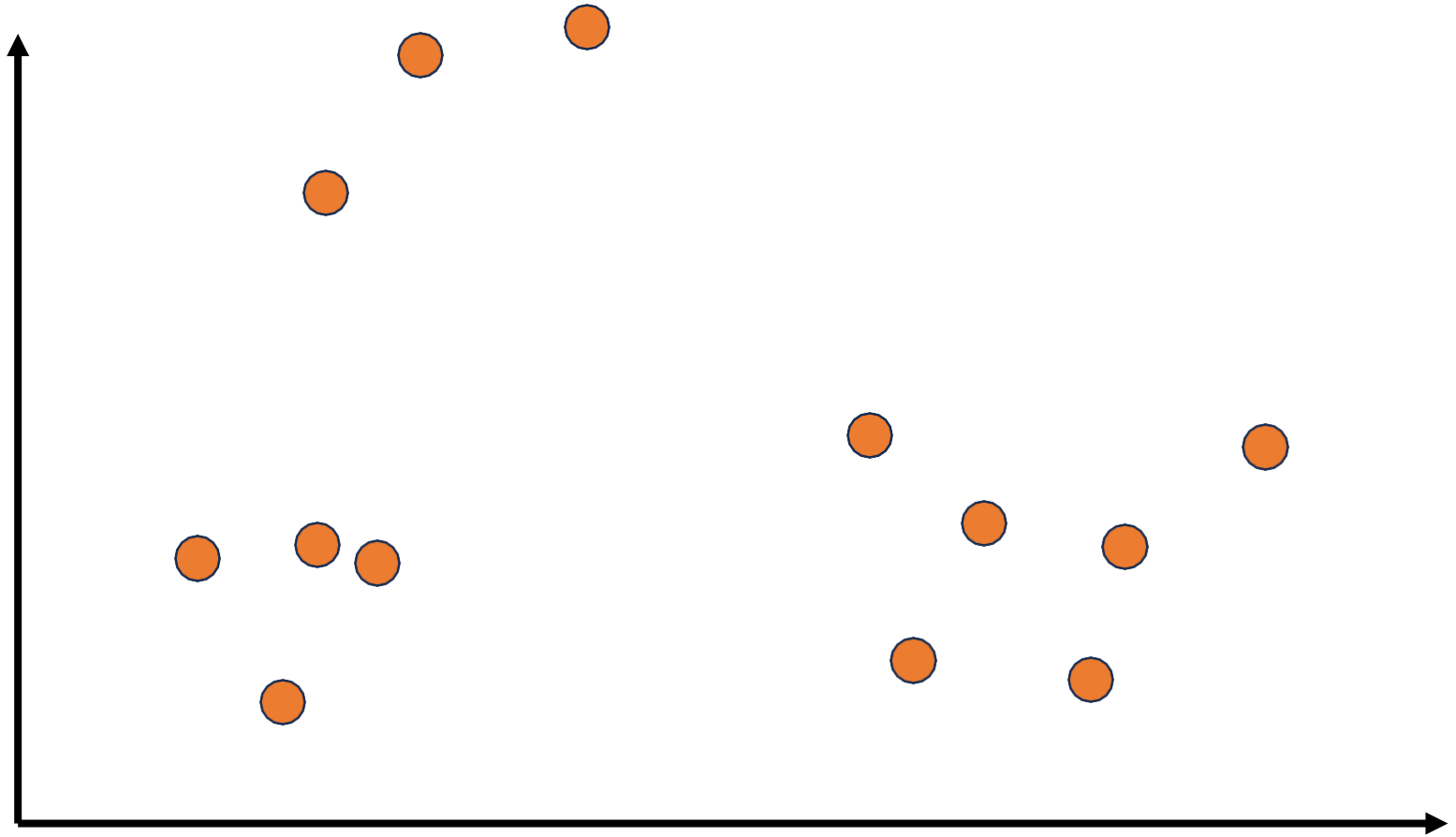
Categories

In **clustering**, the goal is to group similar objects together.

