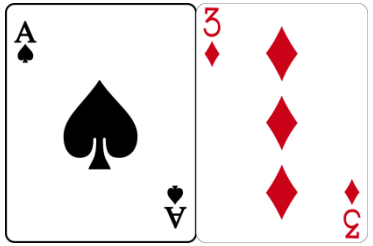
Modul 3

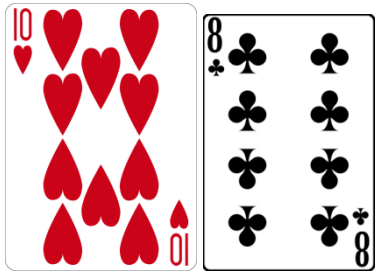
Unsupervised Learning

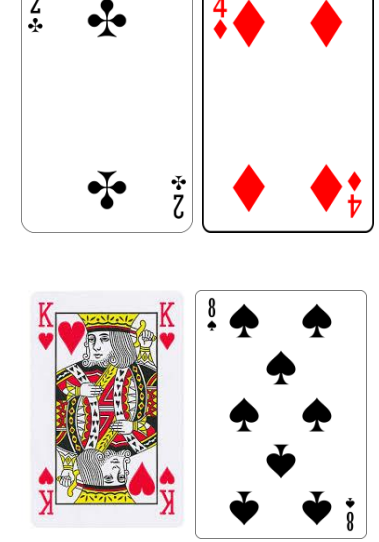
Data Science Program

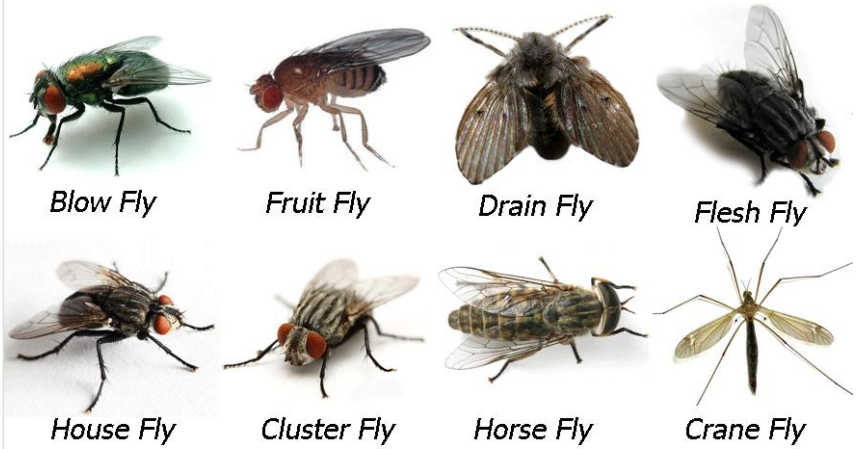
Purwadhika
Startup and Coding School

Supervised vs. unsupervised learning









Purwadhika
Startup and Coding School

Clustering intuitively

Purwadhika
Startup and Coding School

Clustering applications

Customer segmentation (e.g. for cost-benefit analysis of new products)

Topic identification (e.g. to speed up manual vetting)

Image or geo-spatial segmentation (e.g. Gojek's supply-demand optimization)

Maybe most importantly, getting a sense of data prior to in-depth modeling!

Distance measures

Most of clustering method is based on the distance measures. Here below is few formula to calculated the data distance

Euclidean

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

Manhattan

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

Numerical features

Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

Categorical features

Cosine similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

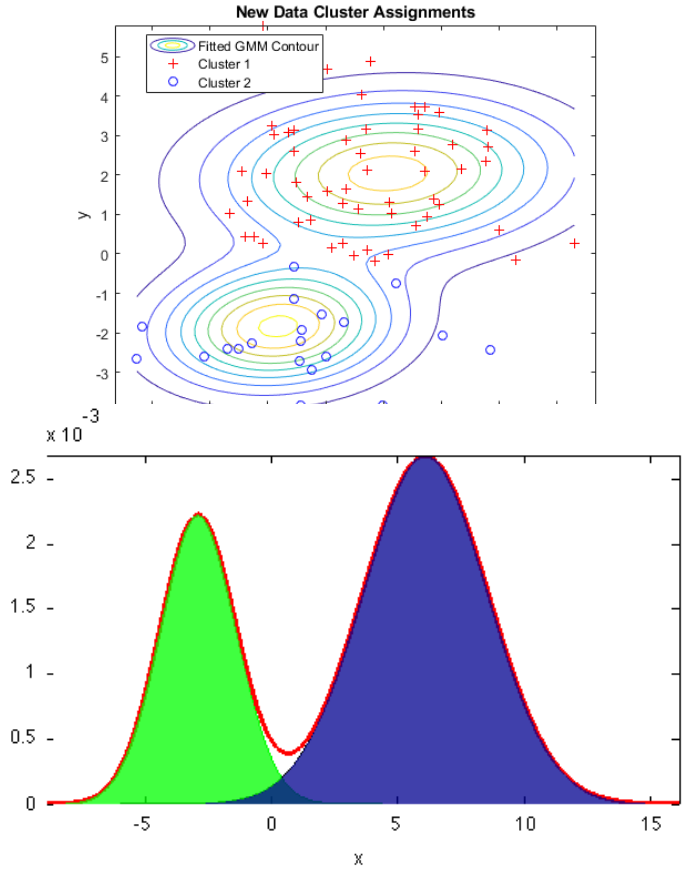
High-dimensional features

Others: correlation, KL divergence, edit distance

Purwadhika
Startup and Coding School

Soft, not hard, partition - Gaussian Mixture Model

Real scenario



We can apply the EM algorithm. We assign random values to parameters Θ as the initial values. We then iteratively conduct the E-step and the M-step as follows until the parameters converge or the change is sufficiently small.

In the **E-step**, for each object, $o_i \in \mathbf{O} (1 \le i \le n)$, we calculate the probability that o_i belongs to each distribution, that is,

$$P(\Theta_j | o_i, \Theta) = \frac{P(o_i | \Theta_j)}{\sum_{t=1}^{k} P(o_i | \Theta_t)}. \quad (11.13)$$

In the **M-step**, we adjust the parameters Θ so that the expected likelihood $P(\mathbf{O} | \Theta)$ in Eq. (11.11) is maximized. This can be achieved by setting

$$\mu_j = \frac{1}{k} \sum_{i=1}^{n} o_i \frac{P(\Theta_j | o_i, \Theta)}{\sum_{t=1}^{n} P(\Theta_j | o_i, \Theta)} = \frac{1}{k} \frac{\sum_{i=1}^{n} o_i P(\Theta_j | o_i, \Theta)}{\sum_{t=1}^{n} P(\Theta_j | o_i, \Theta)} \quad (11.14)$$

and

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{n} P(\Theta_j | o_i, \Theta)(o_i - u_j)^2}{\sum_{t=1}^{n} P(\Theta_j | o_i, \Theta)}}. \quad (11.15)$$

