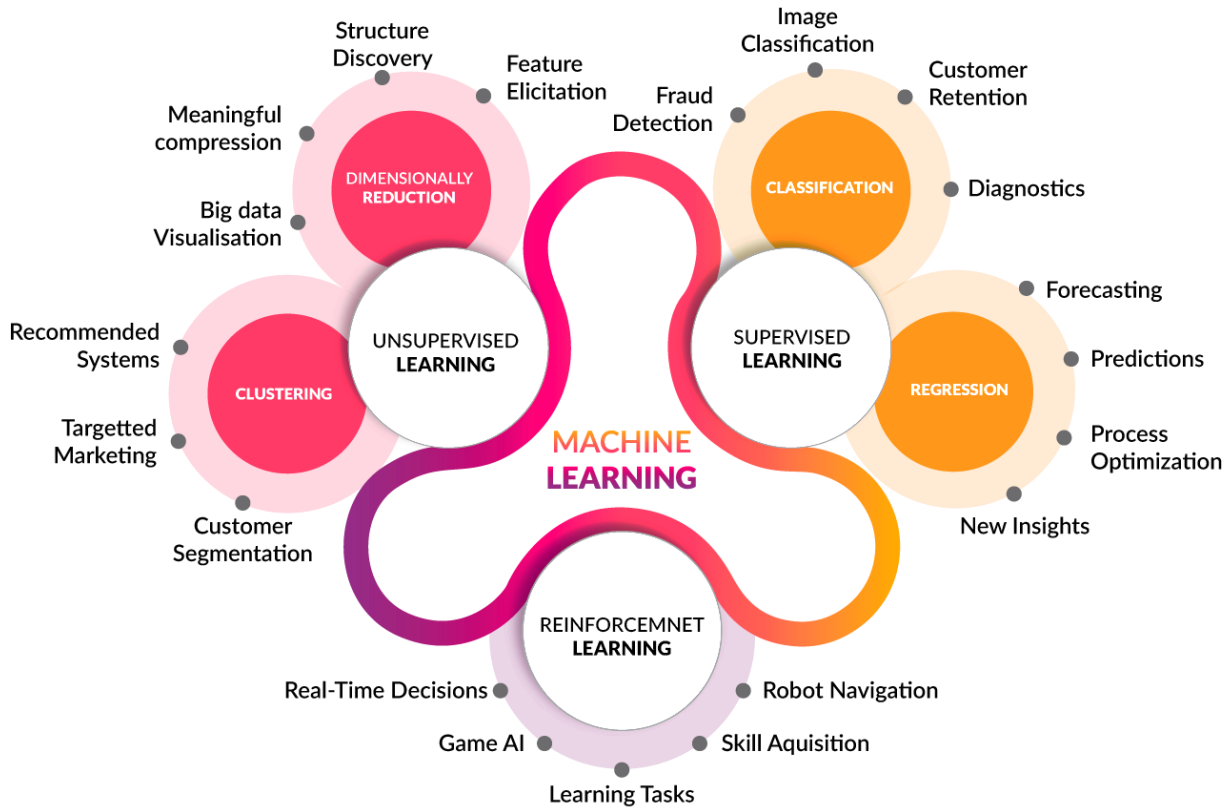


Unsupervised Learning

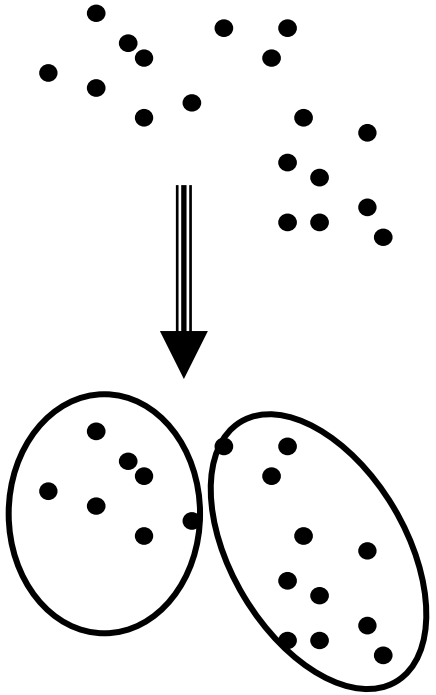
Unit 3

Introduction

- So far in our learning, a training example consisted of a set of input features and a label attached to each input record.
- Unsupervised Learning takes as training examples the set of attributes/features alone without label.
- The purpose of unsupervised learning is to attempt to find natural partitions in the training set through learning from the features.
- Two general strategies for Unsupervised learning include:
    1. Clustering
    2. Dimensionality Reduction
- As unsupervised technique doesn't use any labeled data, it learns intrinsic property from the training data.

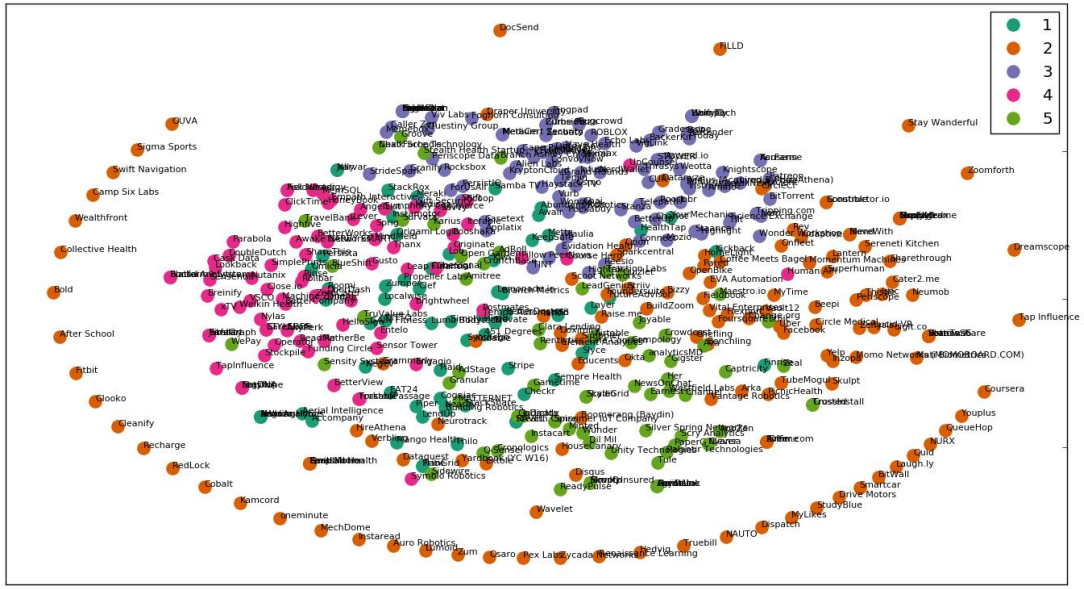
Clustering

- The goal of clustering is to group observations that are similar to each other in an unsupervised manner.
- A cluster is a group of similar observation i.e. the members within a cluster shares some similar characteristics.
- A cluster is represented by a single point known as centroid.
- **Example 1**: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.