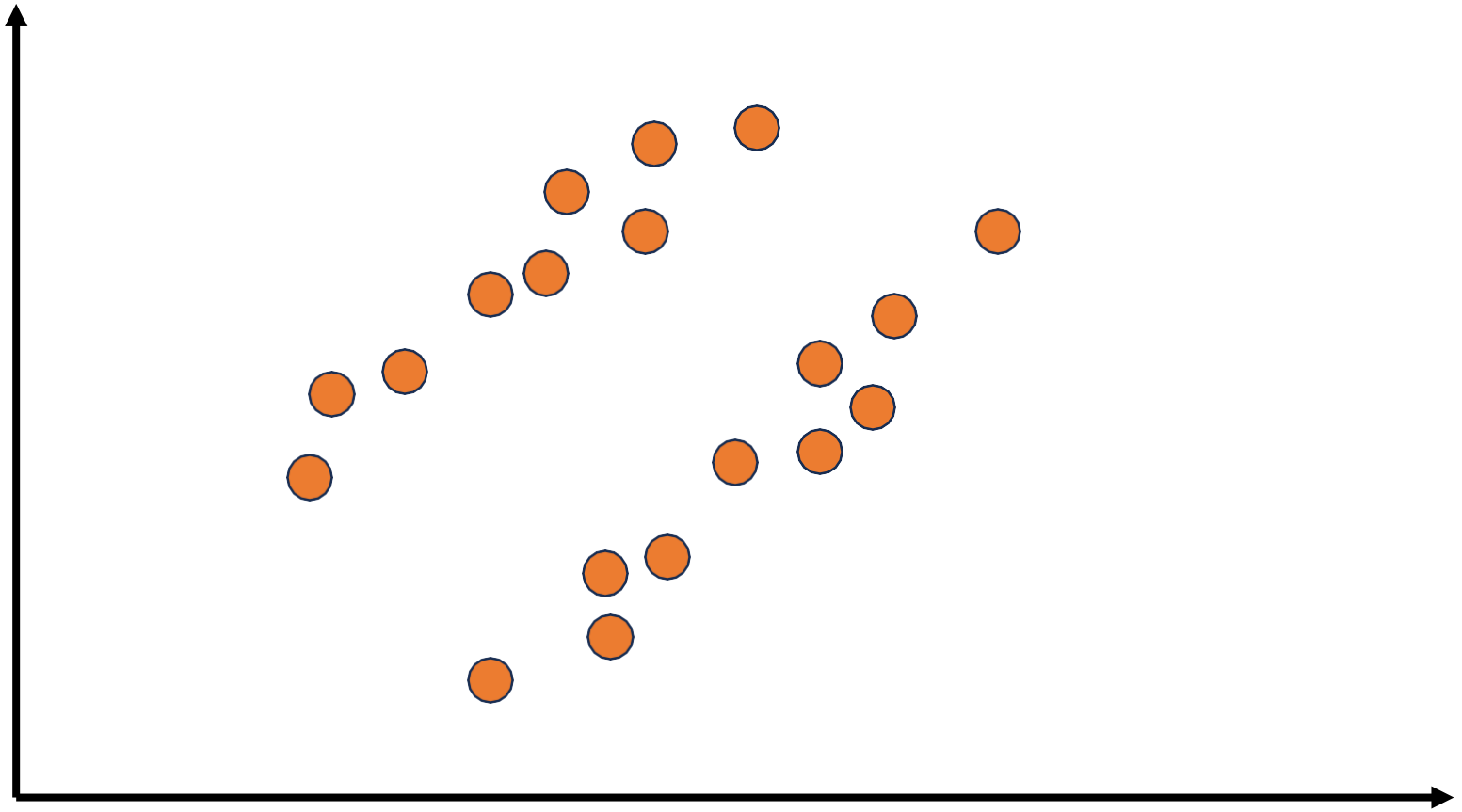
The algorithm needs to determine:

- What the appropriate groups are
- Which elements belong in which group

Example



Example



Unsupervised: Clustering

Clustering

- Grouping unlabeled data based on similarities
- Find patterns
- Sometimes we have minor direction like
  - How many clusters
  - Quality of a good cluster
  - Relationship of clusters

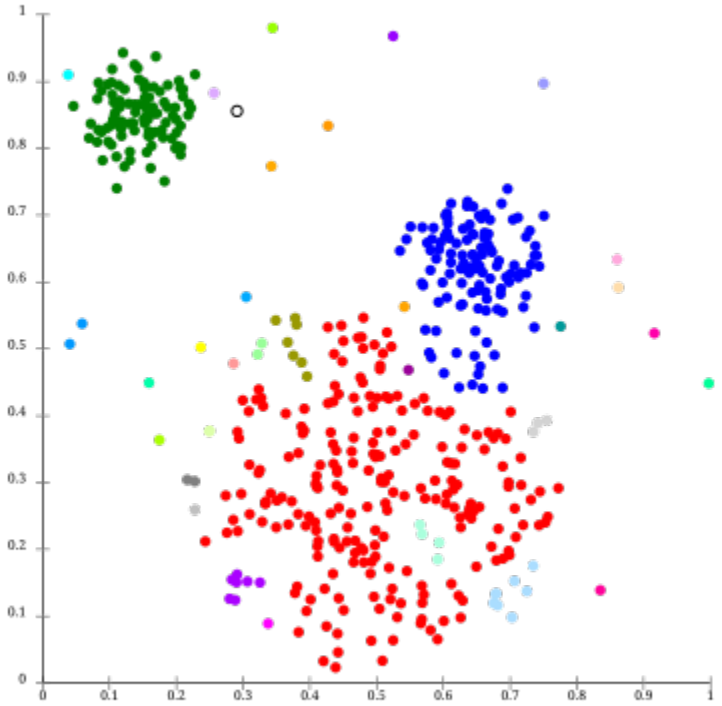
- Most common intro example
  - K-means clustering
    - Ask for k clusters of data and algorithm attempts to create them
  - Often usage has you run k-means with differing sizes and contrasting results

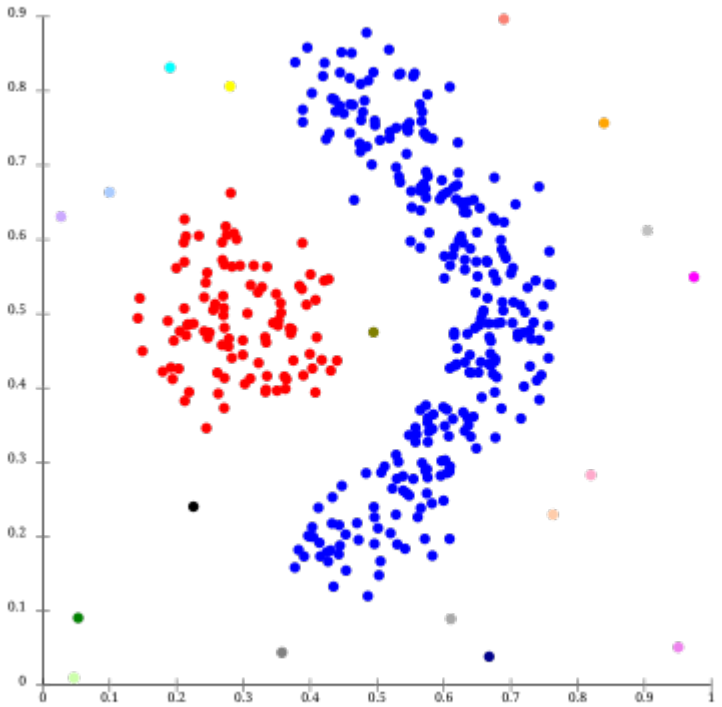
Clustering Usage

- Usages
  - Grouping similar data points
    - Locality
    - Similar properties
    - Similar outcomes
  - Customer segmentation -> Target advertisements
  - Medical imaging -> Find areas that share a property of note
  - Social network analysis - > Who is friends with who
  - Security -> Anomaly detection