https://brilliant.org/wiki/gaussian-mixture-model/

Purwadhika
Startup and Coding School

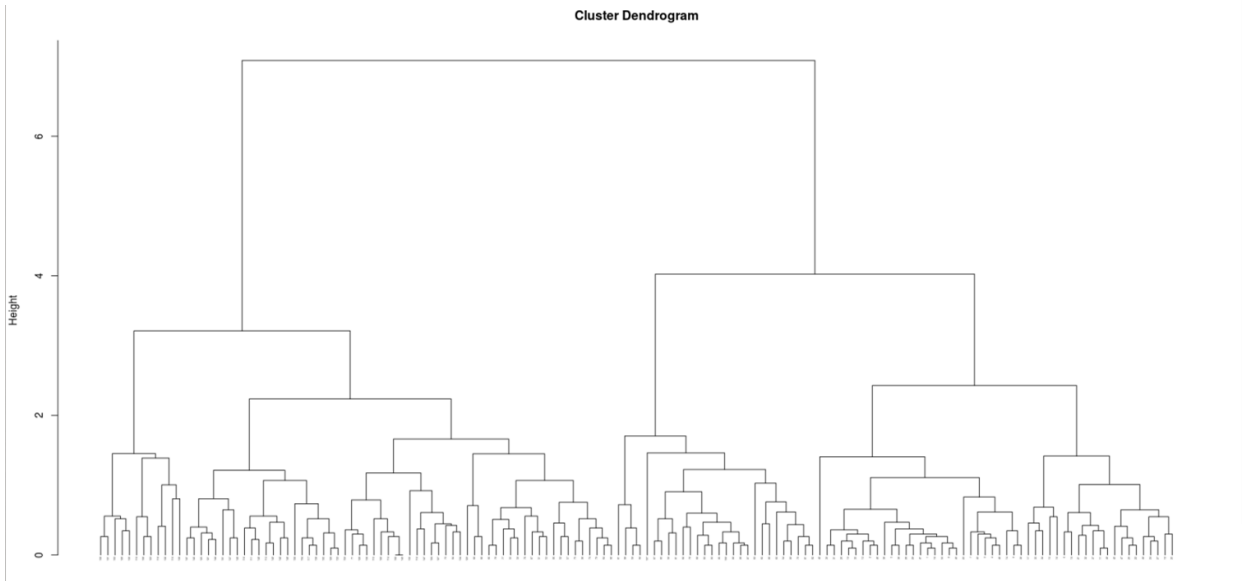
Gaussian Mixture Model (GMM)

- GMM is a probabilistic model that represent normally distributed sub-population within the overall population.
- It belong to the mixture model, means this model does not need to know which sub-population class it belongs to because the model could learn the sub-population automatically.

Hierarchical Clustering

- Hierarchical clustering is an algorithm that group observed data into cluster
- Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps: (1) identify the two clusters that are closest together, and (2) merge the two most similar clusters. This continues until all the clusters are merged together.
- The end result of the hierarchical clustering is dendrogram

Sensing the number of clusters - Hierarchical Clustering



Minimum distance : $dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{| p - p' |\}$

Maximum distance : $dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{| p - p' |\}$

Mean distance : $dist_{mean}(C_i, C_j) = | m_i - m_j |$

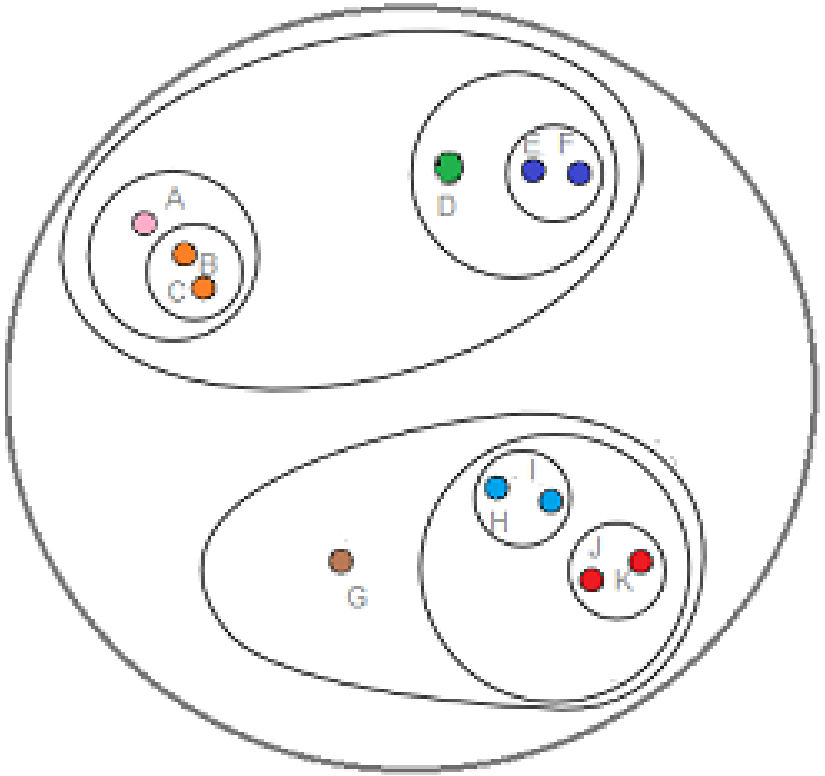
Average distance : $dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} | p - p' |$

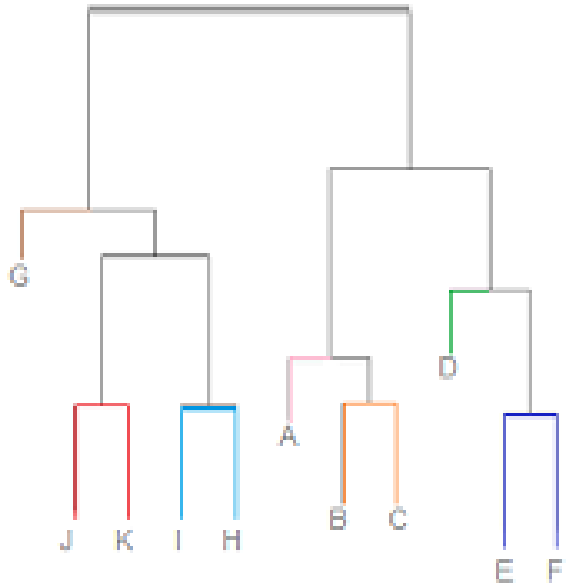
Clustering is based on the distance, often we use Euclidean Distance although we could decide which distance measurement to used based on the user justification

After establish the distance measurement, we need to decide where the distance is measured. For example, it can be computed between the two most similar parts of a cluster (*single-linkage*), the two least similar bits of a cluster (*complete-linkage*), the center of the clusters (*mean* or *average-linkage*), or some other criterion.