Clustering

- **Example 2**: In marketing, segment customers according to their similarities
  - To do targeted marketing.
- **Example 3**: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- **Example 4**: Given a collection of ecommerce transaction, cluster them with fraud and non-fraud credit card transaction
  - To identify future fraud transaction



Clustering

- As said in the previous slide, clustering relies on the similarity/dissimilarity of data points.
- Similarity/dissimilarity is measured through various technique such as:
  - Distance
  - Density
  - Cosine Similarity etc.
- These techniques are used by various clustering algorithm depending on their nature.
- For example, K-Means algorithm uses distance measures to find similarity whereas DBSCAN algorithm uses density. Similarly, in NLP distance between two words or documents are measured using cosine similarity.

K-Means Clustering

- K-Means Clustering is the most common and simple clustering algorithm.
- It is partitioning based algorithm which uses distance as measure to find similarity.
- K in the name refers to the number of cluster expected in the given dataset where the value of K should be pre-known.
- Means in the name refers to the averaging we use to determine the centroid of cluster.
- The K clusters in K-means clustering are searched iteratively. Initially K clusters are taken in random and the cluster is identified after several iterations.
- It is highly likely that different initial cluster center lead to different results. Thus it is better to select cluster centers far away from one another.

K-Means Algorithm

1. Let X be the dataset containing m observations and n features.
2. Randomly select k cluster centers.
3. Calculate the distance between each data point to each cluster centers.
4. Compare the distance of point to each cluster centers and assign the point to cluster with minimum distance.
5. If there is no change in set membership of cluster then terminate.
6. Else, recalculate the new cluster center using mean of all the data points called centroid.
7. Repeat the process from 3 to 7.

K-Means Clustering -Numerical

- Apply K-Means Clustering algorithm to find cluster among given data points where K = 2 using K-Means algorithm.

(2, 3), (8, 2), (9, 3) (3, 1), (2, 5), (10, 3)

K-Means Clustering

- As said before, use of randomization to find initial cluster center may result differently.
- The final clusters are highly dependent on how initial clusters are declared.
- Thus to overcome this, we may use K-means++ algorithm which is improvised version of K-Means.
- Also there are other algorithm such as K-Medoids which can be used instead.
- K-means++ algorithm suggest an approach for initialization of cluster center which follows as below:

K-Means++ Initialization Algorithm

- Randomly select the first cluster center from the data points.
- For each data point in dataset, compute distance with the previously cluster center.
- Select the next cluster center from the data points which is farthest from the previously chosen cluster.
- Repeat this process until K cluster centers have been sampled from the dataset.

You are advised to research on K-Mediods and K-Mode algorithm on your own (not in the scope of syllabus.)

Hierarchical Clustering

- In hierarchical clustering, clusters are identified within a cluster itself i.e. we tend to build algorithm that finds hierarchy of clusters.
- There are two common techniques for hierarchical clustering:
  1. Agglomerative Clustering

     Uses bottom up approach, which considers each observation in dataset as single cluster initially, and gradually clusters are merged as we move up on the hierarchy tree until single cluster – covering whole dataset is found.
  2. Divisive Clustering

     This technique uses top bottom approach in which the whole dataset is first considered into single clusters and then split up into two or multiple clusters such that objects in one subgroup are dissimilar then the objects in another. This process gradually lead us until we find single member in each cluster.

Algorithms

Agglomerative Clustering

1. Consider each data point a cluster.
2. Compute the distance between each data points.
3. Repeat until K cluster remains
    I. Merge the two datapoints forming single clusters.
    II. Update the distance metrics. There are various technique to update the distance metrics.

Divisive Clustering

1. Consider all data points belong to single cluster.
2. Repeat until there is one data point in each cluster.
    I. Split the dataset into cluster using algorithms such as K-Means, K-Mediods etc.
    II. Choose the best cluster among all possibilities.

Agglomerative Clustering - Numerical Example

Apply agglomerative clustering to the following dataset.

$$A\ (1,1), B(2,3), C(3,5), D(4,5), E(6,6), and\ F(7,5)$$

Solution:

First compute the distance from each point to another point.

	A	**B**	**C**	**D**	**E**	**F**
A						
B						
C						
D						
E						
F						

Agglomerative Clustering - Numerical Example

Apply agglomerative clustering to the following dataset.

$$A\ (1,1), B(2,3), C(3,5), D(4,5), E(6,6), and\ F(7,5)$$

Solution:

First compute the distance from each point to another point.

	A	**B**	**C**	**D**	**E**	**F**
A	0	2.236068	4.472136	5	7.071068	7.211103
B	2.236068	0	2.236068	2.828427	5	5.385165
C	4.472136	2.236068	0	1	3.162278	4
D	5	2.828427	1	0	2.236068	3
E	7.071068	5	3.162278	2.236068	0	1.414214
F	7.211103	5.385165	4	3	1.414214	0

Agglomerative Clustering - Numerical Example

Apply agglomerative clustering to the following dataset.

$$A\ (1,1), B(2,3), C(3,5), D(4,5), E(6,6), and\ F(7,5)$$

Solution:

First compute the distance from each point to another point.

	A	**B**	**C**	**D**	**E**	**F**
A	0	2.236068	4.472136	5	7.071068	7.211103
B	2.236068	0	2.236068	2.828427	5	5.385165
C	4.472136	2.236068	0	1	3.162278	4
D	5	2.828427	1	0	2.236068	3
E	7.071068	5	3.162278	2.236068	0	1.414214
F	7.211103	5.385165	4	3	1.414214	0

The upper triangular section is the mirror reflection of lower triangular matrix. And you can omit the upper triangular portion.

Agglomerative Clustering - Numerical Example

Apply agglomerative clustering to the following dataset.

$$A\ (1,1), B(2,3), C(3,5), D(4,5), E(6,6), and\ F(7,5)$$

Solution:

First compute the distance from each point to another point.

	A	B	C	D	E	F
A	0					
B	2.236068	0				
C	4.472136	2.236068	0			
D	5	2.828427	1	0		
E	7.071068	5	3.162278	2.236068	0	
F	7.211103	5.385165	4	3	1.414214	0

The upper triangular section is the mirror reflection of lower triangular matrix. And you can omit the upper triangular portion.

Agglomerative Clustering - Numerical Example

- Now, considering the every point as single cluster, we merge two cluster having minimal distance.