Clustering Goals

- Clusters goal may have a desired shape
  - Ex. Circular
- Or maybe not be restrained to a particular shape
- Goal can also be to force every point to be part of a cluster (hard clustering), or soft where goal is points end up with likelihood of membership





Clustering Categories

- Centroid based Clustering
  - Dictated count of clusters (k), group data based on similarity measure, ex. Euclidean
  - Need methods to guess a good k (scientifically!)
  - Circular centroid relativism (k-means)
- Density based Clustering
  - Density of cluster, irregular shapes much easier
- Connectivity based Clustering
  - Hierarchical, make strong small clusters and then merge them to most similar other clusters
- Distribution based Clustering
  - Group things based on behaviour related to a distribution, similar to density but density decision points have relationship to a more flexible distribution

K-Means

Initialize k **means** with random values

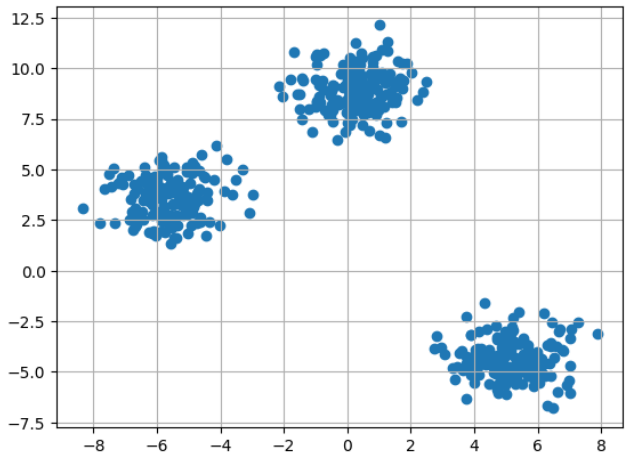
While more iterations left:

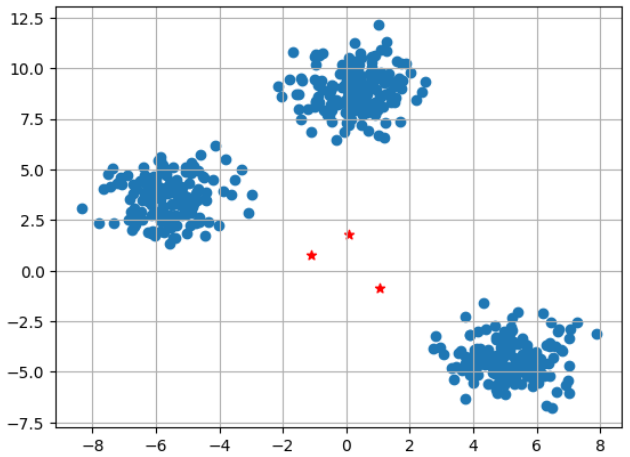
Loop through all the items

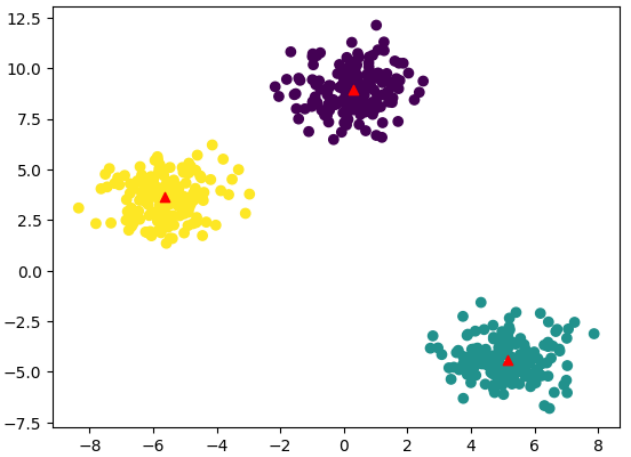
Find the **mean** closest to that item

Assign item to cluster of that **mean**

Update **mean** by shifting it to average of its cluster

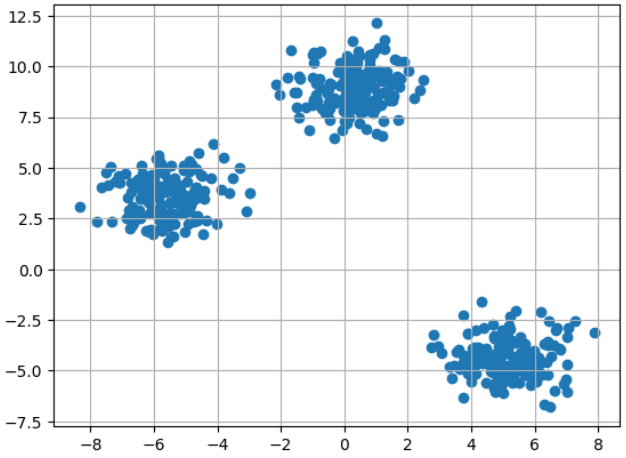






Not too hard to do yourself

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
#Make data
X,y = make_blobs(n_samples = 500,n_features = 2,centers = 3,random_state = 23)
#Plot data
fig = plt.figure(0)
plt.grid(True)
plt.scatter(X[:,0],X[:,1])
plt.show()
```



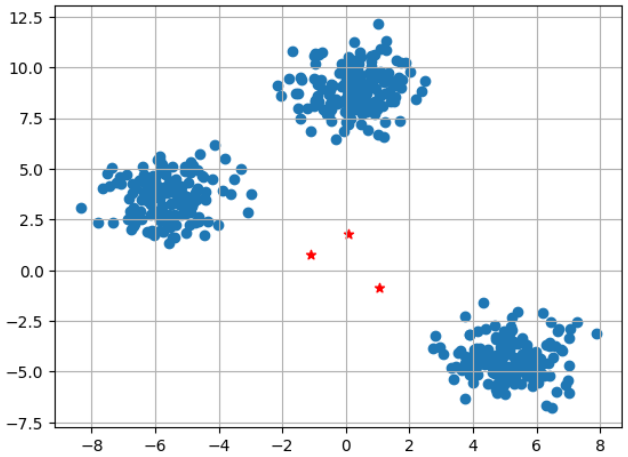
https://www.geeksforgeeks.org/k-means-clustering-introduction/

Make random clusters

```
k = 3
clusters = {}
np.random.seed(23)
for idx in range(k):
    center = 2*(2*np.random.random((X.shape[1],))-1)
    points = []
    cluster = {
        'center' : center,
        'points' : []
    }
    clusters[idx] = cluster
```

Plot clusters

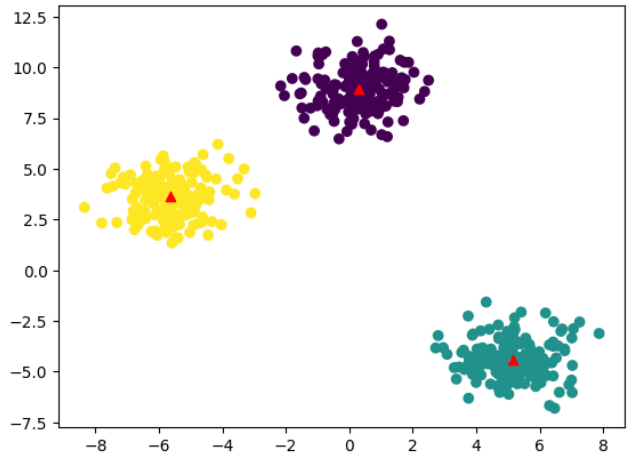
```
#Plot points
plt.scatter(X[:,0],X[:,1])
plt.grid(True)
#Plot cluster means (the random ones)
for i in clusters:
    center = clusters[i]['center']
    plt.scatter(center[0],center[1],marker = '*',c = 'red')
plt.show()
```



Plot clusters