Purwadhika
Startup and Coding School

Hierarchical Clustering

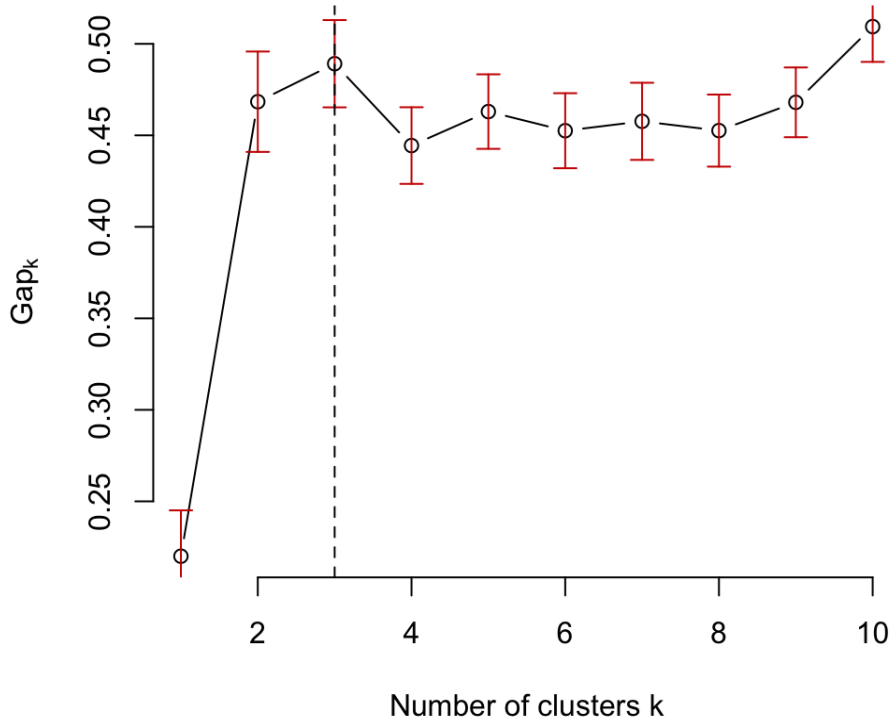




Purwadhika
Startup and Coding School

Validation - Finding the number of clusters more systematically (Silhouette index)

- **Key principle:** minimum intra-cluster distance, maximum intra-cluster distance



- Silhouette index is a clustering validation technique.
- This technique measure the similarity of the data within their own cluster compared to the other cluster.
- The measurement range from +1 to -1 with higher score mean the data is better fit their cluster compared to the other cluster.
- Usually this technique is used alongside graphical representation

Purwadhika
Startup and Coding School

How Silhouette index calculated

where s(i): Silhoutte index

where a(i) :average of intra-cluster(between the data point(i) to the other data between the cluster C_i) distance. d(i,j) is the distance between data point i to j. The smaller the a(i) value, the better fit it is.

where b(i): the smallest mean distance from data point i to all closest object from another cluster, where i is not the member.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

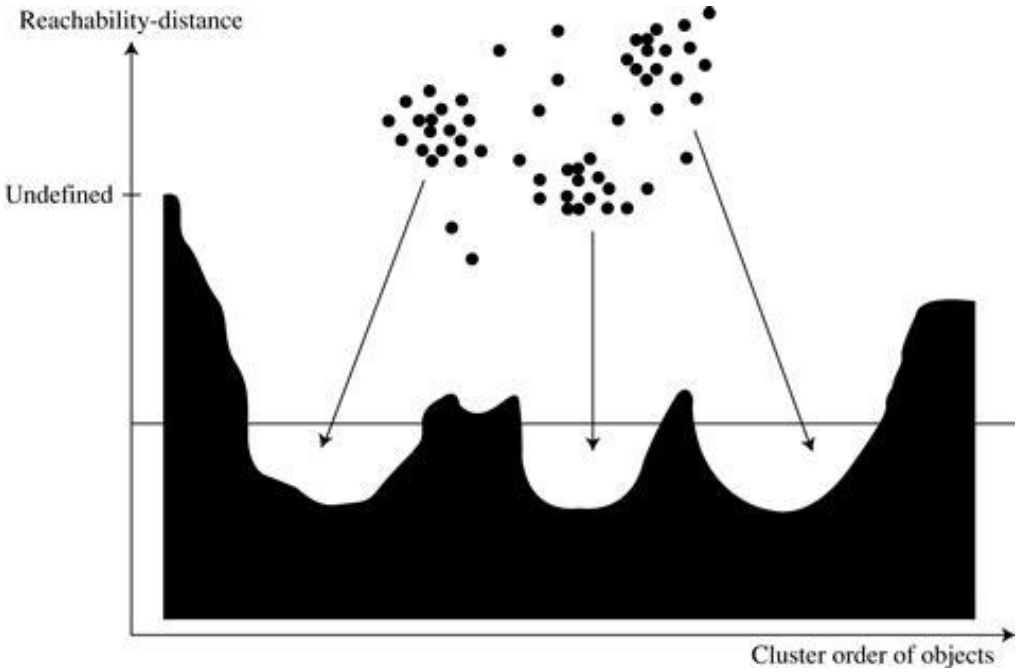
$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

Purwadhika
Startup and Coding School

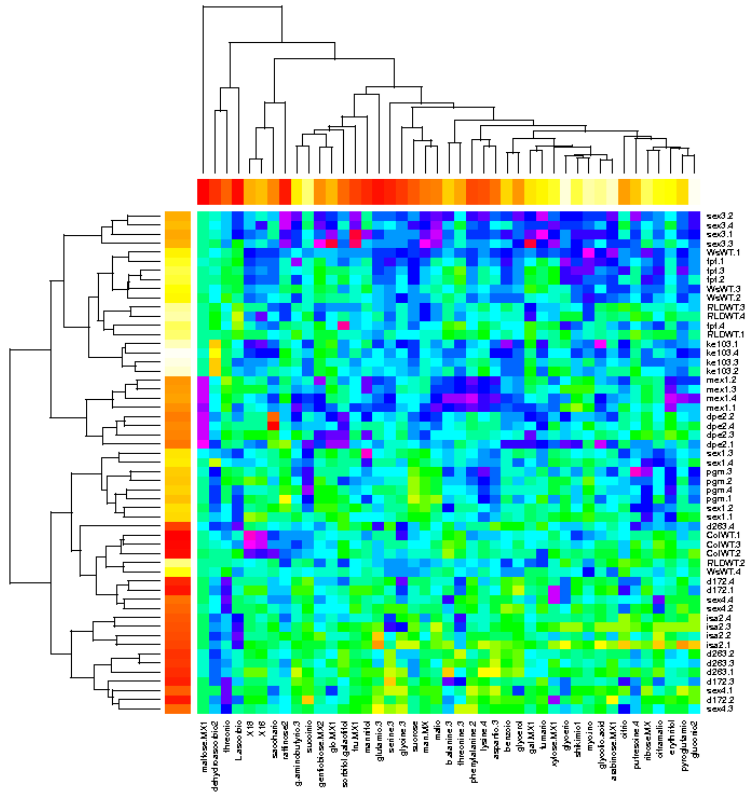
Other clustering variants: density and co-clustering

Density-based clustering (dealing with outliers)

DBScan, OPTICS



Co-clustering or biclustering (dealing with high-dimensional features)