	A	B	C	D	E	F
A	0					
B	2.236068	0				
C	4.472136	2.236068	0			
D	5	2.828427	1	0		
E	7.071068	5	3.162278	2.236068	0	
F	7.211103	5.385165	4	3	1.414214	0

C and D are having smallest distance between them. So we merge two of them.

Agglomerative Clustering - Numerical Example

After merging two cluster, we get

	A	B	{C, D}	E	F
A	0				
B	2.236068	0			
{C, D}	4.472136	2.236068	0		
E	7.071068	5	2.236068	0	
F	7.211103	5.385165	3	1.414214	0

Now, E and F are having smallest distance between them. So we merge two of them.

Agglomerative Clustering - Numerical Example

After merging two cluster, we get

	A	**B**	**{C, D}**	**{E, F}**
A	0			
B	2.236068	0		
{C, D}	4.472136	2.236068	0	
{E, F}	7.071068	5	2.236068	0

Now, E and F are having smallest distance between them. So we merge two of them.

Agglomerative Clustering - Numerical Example

After merging two cluster, we get

	A	{B, C, D}	{E, F}
A	0		
{B, C, D}	2.236068	0	
{E, F}	7.071068	2.236068	0

Now, A and {B, C, D} and {B, C, D} and {E, F} are having equal but smallest distance between them. But we take one only. Considering A and {B, C, D}, So we merge two of them.

Agglomerative Clustering - Numerical Example

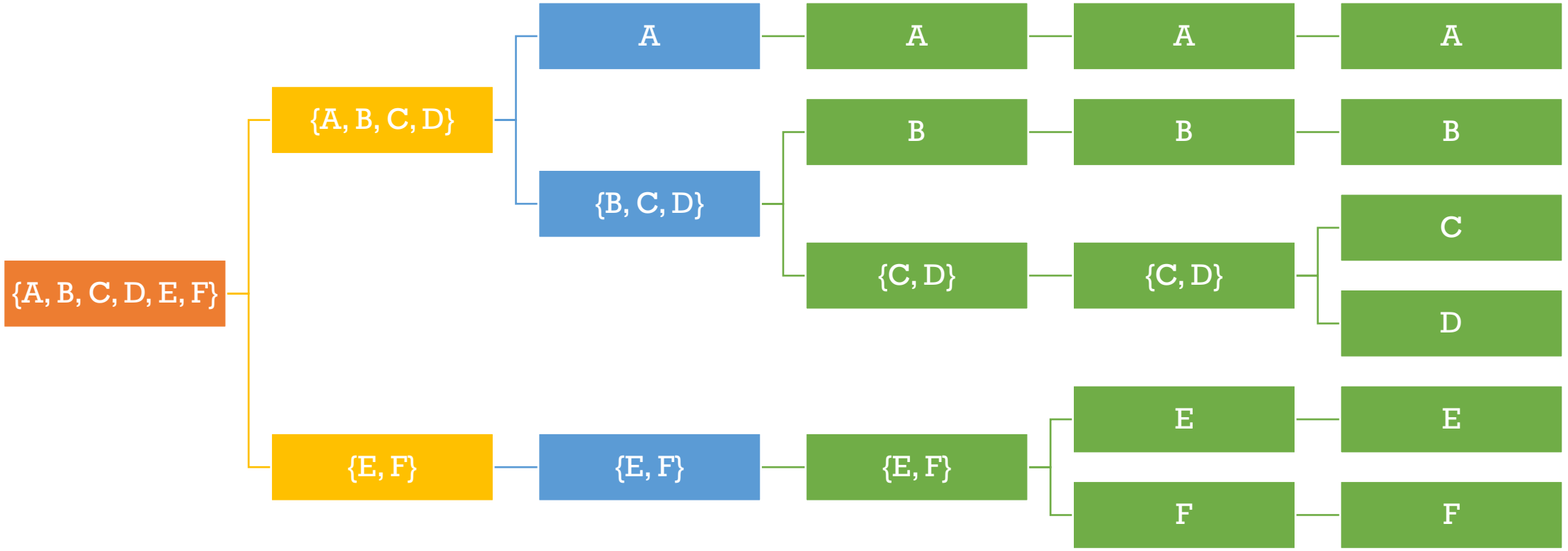
After merging two cluster, we get

	{A, B, C, D}	{E, F}
{A, B, C, D}	0	
{E, F}	2.236068	0

Finally, we merge {A, B, C, D} and {E, F} to get single cluster {A, B, C, D, E, F}.

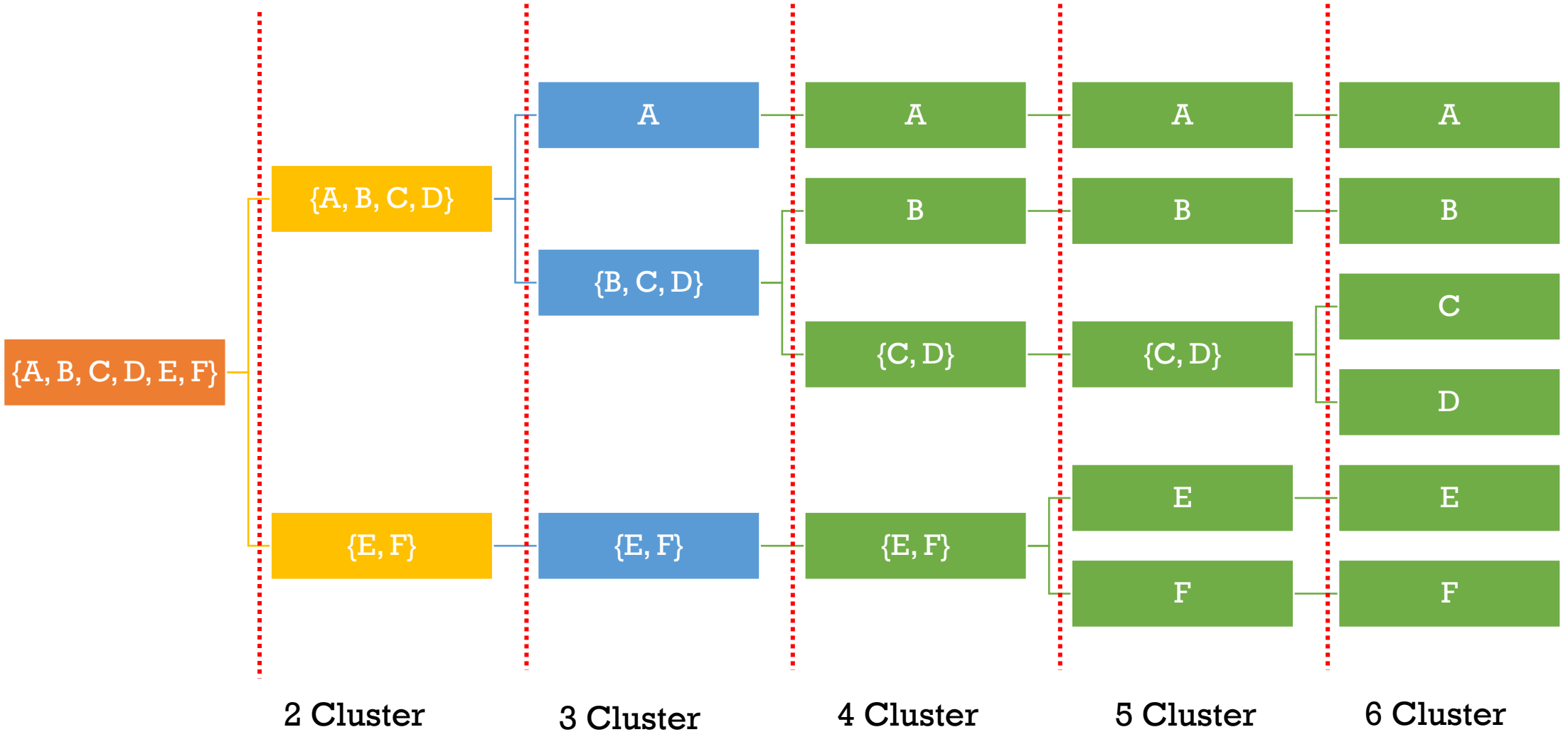
Agglomerative Clustering - Numerical Example