```
clusters = assign_clusters(X,clusters) #Put points in X into clusters
clusters = update_clusters(X,clusters) #Update cluster means
pred = pred_cluster(X,clusters)        #Points -> cluster, for plotting coloured clusters


#Plot points
plt.scatter(X[:,0],X[:,1],c = pred)
#Plot means
for i in clusters:
    center = clusters[i]['center']
    plt.scatter(center[0],center[1],marker = '^',c = 'red')
plt.show()
```



Sci Kit (tutorials)

```
from sklearn.cluster import Kmeans

# Make model

# Fit model

# Use model to predict y (pred)

#Plot data is the same
plt.scatter(X[:,0],X[:,1],c = pred)
#We use use kmeans cluster data
for i in kmeans.cluster_centers_:
    plt.scatter(i[0],i[1],marker = '^',c = 'red')
plt.show()
```

Unsupervised: Association Rules Learning

Association Rule Learning

- Finding relationships in data (implications)
  - If this then that

- Market Analysis is common
  - If you bought milk, then you also bought cookies

  - So we will arrange those items in store to create desired behaviour
  - For example put them at locations in store that require you to traverse it and be exposed to more purchase choices

Apriori Algorithm

- Agrawal & Srikant 1994
  - Find frequent itemsets
  - Boolean association rule mining
  - Finds an answer to size k, to then find size k+1 (apriori)

- Apriori property
  - All subsets g of a frequent itemset f (g subset of f)
    - Must be frequent themselves
- Lemma
  - If an itemset g is infrequent then all its supersets (g superset of f)
    - Are infrequent

Association Rule Learning

- Input

TRANS.	Item1	Item2	Item3	Item4	Item5
1	YES	YES			YES
2		YES		YES	
3		YES	YES		
4	YES	YES		YES	
5	YES		YES		
6		YES	YES		
7	YES		YES		
8	YES	YES	YES		YES
9	YES	YES	YES		

Association Rule Learning

- Input

TRANS.	Item1	Item2	Item3	Item4	Item5
1	YES	YES			YES
2		YES		YES	
3		YES	YES		
4	YES	YES		YES	
5	YES		YES		
6		YES	YES		
7	YES		YES		
8	YES	YES	YES		YES
9	YES	YES	YES		

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

Association Rule Learning

- Input

TRANS.	Item1	Item2	Item3	Item4	Item5
1	YES	YES			YES
2		YES		YES	
3		YES	YES		
4	YES	YES		YES	
5	YES		YES		
6		YES	YES		
7	YES		YES		
8	YES	YES	YES		YES
9	YES	YES	YES		

- Requirements
  - Support count minimum: 2
  - Confidence: 60%
- K = 1

Item	Count
1	6
2	7
3	6
4	2
5	2

Association Rule Learning

- Input

TRANS.	Item1	Item2	Item3	Item4	Item5
1	YES	YES			YES
2		YES		YES	
3		YES	YES		
4	YES	YES		YES	
5	YES		YES		
6		YES	YES		
7	YES		YES		
8	YES	YES	YES		YES
9	YES	YES	YES		

- Requirements
  - Support count minimum: 2
  - Confidence: 60%
- K = 1

Item	Count
1	6
2	7
3	6
4	2
5	2

- All pass minimum support count so move to K=2

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%
- K = 1

Item(1)	Count
1	6
2	7
3	6
4	2
5	2

- K=2

Item(2)	Count
12	
13	
14	
15	
23	
24	
25	
34	
35	
45	

Join prior size
Each subset should have K-2 elements in common
So for here 0 in common

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

TRANS.	Item1	Item2	Item3	Item4	Item5
1	YES	YES			YES
2		YES		YES	
3		YES	YES		
4	YES	YES		YES	
5	YES		YES		
6		YES	YES		
7	YES		YES		
8	YES	YES	YES		YES
9	YES	YES	YES		

- K=2

Item(2)	Count
12	4
13	4
14	1
15	2
23	4
24	2
25	2
34	0
35	1
45	0

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

- K=2

Item(2)	Count
12	4
13	4
14	1
15	2
23	4
24	2
25	2
34	0
35	1
45	0

- All subsets of these itemsets are in K=1

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

- K=2

Item(2)	Count
12	4
13	4
15	2
23	4
24	2
25	2

- All subsets of these itemsets are in K=1
- Shorten to minimum support of 2

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%
- K = 2

Item(2)	Count