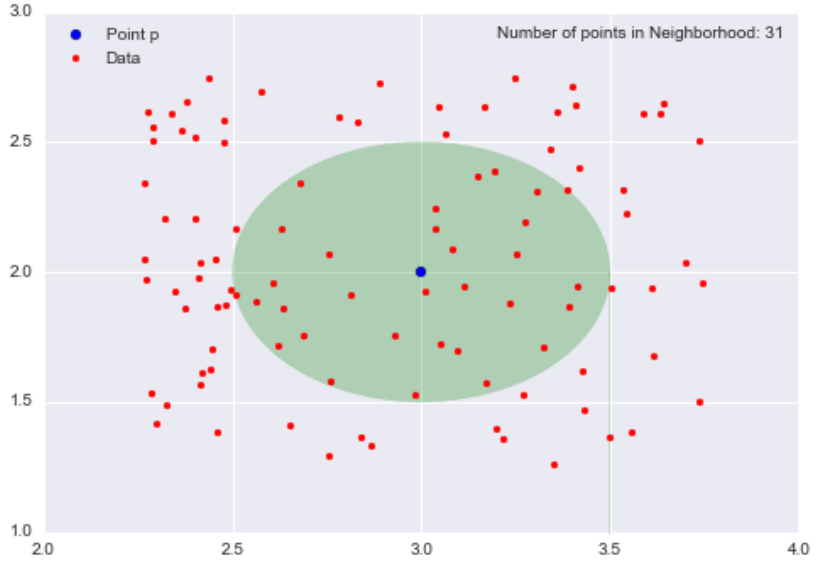


adhika
l Coding School

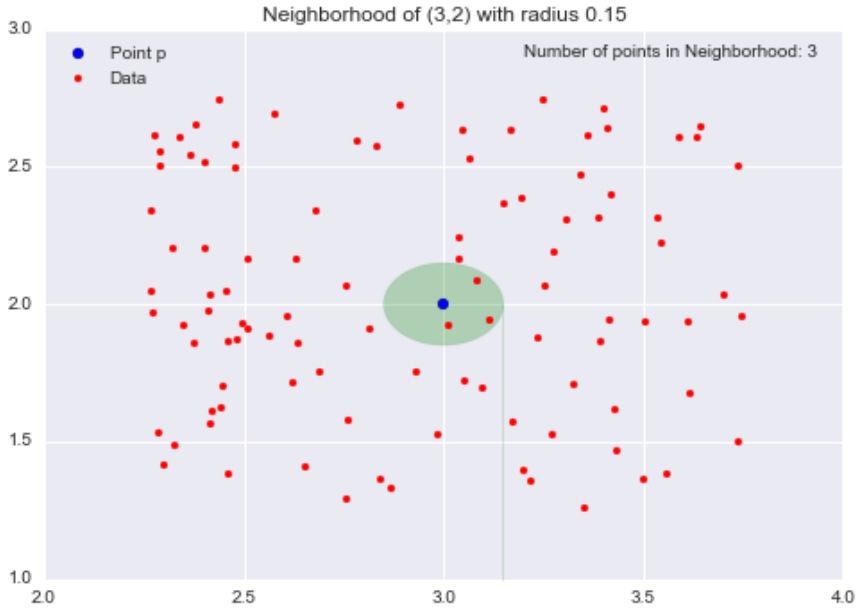
Density-Based Spatial Clustering of Applications with Noise (DBScan)

- Density-based clustering work by identifying ‘dense’ cluster of points, this allow the model to learn the cluster arbitrary shape and identifying outlier
- To approximate the dense cluster point, DBScan rely on the concept called ε-neighborhoods.
- ε-neighborhoods could be described as a data point that in 2D space, the ε-neighborhood of a point *p* is the set of points contained in a circle of radius ε, centered at *p.*

ε-neighborhood



Example of ε-neighborhood in point P(3,2) with radius ε = 0.5. Resulted in 31 points in Neighborhood.



Example of ε-neighborhood in point P(3,2) with radius ε = 0.15. Resulted in 3 points in Neighborhood.

Purwadhika
Startup and Coding School

DBScan

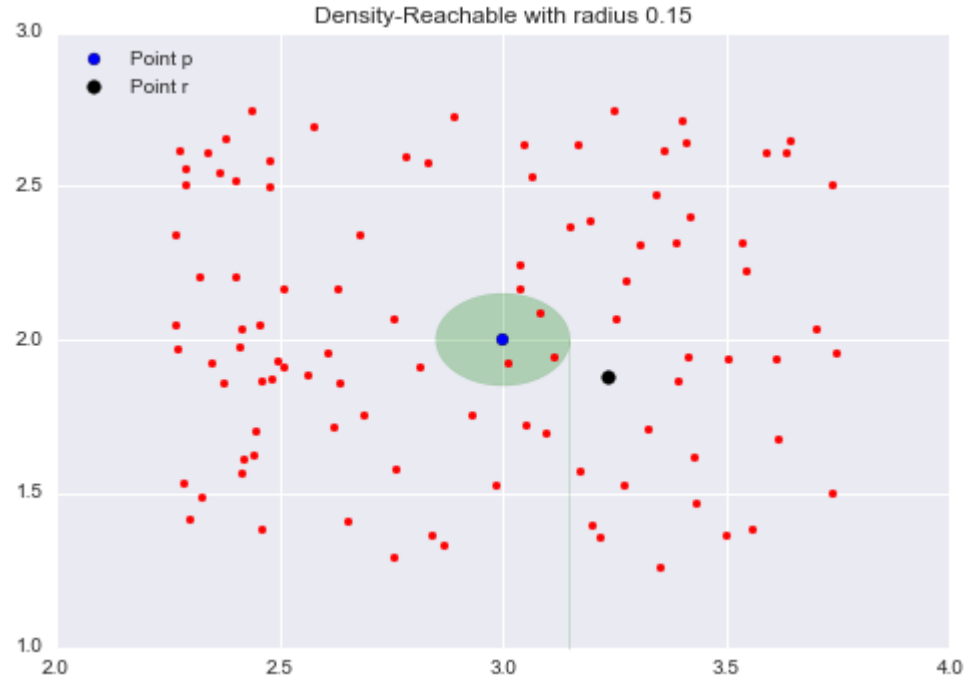
- When applying the DBScan model, we need to set 2 parameters that is:
  - ε: The radius of the neighborhoods around a data point *p*.
  - *minPts*: The minimum number of data points in the neighborhood to define a cluster.
- Using these two parameters, DBSCAN categories the data points into three categories:
  - *Core Points*: A data point *p* is a *core point* if **Nbhd**(*p*,ε) [ε-neighborhood of *p*] contains at least *minPts* ; |**Nbhd**(*p*,ε)| >= *minPts*.
  - *Border Points: A data point *q* is a border point* if **Nbhd**(*q*, ε) contains less than *minPts* data points, but *q* is *reachable* from some *core point p*.
  - *Outlier*: A data point *o* is an *outlier* if it is neither a core point nor a border point. Essentially, this is the “other” class.

DBScan Core Points

- Core points are data points that satisfy a minimum density requirement (*minPts)*.
- Clusters are built around our *core points* (hence the *core* part), by adjusting the *minPts* parameter, we can fine-tune how dense our clusters cores must be.

DBScan Border Points

- *Border Points* are the points in clusters that are not core points. It is *density-reachable*. To explain this concept, let's revisit the neighborhood example with ε = 0.15. Consider the point *r* (the black dot) that is outside of the point *p*'s neighborhood.

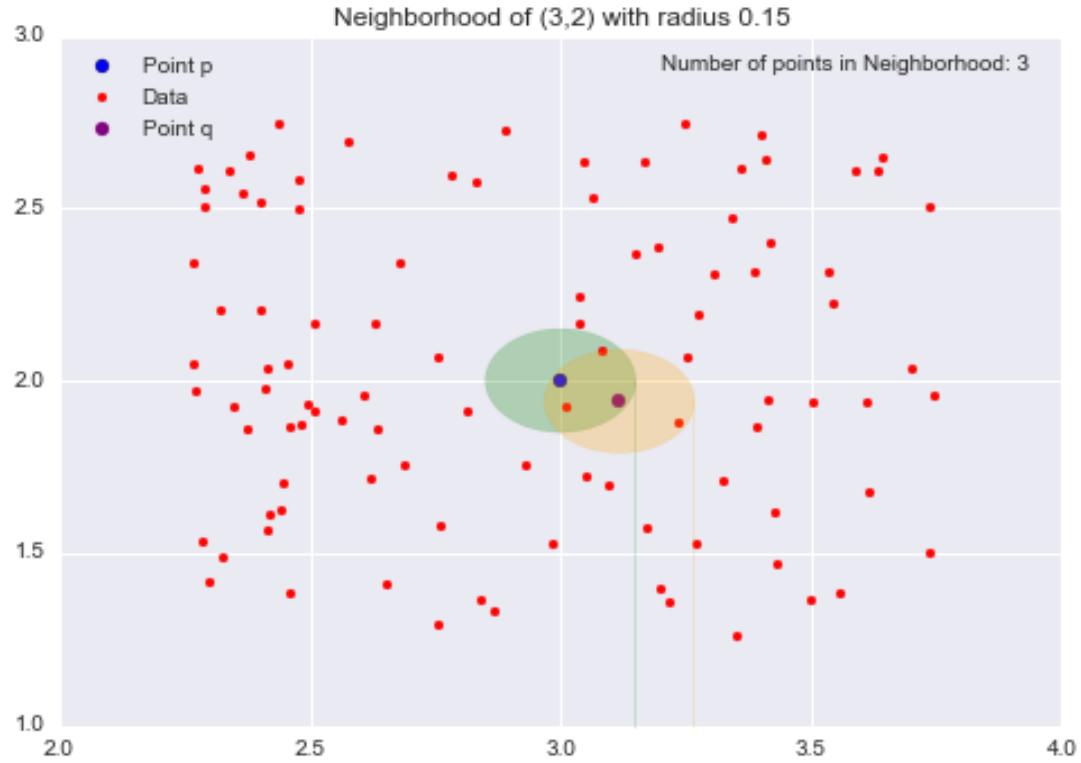


All the points inside the point *p*'s neighborhood are said to be *directly reachable* from *p*.

Purwadhika
Startup and Coding School

DBScan Border Points

- The neighborhood of point *q*, is a point *directly reachable* from *p*. The yellow circle represents *q*‘s neighborhood.

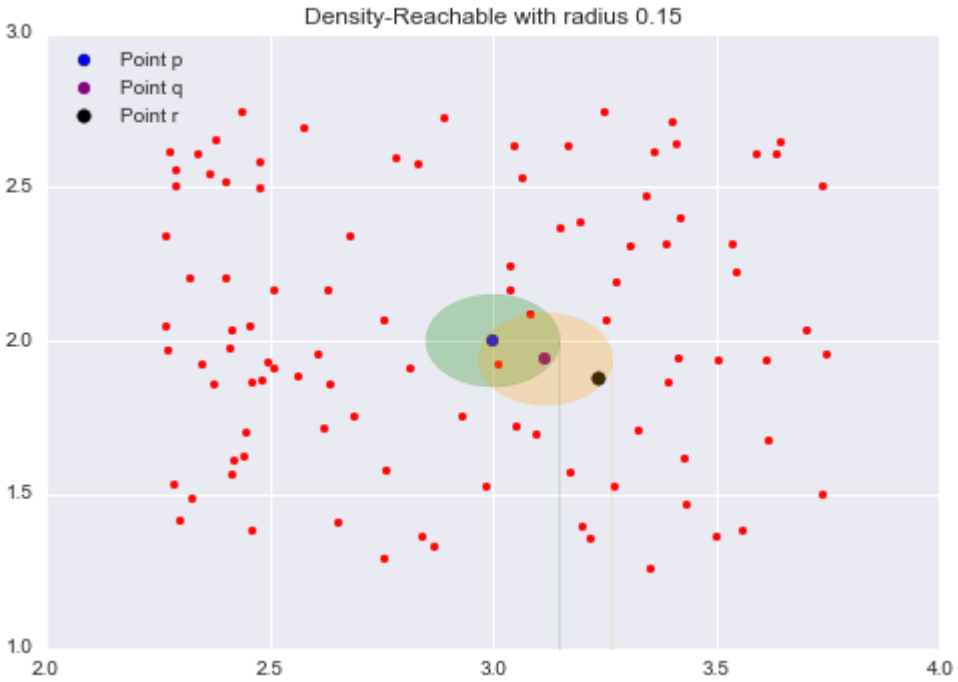


Now while our target point *r* is not our starting point *p*‘s neighborhood, it is contained in the point *q*‘s neighborhood. This is the idea behind *density-reachable*: If I can get to the point *r* by jumping from neighborhood to neighborhood, starting at a point *p*, then the point *r* is density-reachable from the point *p*.

Purwadhika
Startup and Coding School

DBScan Border Points

- It would go on until no more neighborhood jump could be done.
- The jump is not limited to the closest point, as long as the point is density-reachable, it is possible to do the neighborhood jump.



Purwadhika
Startup and Coding School

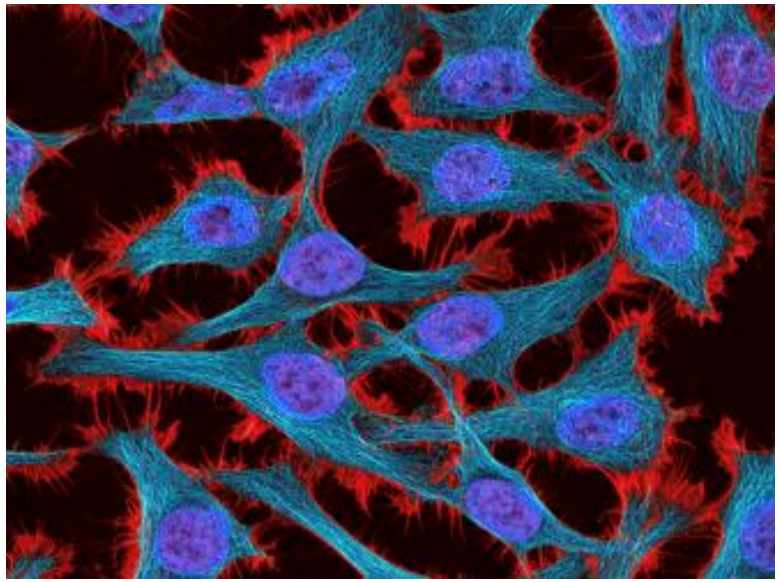
DBScan Outlier

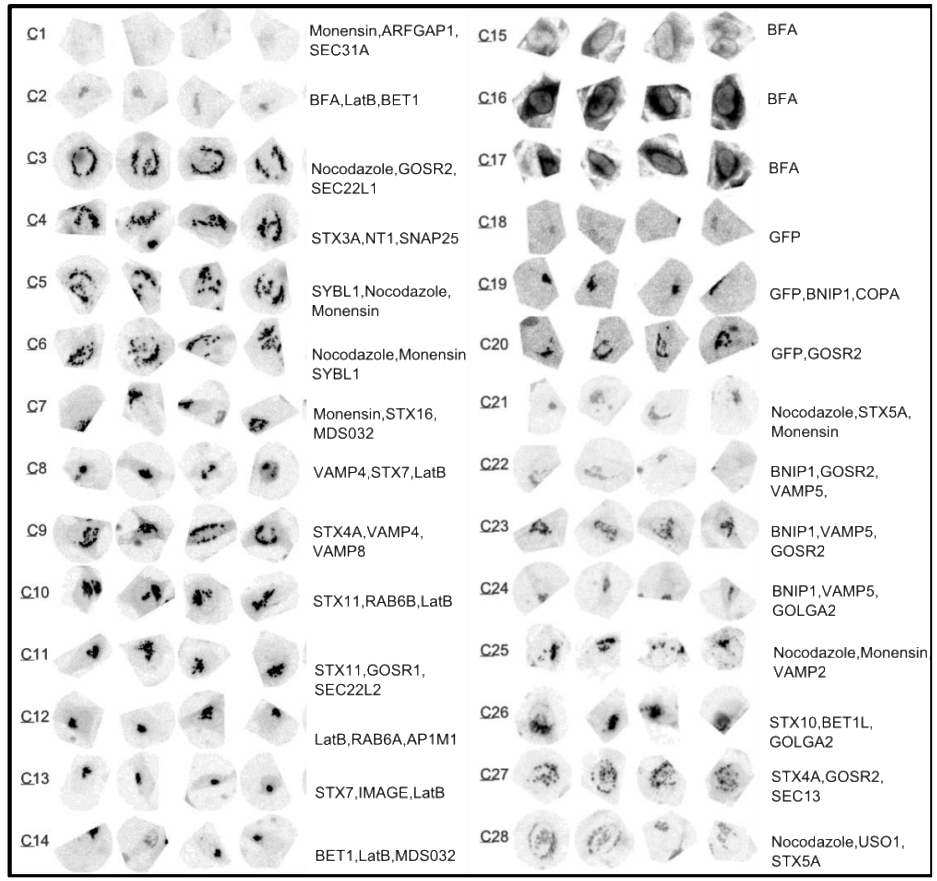
- *Outliers* are points that are neither *core points* nor are they close enough to a cluster to be *density-reachable* from a *core point*. *Outliers* are not assigned to any cluster and, depending on the context, may be considered anomalous points.

DBSCAN algorithm steps

- The steps to the DBSCAN algorithm are:
  - Pick a point at random that has not been assigned to a cluster or been designated as an *outlier*. Compute its neighborhood to determine if it's a *core point*. If yes, start a cluster around this point. If no, label the point as an *outlier*.
  - Once *core point* found and thus a cluster, expand the cluster by adding all *directly-reachable* points to the cluster. Perform "neighborhood jumps" to find all *density-reachable* points and add them to the cluster. If an *outlier* is added, change that point's status from *outlier* to *border point*.
  - Repeat these two steps until all points are either assigned to a cluster or designated as an *outlier*.

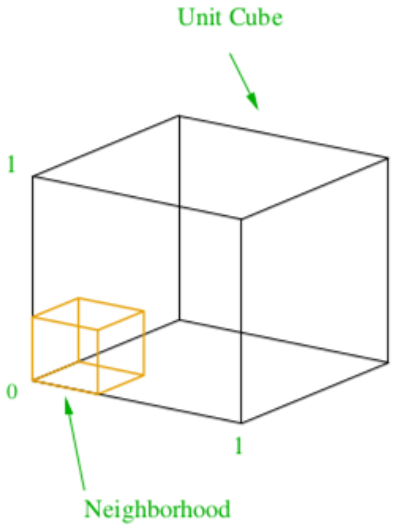
Real world clustering is often messy

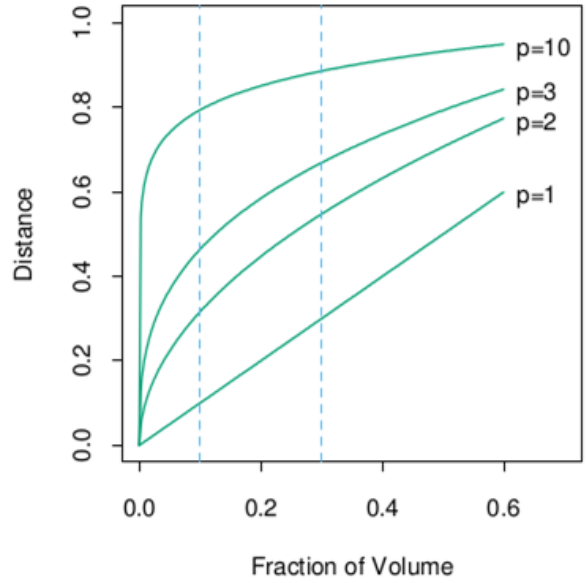




Purwadhika
Startup and Coding School

The problems with high-dimensionality

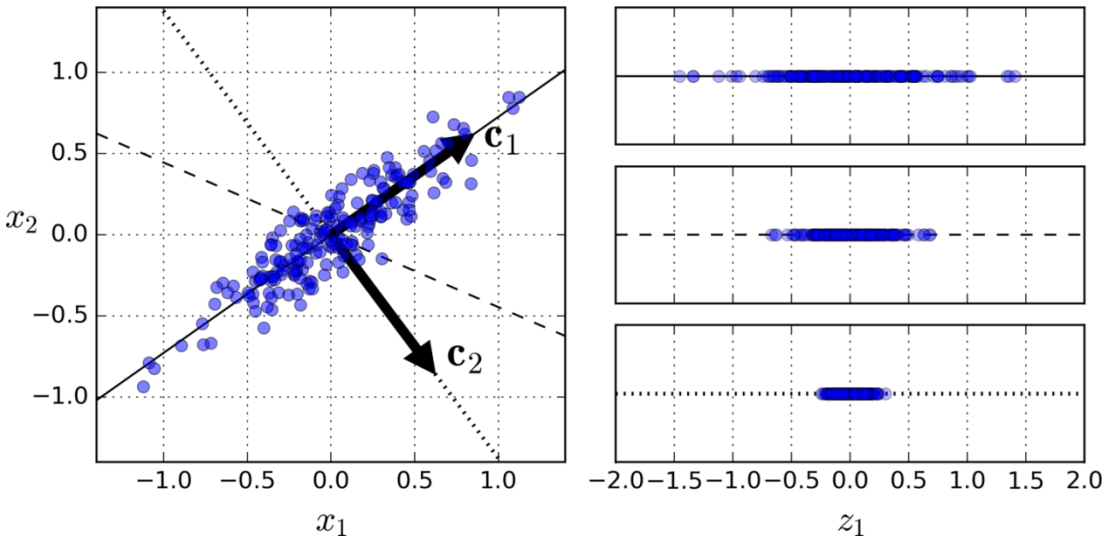


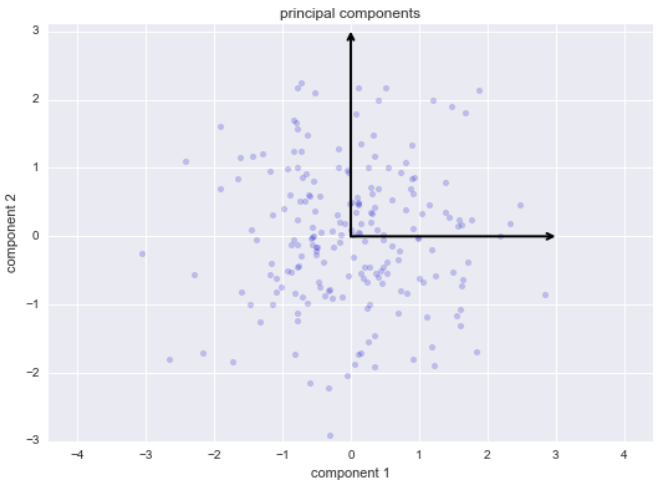


Hard to visualise, some dimensions are noisy, some dimensions are correlated, sparsity requiring impractical volume of training data and strange distance measure behavior (http://mlwiki.org/index.php/Euclidean_Distance)

Purwadhika
Startup and Coding School

Principal component analysis





Optimization problem: project x but try to maximize the variance

Setting the gradient to 0 gives eigenvalue equation:

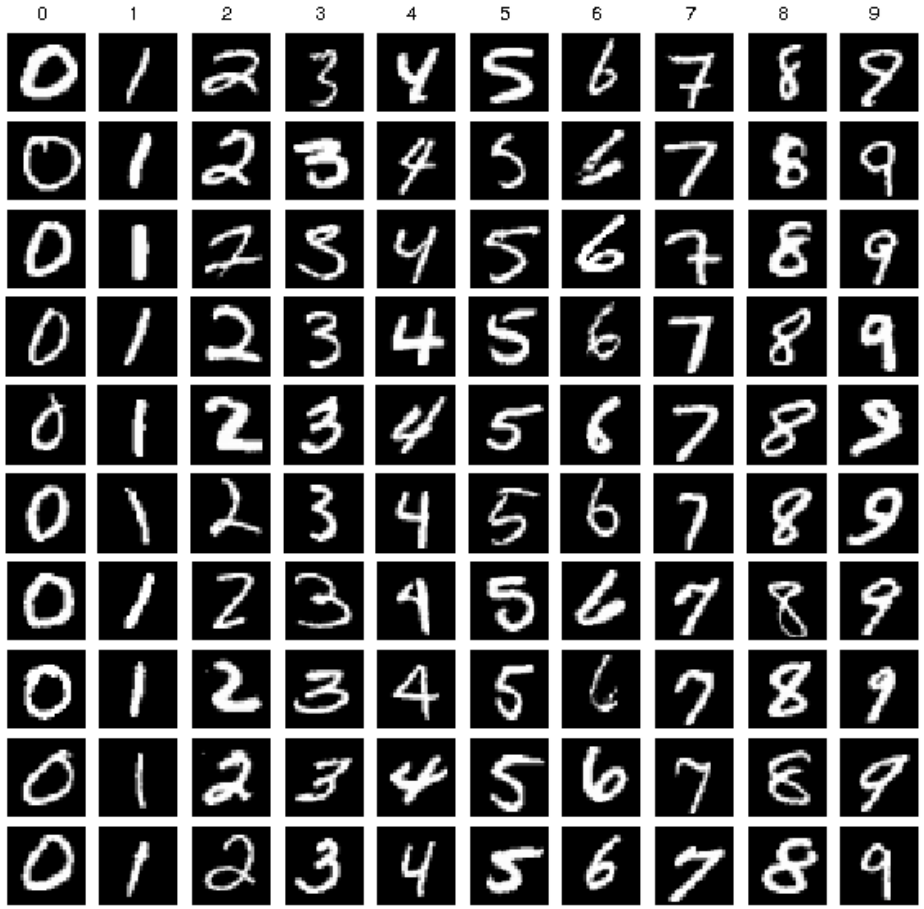
$\Sigma a1 = \lambda 1 a1$

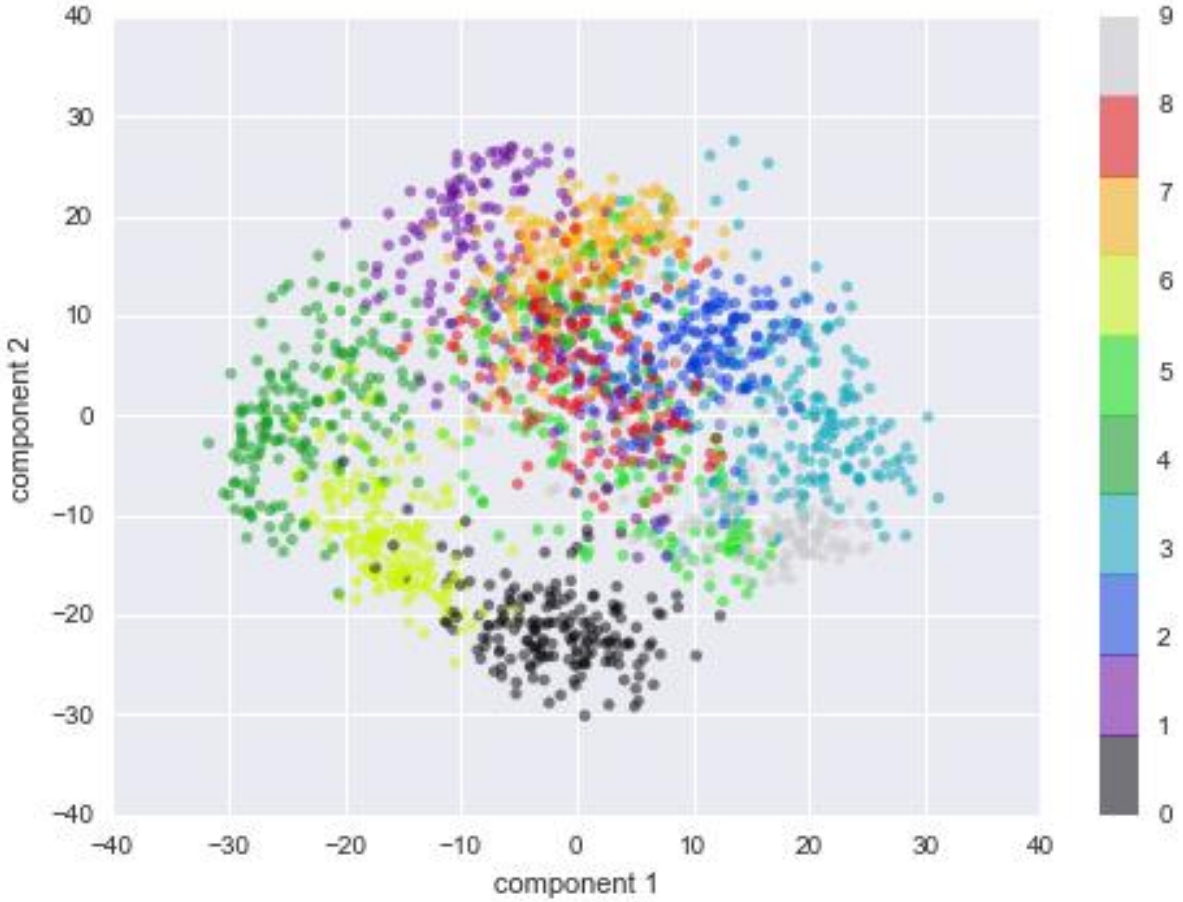
i.e. projecting x in the direction of the eigenvector with the highest eigenvalue gives the highest variance

Repeat the process for the second axis to project results in eigenvector corresponding to the second highest eigenvalue, and so on …

These eigenvectors are called the principal components

Demo PCA on Digits





Purwadhika
Startup and Coding School

What is association rules mining?

It is a rule-based machine learning which focus in finding frequent co-occurring associations among a collection of items. It is sometimes referred to as “Market Basket Analysis”.

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.3		

Association rules mined

	antecedants	consequents	support	confidence	lift
8	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.888889	6.968889
9	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.127551	0.960000	6.968889
10	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.815789	8.642959
11	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.837838	8.642959
16	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.122449	0.812500	6.125000
17	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.975000	7.644000
18	(SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.975000	7.077778
22	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.800000	6.030769

Association rules mining steps:

Transaction table ->
Frequent itemsets (support) ->
Association rules (confidence)

https://mapr.com/blog/association-rule-mining-not-your-typical-data-science-algorithm/

Support, confidence, lift

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Support (A -> B): **P(A U B)**

Is it rare? (indication of how frequently the itemset appears in the dataset)

Confidence (A -> B): **P(B | A)** *How strong is the rule?* (indication of how often the rule has been found to be true)

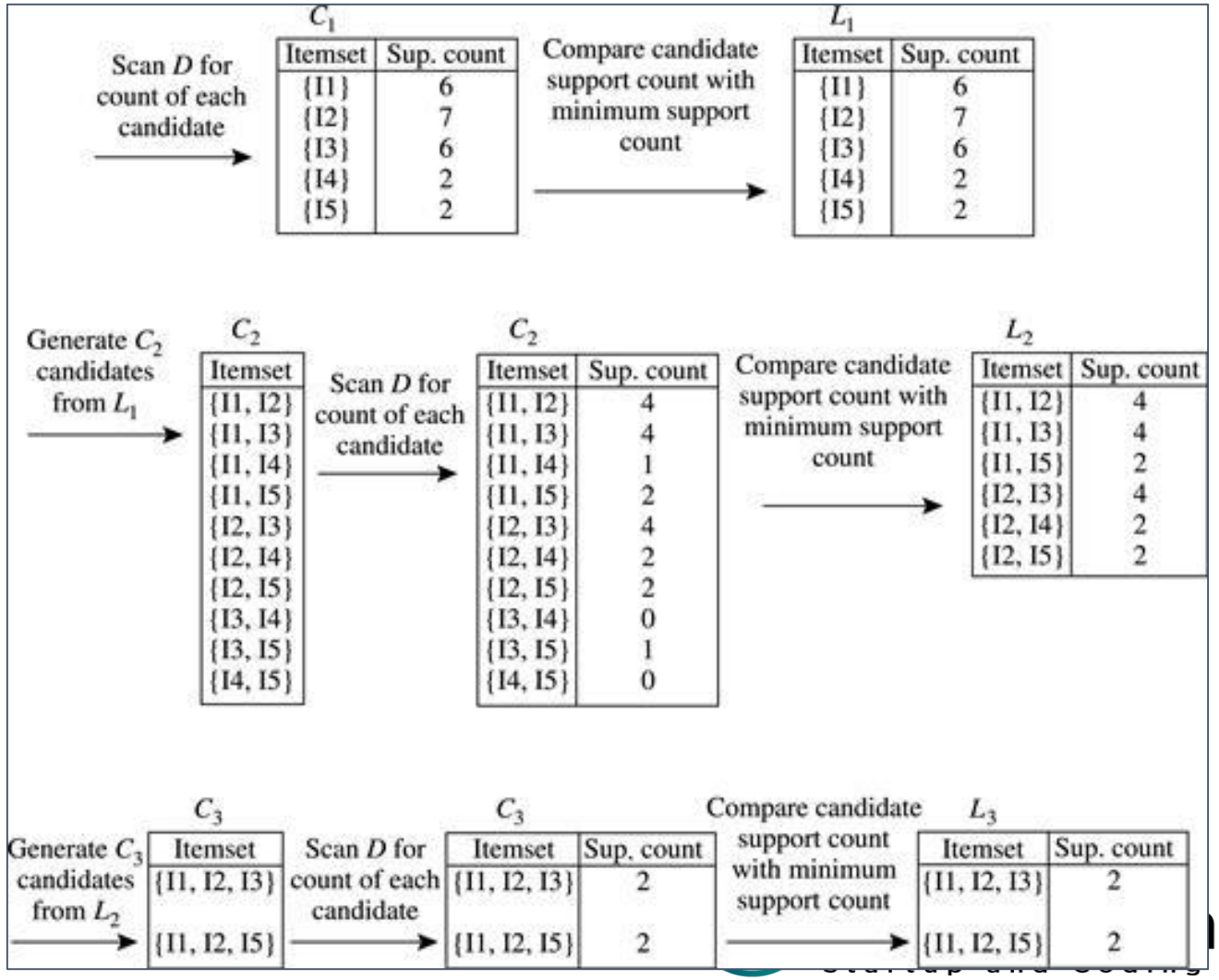
Lift (A -> B): **P(A U B) / {P(A) P(B)}**
Complementary or cannibalism? (ratio of the observed frequency of co-occurrence to the expected frequency)

Scale is the key issue

- **Apriori principle:** the subsets of a frequent itemset must be frequent
- When items are consistently ordered across sets, **to avoid redundancy**, generate k-itemset only from two (k-1)-itemsets that share the same (k-2) items.

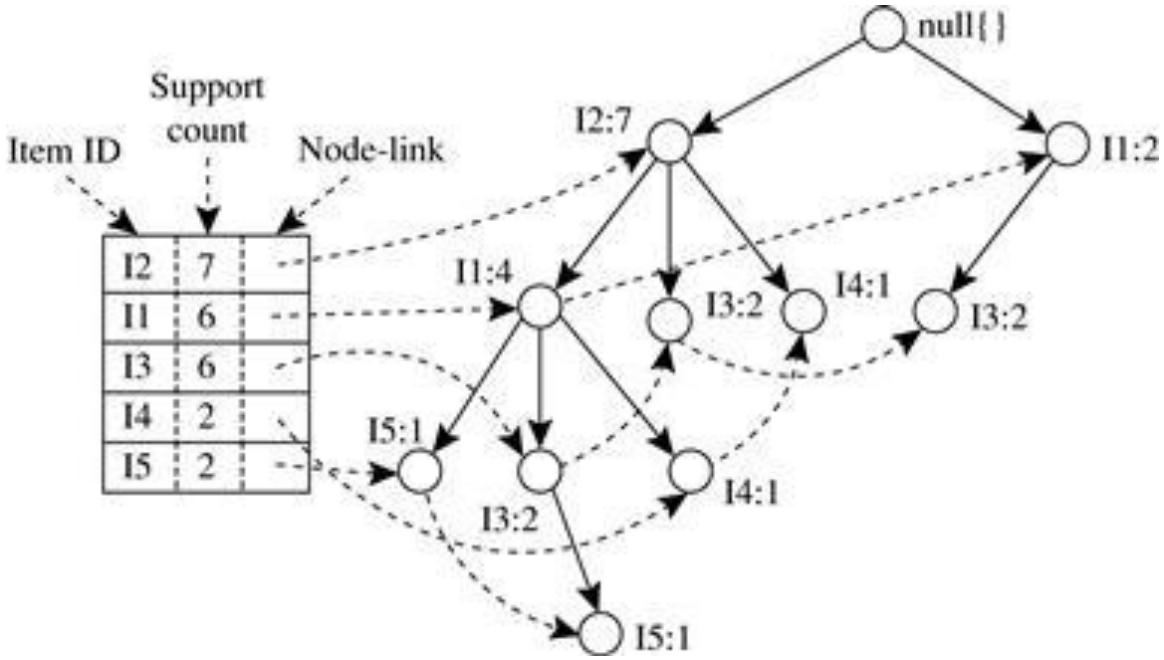
Finding frequent itemsets by Apriori

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



FP-Growth – a faster algorithm

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

Purwadhika
Startup and Coding School