Constructing a dendrogram from the above we get,



Agglomerative Clustering - Numerical Example

Applying cut at different level, we get different number of cluster.



Density Based Clustering

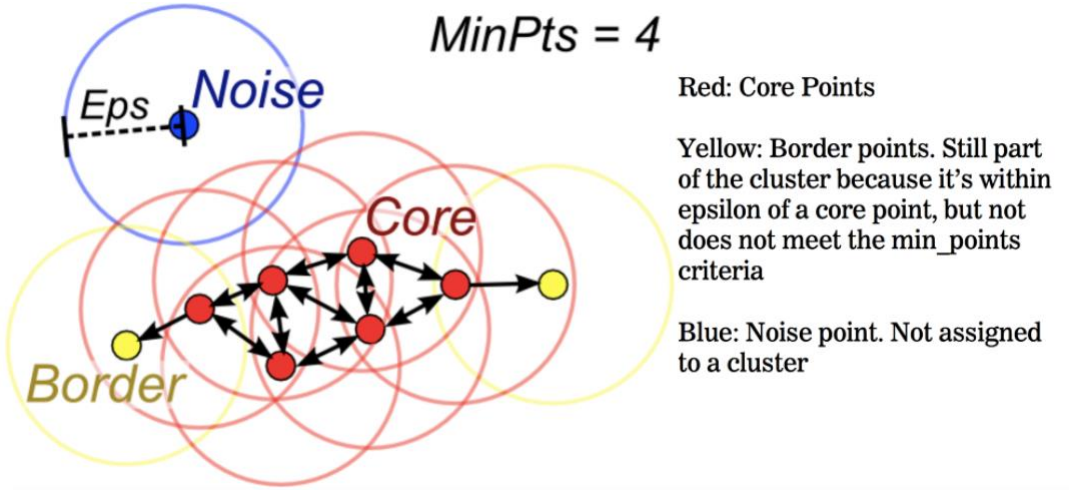
- Partition based clustering are suitable for compact and well separated clusters.
- Density based clustering is based on detecting region where observations are concentrated and where they are sparse.
- This method automatically detect patterns based on sparsity/density at particular location and the distance to some prespecified number of neighbors.
- Observation that are not part of cluster are called Noise.
- There are multiple techniques under density based clustering. Few of them are:
  - DBSCAN (In Scope of syllabus)
  - HDBSCAN (Out of Scope)
  - OPTICS (Out of Scope)

DBSCAN

- DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise.
- It groups ‘densely grouped’ observation into a single cluster and can identify clusters in large spatial datasets by looking at the local density of the data points.
- Like as Hierarchical clustering, it also doesn’t require no. of cluster to be pre-known.
- And, most importantly it is robust to outlier.
- DBSCAN uses two parameters:
  - Epsilon
  - minPoints

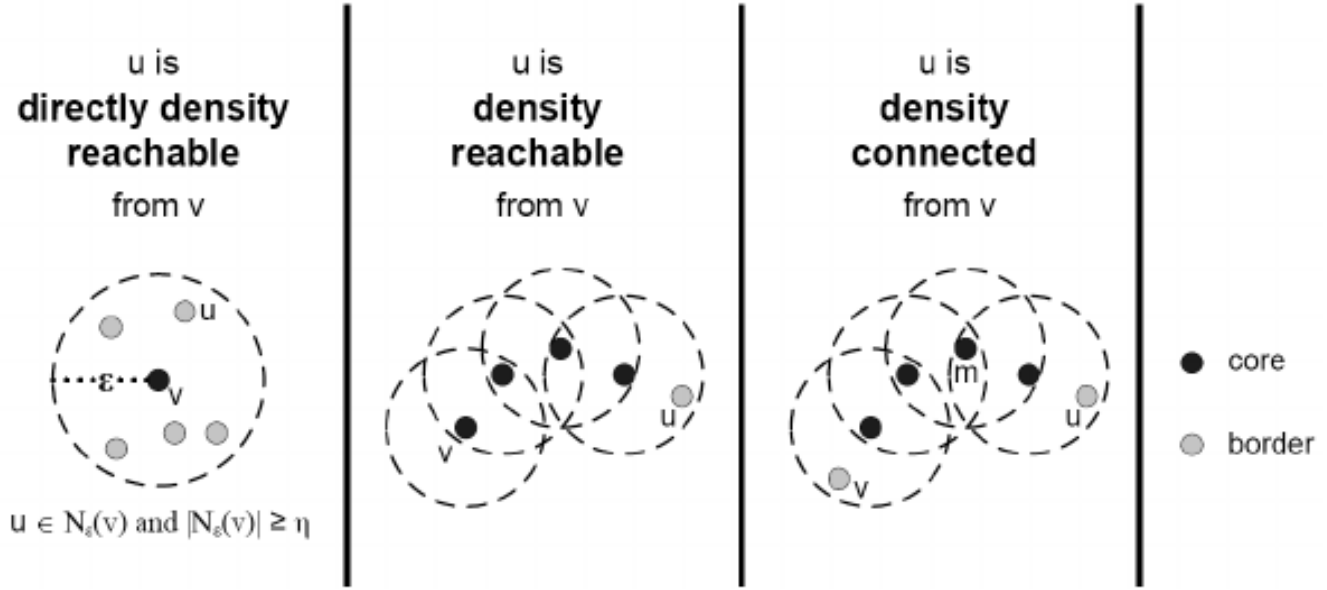
DBSCAN

- **Epsilon** (ϵ) is the radius of circular region around which the density is to be checked up.
- **minPoints** (n) is the minimum number of data points required inside that circle for that data point to be classified as a **Core point**.
- A data point is a **Core point** (x) if the circle around it contains at least n number of points.
- **Border point** (y) means Data point that has at least one point in core within epsilon (ε) distance but is lower than minPoints (n) within epsilon (ε) distance from it.
- **Noise Point** (z): Data point that has no core points within epsilon (ε) distance.



DBSCAN

- Density reachable and Density Connected
  - **Density reachable:** A point "A" is density reachable from "B" if there are a set of core points leading from "B" to "A.
  - **Density connected:** Two points "A" and "B" are density connected if there are a core point "C", such that both "A" and "B" are density reachable from "C".



DBSCAN

- Now, a density-based cluster is defined as a group of density connected points.
- We consider minPoints as a threshold for considering a cluster as a cluster. And, a cluster is only recognized if the number of observations is greater than or equal to the minPoints.
- A point x is directly density reachable from point p if a point p is a core point and x is in p's ε-neighborhood.
- All points within a ε-neighborhood of a core point belongs to same clusters and all points in a cluster are directly density reachable to core point.

DBSCAN - Algorithm

1. Choose any point p randomly
2. Identify all density reachable points from p with ε and n parameter
3. If p is a core point, create a cluster if ε and n satisfy.
4. If p is a border point, visit the next point in a dataset
5. Continue the algorithm until all points are visited.

DBSCAN – Numerical Example

Apply DBSCAN clustering to the following dataset with epsilon 3 and minimum data points 3.

$$A\,(1,1), B(2,3), C(3,5), D(4,5), E(6,6), and\ F(7,5)$$

Solution:

First compute the distance of each data point to each other.

	A	B	C	D	E	F
A	0	2.236068	4.472136	5	7.071068	7.211103
B	2.236068	0	2.236068	2.828427	5	1.414214
C	4.472136	2.236068	0	1	3.162278	4
D	5	2.828427	1	0	2.236068	3
E	7.071068	5	3.162278	2.236068	0	1.414214
F	7.211103	5.385165	4	3	1.414214	0

DBSCAN – Numerical Example

Now compare the distance to epsilon and get for which distance is greater than epsilon

For A: B
For B: A, C, D
For C: B, D
For D: B, C, E, F
For E: D, F
For F: D, E

	A	B	C	D	E	F
A	0	2.236068	4.472136	5	7.071068	7.211103
B	2.236068	0	2.236068	2.828427	5	5.385165
C	4.472136	2.236068	0	1	3.162278	4
D	5	2.828427	1	0	2.236068	3
E	7.071068	5	3.162278	2.236068	0	1.414214
F	7.211103	5.385165	4	3	1.414214	0

DBSCAN – Numerical Example

Next, we identify core points and noise

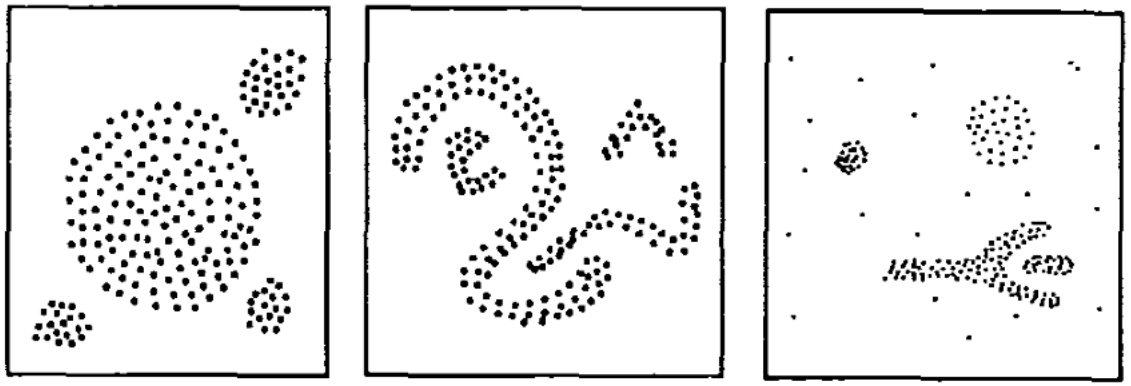
Points	Status
A	Border
B	Core Points
C	Border Points
D	Core Points
E	Border Points
F	Border

For A: B
For B: A, C, D
For C: B, D
For D: B, C, E,
For E: D, F
For F: E

And we now check for border points if any noise points are.

DBSCAN

- One of the biggest benefit of using Density based clustering is they can identify any shape of cluster. And outliers as well.



Gaussian Mixture Model

- Gaussian mixture model is soft clustering algorithm i.e. the algorithm may assign data point to multiple cluster.
- i.e. there are likely to be overlapping clusters.
- Gaussian Mixture Models assume the number of cluster to be pre-known.
- Hence, a Gaussian Mixture Model forms a cluster belonging to single distribution.
- Thus it is a statistical model that involves latent variables and hence cannot be solved using MLE method.

Gaussian Mixture Model

- A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.
- Thus, this model attempts to model the dataset as a mixture of several Gaussian Distribution
- **Gaussian Distribution**
  - Gaussian Distribution is also known as Normal Distribution.
  - It is a symmetric distribution where most of observations cluster around the centre peak and the probabilities further away from the Mean tapped off easily in both directions Extreme values of the distribution are most unlikely.
  - It has two parameters:
    1. Mean, and
    2. Standard Deviation

Gaussian Mixture Model

- Gaussian Distribution (contd.)
  - In one dimension, the pdf of Gaussian Distribution is given as:

$$G(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\,\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  - And, in multi dimension, the pdf is given as:

$$G(X|\mu,\Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}}\exp(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu))$$

Where,

μ is d dimensional vector denoting the mean of the distribution

And Σ is the $d \times d$ covariance matrix.

Gaussian Mixture Model

- If we have only one dimension, we can easily estimate the parameters μ and σ using ***maximum-likelihood*** method.
- But if there are say K clusters i.e. there are K different distributions first we define the combined pdf of K clusters as linear function of densities of all these K distributions.

$$i.e.\, p(X) = \sum_{k=1}^{K} \pi_k G(X|\mu_k, \Sigma_k)$$

Where π_k is the mixing coefficient for k-th distribution.

- Now, we estimate the parameters by the maximum log-likelihood method.

Gaussian Mixture Model

- And, finally we get,

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_k(x_n) x_n}{\sum_{n=1}^{N} \gamma_k(x_n)}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma_k(x_n)(x_n - \mu_k)(x_n - \mu_k)\text{^}T}{\sum_{n=1}^{N} \gamma_k(x_n)}$$

And,

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_k(x_n)$$

Here, $\gamma_k(x)$ is a random variable.

Gaussian Mixture Model

- To conclude, we can see that the parameters can be estimated easily through maximum likelihood.
- For this Expectation Maximization algorithm is beneficial.
- The Expectation-Maximization (EM) algorithm is an iterative way to find maximum-likelihood estimates for model parameters when the data is incomplete or has some missing data points or has some hidden variables.
- EM chooses some random values for the missing data points and estimates a new set of data (on missing part).
- These new values are then recursively used to estimate a better first data, by filling up missing points, until the values get fixed.

Gaussian Mixture Model

- There are two steps in the Expectation-Maximization algorithm.

1. Estimation Step (E-Step)
   I. First, initialize our model parameters i.e. μ_k, Σ_k, and γ_k mean, covariance matrix, and mixing coefficients respectively.
   II. Now, calculate posterior probabilities, using bayes theorem, of data points belonging to each cluster using current values. These probabilities are represented by γ_k.

$$\gamma_k(X) = \frac{p(X|k)p(k)}{\sum_{k=1}^{K} p(k)p(X|k)}$$

   I. Finally, estimate the value of the lantent variables γ_k based on the current parameter values.

2. Maximization Step (M-Step)

Gaussian Mixture Model

2. Maximization Step (M-Step)

- Update the parameter values using the estimated latent variables (γ_k)
  - Update the mean of the cluster point (μ_k) by taking the weighted average of data points using the corresponding latent variable probabilities.

$$\mu_k = \frac{1}{K}(\gamma_1 x_1 + \cdots + \gamma_K x_K)$$

  - Similarly, update the covariance matrix by taking the weighted average of the squared differences between the data points and the mean, using the latent variable probabilities.

$$\Sigma_k = \frac{1}{K}(\gamma_1 (x_1 - \mu_1)^2 + \cdots + \gamma_K (x_K - \mu_K)\text{^}2\ )$$

  - And, finally update the mixing coefficients by taking the average of the latent variable probabilities for each component.

Gaussian Mixture Model

- Repeat E-step and M-step until convergence.

Derivation: https://core.ac.uk/download/pdf/82750424.pdf

Dimensionality Reduction

- Higher the dimension of data, more the model complexity is and difficult to train the model.
- Also, increasing the features will not always improve classification accuracy.
- Thus, in machine learning world, **Curse of Dimensionality** is common phrase which basically refers to the point where having more dimension on data doesn't help in the conclusion, rather it would be the pain.
- Thus, the purpose of dimensionality reduction is to reduced the number of dimension or features from dataset.
- But while doing so, the pattern from the data we are trying to learn shouldn't be distorted.

Dimensionality Reduction

- However, reducing the features always comes with the cost and may impact the accuracy.
- Thus it may be challenging to reduce the feature without any significance loss in accuracy.
- Thus, dimensionality reduction deals with the process of reducing the number of variables under consideration by obtaining smaller set of variables.
- There are many methods common for dimensionality reduction like:
  - Principal Component Analysis
  - Feature Selection
  - Low Variance Filter
  - Linear Discriminant Analysis etc.

PCA Algorithm

1. Consider a dataset as below:

x_1	x_2	...	x_n
x_{11}	x_{12}	...	x_{1n}
...	...	...	...
x_{m1}	x_{m2}	...	x_{mn}

2. Standardize each columns in the dataset
3. Compute the covariance matrix as:

$$\begin{bmatrix} cov(x_1, x_1) & \dots & cov(x_1, x_n) \\ \dots & \dots & \dots \\ cov(x_n, x_1) & \dots & cov(x_n, x_n) \end{bmatrix}$$

PCA Algorithm

Where,

$$cov(x_j, x_k) = \frac{\sum_{i=1}^{m}(x_{ij} - \bar{x_j})(x_{ik} - \bar{x}_k)}{m}$$

Note that the covariance matrix is symmetric i.e. entries are reflection across diagonal of matrix.

Why Covariance?

Covariance matrix is a tabular like structure that summarizes the correlations between all the possible pairs of features.

PCA Algorithm

4. Compute the Eigen values and eigen vectors from the covariance matrix.

Compute Eigen Value and Vectors from

$$(A - \lambda I)\boldsymbol{x} = 0$$

Where,

$$\det(A - \lambda I) = 0$$

Gives us the eigen values and

$$(A - \lambda I)x = 0$$

Give us the eigen vectors. By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

Thus, the eigenvectors of the Covariance matrix belonging to largest eigen values are actually the directions of the axes where there is the most information (i.e. explains the most variance from data). And this is the first principal component (PC_1). And so on…

PCA Algorithm

After having the principal components, in order to compute the percentage of information accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues.

For example,

Say, our result from the computation is $v1 = \begin{bmatrix} 0.782 \\ -0.435 \end{bmatrix}$ and $v2 = \begin{bmatrix} 0.646 \\ 0.553 \end{bmatrix}$ for $\lambda_1 = 0.1$ and $\lambda_2 = 1.2$.

Since $\lambda_2 > \lambda_1$, v_2 is the first principal component and v_1 is the second principal component.

Now, to compute the percentage of variance explained by v_2 is $\frac{1.2}{0.646 + 0.553} = 0.9675$ and the variance explained by v_1 is 0.0325.

PCA Algorithm

Now we can discard the eigenvector v_1, which is the one of lesser significance, and form a feature vector with v_2 only.

Discarding the eigenvector v_1 reduces the dimensionality of our dataset by 1which consequently cause a loss of information in the final data set.

But remember, v_1 was carrying almost 3.5% of variance which has been compromised. Also, v_2 alsone is carrying 96.5% of variance which is valuable for model to train.

5. Finally, reorient the data from their original axes to the ones you have calculated from the Principal components.

$$Final\ Data\ Set = Standardized\ Original\ Data\ Set^T * FeatureVector$$

Properties of Principal Component

- The new features generated using PCA are distinct i.e. the covariance between the new features is 0.
- The principal components are generated in order of the variance explained by them. Hence, the first principal component captures the maximum variance, the second one captures the next highest variability and so on.
- The sum of the variance explained by the new features / the principal components should be equal to the sum of the variance of the original features.

Variation of PCA

- There are several variation of PCA:
  - **Kernel PCA**: This variation of PCA uses a kernel trick to transform the data into a higher-dimensional space where it is more easily linearly separable. This can be useful for handling non-linearly separable data.
  - **Incremental PCA**: This variation of PCA allows for the analysis of large datasets that cannot be fit into memory all at once. It is useful for handling big data problems.
  - **Sparse PCA**: This variation of PCA adds a sparsity constraint to the PCA problem, which encourages the algorithm to find a lower-dimensional representation of the data with fewer non-zero components.
  - **Robust PCA**: This variation of PCA is designed to handle datasets with outliers or noise. It separates the data into a low-rank component and a sparse component, where the sparse component represents the outliers or noise.

Variation of PCA

- **Randomized PCA:** Randomized PCA is a variation of Principal Component Analysis (PCA) that is designed to approximate the first k principal components of a large dataset efficiently. Instead of computing the eigenvectors of the covariance matrix of the data, as is done in traditional PCA, randomized PCA uses a random projection matrix to map the data to a lower-dimensional subspace. The first k principal components of the data can then be approximated by computing the eigenvectors of the covariance matrix of the projected data

```
>>> import numpy as np
>>> from sklearn.decomposition import RandomizedPCA
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> pca = RandomizedPCA(n_components=2)
>>> pca.fit(X)
RandomizedPCA(copy=True, iterated_power=3, n_components=2,
       random_state=None, whiten=False)
>>> print(pca.explained_variance_ratio_)
[ 0.99244...  0.00755...]
```

Low Rank Approximations

- Low-rank approximations are a commonly used technique in data analysis and machine learning for reducing the dimensionality of high-dimensional data.
- Given a matrix A, a low-rank approximation involves finding a matrix B that is of lower rank than A but approximates A as closely as possible.
- Rank of matrix:
  - Rank of a (m x n) matrix is determined by the number of linearly independent rows present in the matrix.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The rank of matrix A is 3 since all columns are independent of each other.

Low Rank Approximations

- Rank of matrix:

$$B = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The rank of matrix B is 2 as column 2 dependent in column 1.

Mathematically, LRA is a minimization problem, in which we measure the fit between a given matrix (the data) and an approximating matrix (the optimization variable).

Intuitively, we tend to see how linearly similar matrix B is to the input matrix A.

Singular Value Decomposition

- Let $A \in R^{m \times n}$ with $m >= n$. Then there are orthogonal matrices $U \in R^{m \times m}$ and $V \in R^{n \times n}$ such that

$$A = U\Sigma V^T$$

where,

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ 0 & \dots & 0 \\ 0 & \dots & \sigma_n \end{bmatrix}; \Sigma \in R^{m \times n}$$

In Sigma, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Here, $\sigma_1, \dots, \sigma_n$ are called singular values, $u_1, \dots, u_n$ are called left singular vectors and $v_1, \dots, v_n$ are called right singular vectors.

Singular Value Decomposition

And,

$$Av_i = \sigma_i u_i \text{ for i} = 1, \ldots, \text{n}.$$

A very significant feature of SVD is that it allows us to truncate few contexts which are not necessarily required by us. The $\sum$ matrix provides us with the diagonal values which represent the significance of the context from highest to the lowest. By using these values we can reduce the dimensions and hence this can be used as a dimensionality reduction technique too.

Latent Semantic Analysis

Latent Semantic Analysis is the application in natural language processing of Singular Value Decomposition. It is used to analyze the relationships between a set of documents and the terms they contain.

LSA is primarily used for concept searching and automated document categorization. However, it's also found use in software engineering (to understand source code), publishing (text summarization), search engine optimization, and other applications.

Consider the collection of documents, m be the number of documents and n be the number of unique words in the collection. Then, D is the matrix of m x n dimension.

Latent Semantic Analysis

Applying SVD,

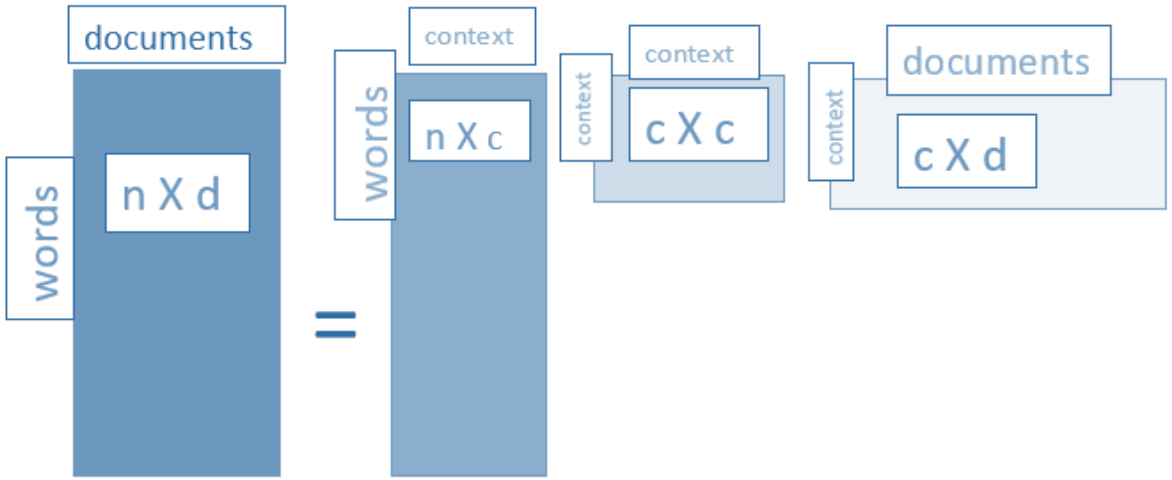
$$D = U\Sigma V^T$$

Where,

U gives the distribution of words across the different contexts,

Σ is the diagonal matrix that contains the association among the contexts

And V refers the distribution of contexts across the different contexts



Latent Semantic Analysis

Thus, If we select the k largest diagonal values in Σ, then

$$D_k = U_k \Sigma_k V_k^T$$

Where,

D_k is the approximated matrix of D.

Canonical Correlation Analysis

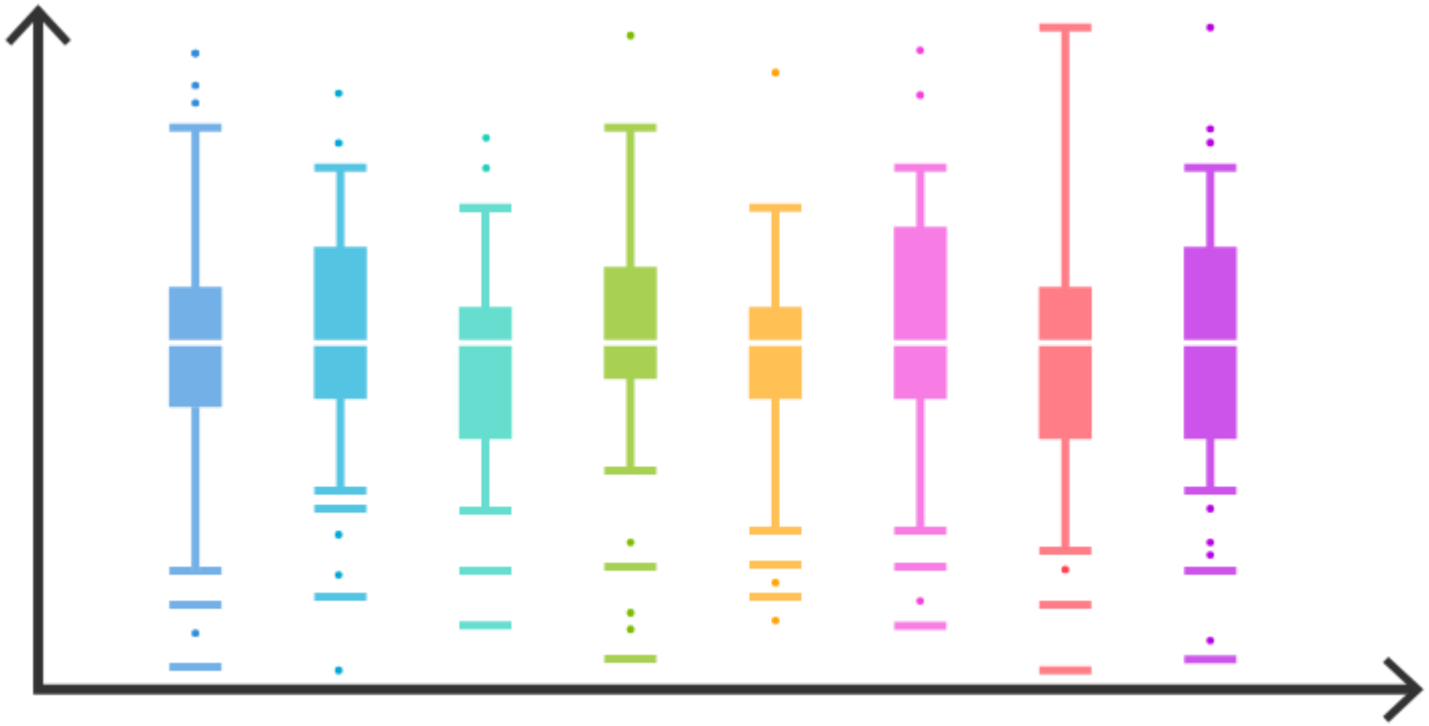
- It is another technique for dimensionality reduction.
- Canonical Correlation Analysis explores the relationship between two variables in datasets.
- Canonical Correlation Analysis allows us to summarize the relationships into fewer statistics while preserving the main facets of the relationships.
- In a way, the motivation for canonical correlation is very similar to principal component analysis.
- Consider two features f_1 and f_2 in a dataset D.
- We now look out for the linear combination of the data similar to PCA.

Canonical Correlation Analysis

- We define a set of linear combinations named U and V.
- U corresponds to the linear combinations from the first set of variables, f_1, and V corresponds to the second set of variables, f_2.
- Each member of U is paired with a member of V. For example, below is a linear combination of the f_1 variables and is the corresponding linear combination of the f_2 variables.

Outlier Detection

- Outlier detection is the process of detecting outliers, or a data point that is far away from the most of the data points in datasets.
- Depending on what we are trying to accomplish, potentially removing or resolving outlier from the analysis is to avoid any potential skewing.
- Outlier detection is one of the most important processes taken to create good, reliable data.
- Outliers are extreme data points that are beyond the expected norms for their type. This can be a whole data set that is confounding, or extremities of a certain data set.



What Causes an Outlier?

- There are various causes of outlier, some of the common cause are:
  1. Incorrect data entry by humans
  2. Codes used instead of values
  3. Sampling errors, or data has been extracted from the wrong place or mixed with other data
  4. Unexpected distribution of variables
  5. Measurement errors caused by the application or system
  6. Experimental errors in extracting the data or planning errors
  7. Intentional dummy outliers inserted to test the detection methods
  8. Natural deviations in data, not actually an error, that are indicate fraud or some other anomaly you are trying to detect

Outlier Analysis

- Outlier Analysis is a process that involves identifying the anomalous observation in the dataset.
- Outlier Analysis is the thorough study of what and why have they occurred.
- And most importantly outlier are not always thrown or removed from the original dataset. Sometimes, outliers are observations of interest introducing new phenomenon or crucial point of error.

Clustering based method

- Clustering-based approaches detect outliers by examining the relationship between Objects and Cluster.
- Clustering-based outlier detection methods assume that the normal data objects belong to large and dense clusters, whereas ***outliers belong to small or sparse clusters, or do not belong to any clusters***.
- An object is identified as an outlier as below:
  - Does the object belong to any cluster? If not, then it is identified as an outlier.
  - Is there a large distance between the object and the cluster to which it is closest? If yes, it is an outlier.
  - Is the object part of a small or sparse cluster? If yes, then all the objects in that cluster are outliers.

Clustering based method…

- There are several clustering technique used for outlier detection:
  - Detecting outliers as objects that do not belong to any cluster.
  - Clustering-based outlier detection using distance to the closest cluster.
  - Clustering-based outlier detection using lower dense cluster as outliers cluster.

Classification based approaches

- Outlier detection can be treated as a classification problem if a training dataset with class labels is available.
- The general idea of classification based outlier detection methods is to train a classification model that can distinguish normal data from outliers.
- Consider a training data set that contains samples labeled as **normal** and others labeled as **outlier**.
- A classifier can then be constructed based on the training set. Any classification method can be used.

Classification based approaches…

- Consider intrusion detection in a system, for example. Because most system access are normal, it is easy to obtain a good representation of normal events. However, it is infeasible to enumerate all potential intrusions, as new and unexpected attempts occur from time to time. Hence, we are left with an insufficient representation of the outlier (or intrusion) samples.
- To overcome this challenge, classification based outlier detection methods often use a ***one-class model***. That is, a classifier is built to describe only the normal class. Any samples that do not belong to the normal class are regarded as outliers.

Classification based approaches…

- **Outlier detection using a one-class model**
  - Consider the training set shown in figure, where points are samples labeled as normal and black points are samples labeled as outlier. To build a model for outlier detection, we can learn the decision boundary of the normal class using classification methods.
  - Given a new object, if the object is within the decision boundary of the normal class, it is treated as a normal case. If the object is outside the decision boundary, it is declared an outlier.