12	4
13	4
15	2
23	4
24	2
25	2

- K=3

Item(3)	Count
123	
124	
125	
135	
234	
235	
245	

Join prior size
Each subset should have K-2 elements in common
So for here 1 in common

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

TRANS.	Item1	Item2	Item3	Item4	Item5
1	YES	YES			YES
2		YES		YES	
3		YES	YES		
4	YES	YES		YES	
5	YES		YES		
6		YES	YES		
7	YES		YES		
8	YES	YES	YES		YES
9	YES	YES	YES		

- K=3

Item(3)	Count
123	2
124	1
125	2
135	1
234	0
235	1
245	0

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

- K=3

Item(3)	Count
123	2
124	1
125	2
135	1
234	0
235	1
245	0

- All subsets of these itemsets are in K=2

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%

- K=3

Item(3)	Count
123	2
125	2

- All subsets of these itemsets are in K=2
- Shorten to minimum support of 2

Association Rule Learning

- Requirements
  - Support count minimum: 2
  - Confidence: 60%
- K = 3

Item(3)	Count
123	2
125	2

- K=4

Item(4)	Count
1235	1

Join prior size
Each subset should have K-2 elements in common
So for here 2 in common

- Below minimum support count of 2
- So we would remove and since we have none left at K=4 we stop algorithm

- But we aren't done
- Confidence 60% needed

Association Rule Learning

- Remember we are looking for association rules
  - So far we have these two tables on the right
- These are just all bought at same time
- But we want to know if user buys 1 if they also buy 2
- Or if user buys 12 do they also buy 3
- Or if user buys 1 do they also buy 23
- Etc.
- So how do we get these

Item(2)	Count
12	4
13	4
15	2
23	4
24	2
25	2

Item(3)	Count
123	2
125	2

Association Rule Learning

- Confidence(A->B)=
  - Support_count(AUB) / Support_count(A)
- Confidence(1->2)=
  - Support_count(12) / Support_count(1)
  - 4/6
  - =2/3=66.6%
- So this rule is one we would keep
- Let’s try a bigger rule

Item(2)	Count
12	4
13	4
15	2
23	4
24	2
25	2

Item(3)	Count
123	2
125	2

Association Rule Learning

- Confidence(A->B)=
  - Support_count(AUB) / Support_count(A)
- Confidence(12->3)=
  - Support_count(123) / Support_count(12)
  - 2/4
  - 50%
- Confidence(1->23)=
  - Support_count(123) / Support_count(1)
  - 2/6
  - 33.333%
- Neither of these are exceed our threshold of 60% (0.6)

Item(2)	Count
12	4
13	4
15	2
23	4
24	2
25	2

Item(3)	Count
123	2
125	2

Limitations of Apriori

- Often rather slow with interesting sized dataset
- For example, if there are 10^4 from frequent 1- itemsets, it need to generate more than 10^7 candidates into 2-length which in turn they will be tested and accumulate.
- Furthermore, to detect frequent pattern in size 100 i.e. v1, v2… v100, it have to generate 2^100 candidate itemsets that yield on costly and wasting of time of candidate generation.
- So, it will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets.
- Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.

Sci Kit

- mlxtend library based on ci kit
- Particularly mlxtend.frequent_patterns module

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
dataset = [["item1","item2","item5"],["item2","item4"],…]
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
display(df)
```

Sci Kit

```
count = len(dataset)
threshold = 0.6
frequent_itemsets = apriori(df,
                            min_support=2/count,
                            use_colnames=True)
print(frequent_itemsets)

rules = association_rules(frequent_itemsets,
                                    metric="confidence",
                                    min_threshold=threshold,
                                    num_itemsets=rule_count)
print(rules)
```

Next…supervised learning

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY