



MACHINE INTELLIGENCE

Dr. N MEHALA

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

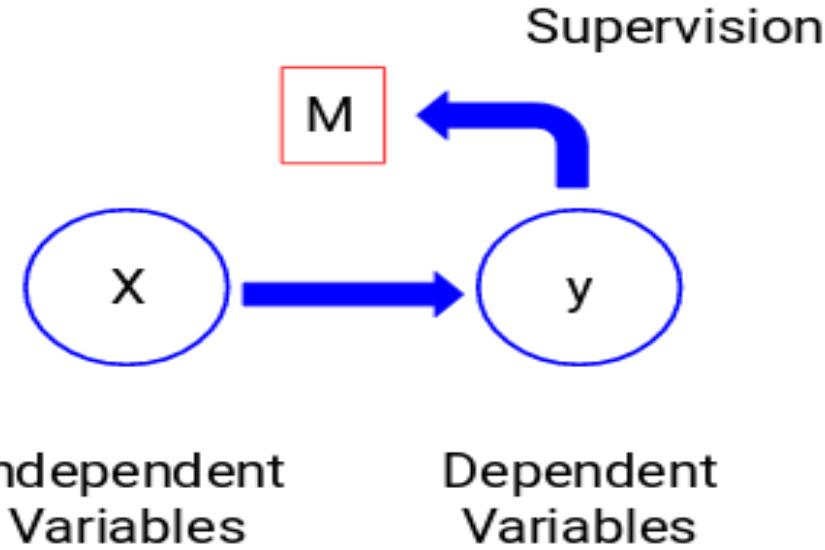
Dr. N MEHALA

Department of Computer Science and Engineering

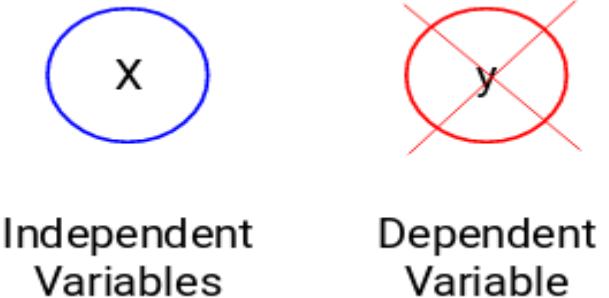
MACHINE INTELLIGENCE

Supervised Vs Unsupervised Learning





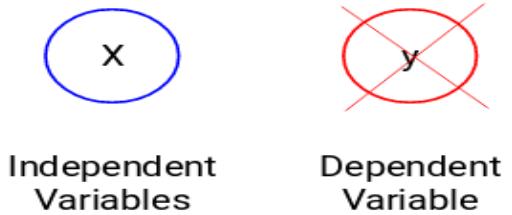
We train our model using the independent variables in the supervision of the target variable and hence the name supervised learning.



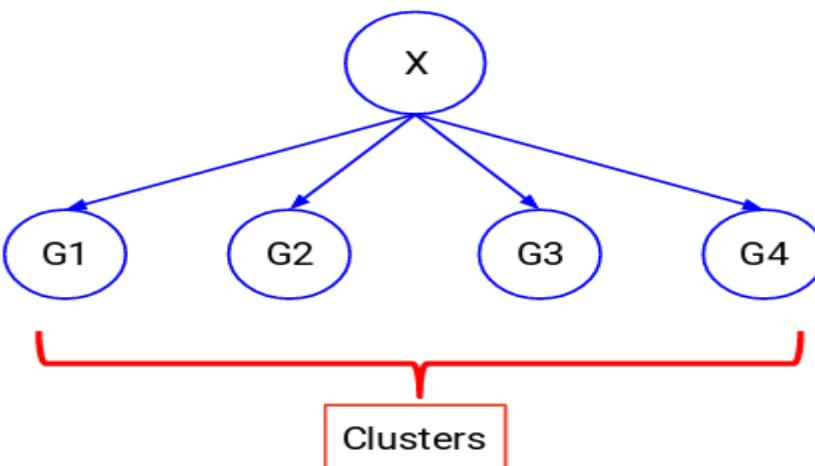
- There might be situations when we do not have any target variable to predict.
- Such problems, without any explicit target variable, are known as unsupervised learning problems.
- We only have the independent variables and no target/dependent variable in these problems.

MACHINE INTELLIGENCE

UnSupervised Learning



- Divide the entire data (X) into a set of groups. These groups are known as clusters and the process of making these clusters is known as **clustering**.



MACHINE INTELLIGENCE

Supervised Vs Unsupervised Learning



Why Supervised Learning?

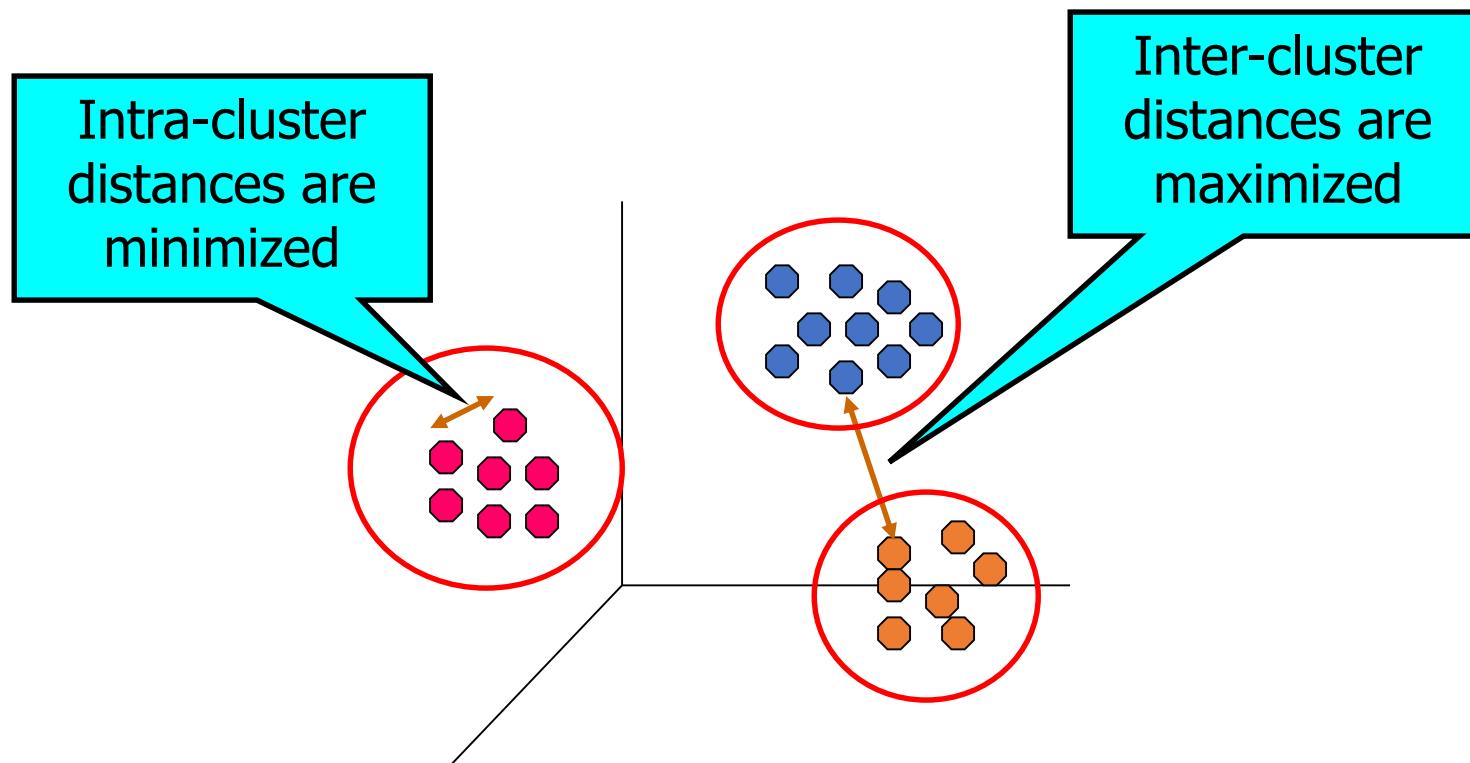
- Supervised learning allows you to collect data or produce a data output from the previous experience.
- Helps you to optimize performance criteria using experience
- Supervised machine learning helps you to solve various types of real-world computation problems.

MACHINE INTELLIGENCE

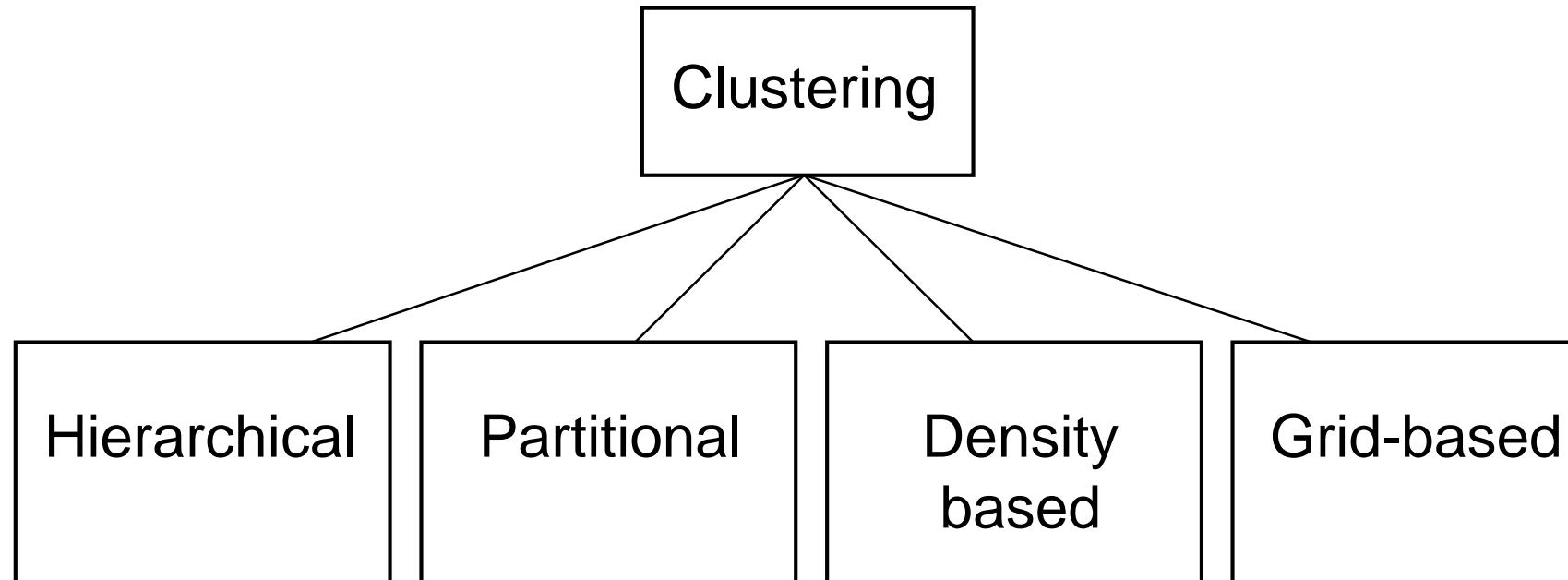
Supervised Vs Unsupervised Learning

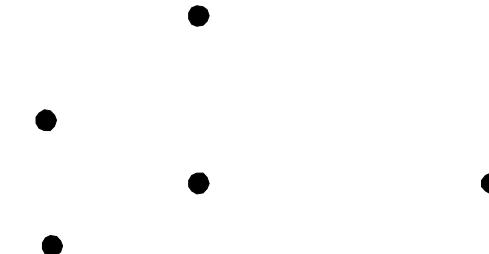
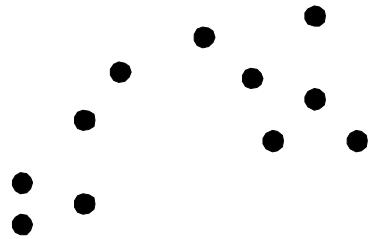


Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups

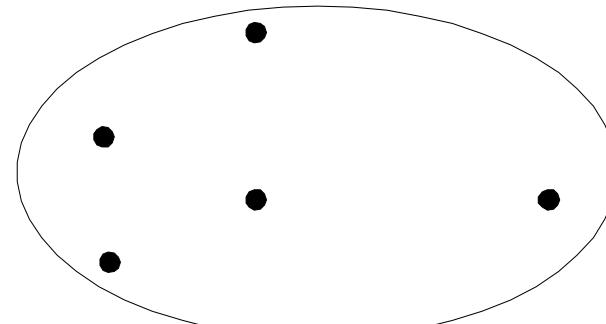
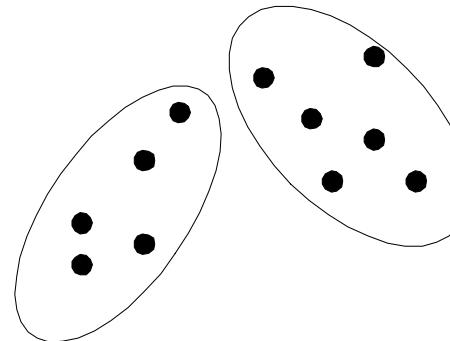


- Marketing
- Insurance
- City-planning
- Earth-quake studies

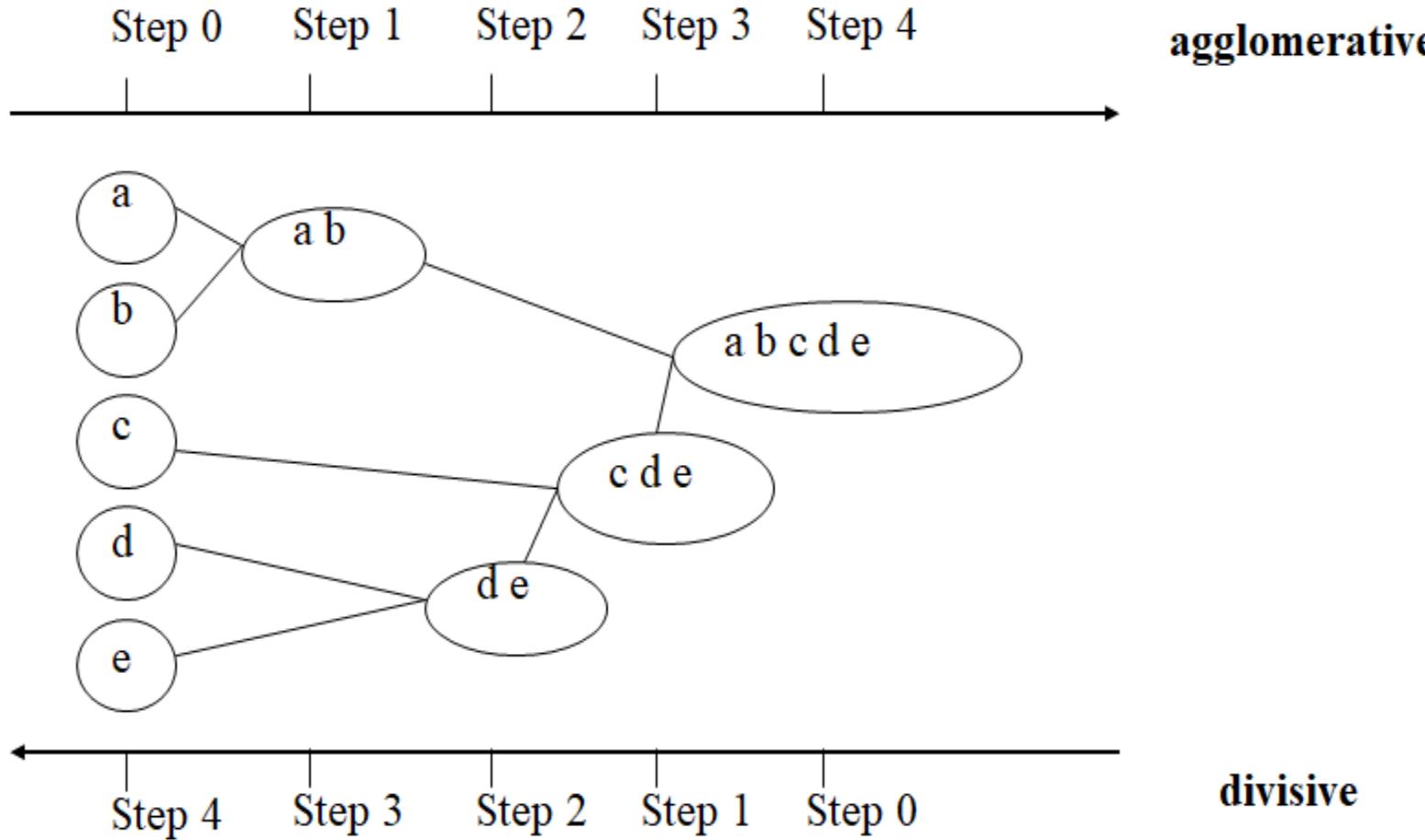


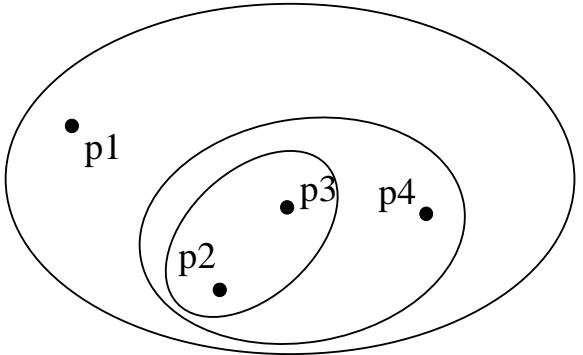


Original Points

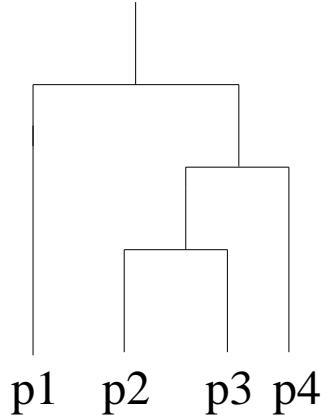


A Partitional Clustering

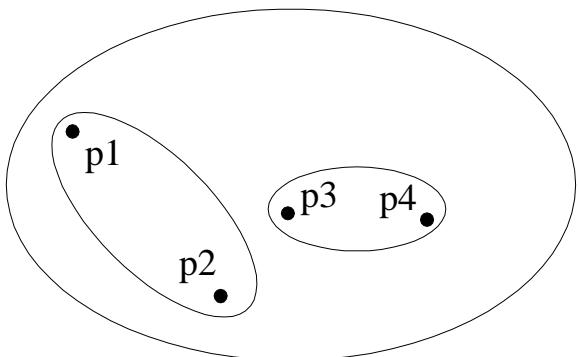




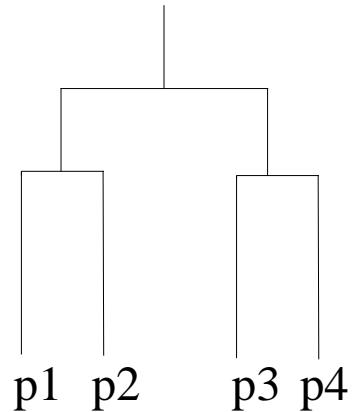
Traditional Hierarchical Clustering



Traditional Dendrogram

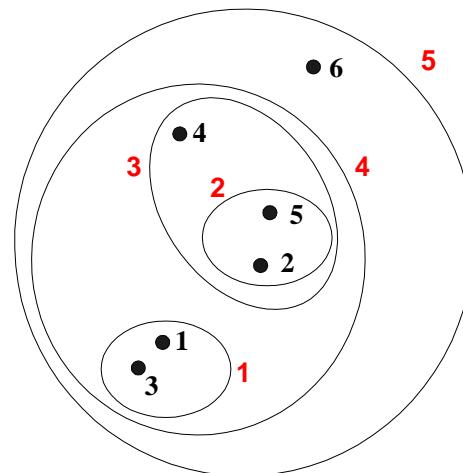
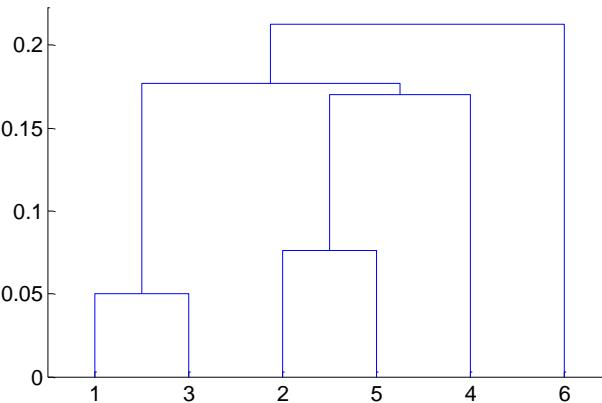


Non-traditional Hierarchical Clustering



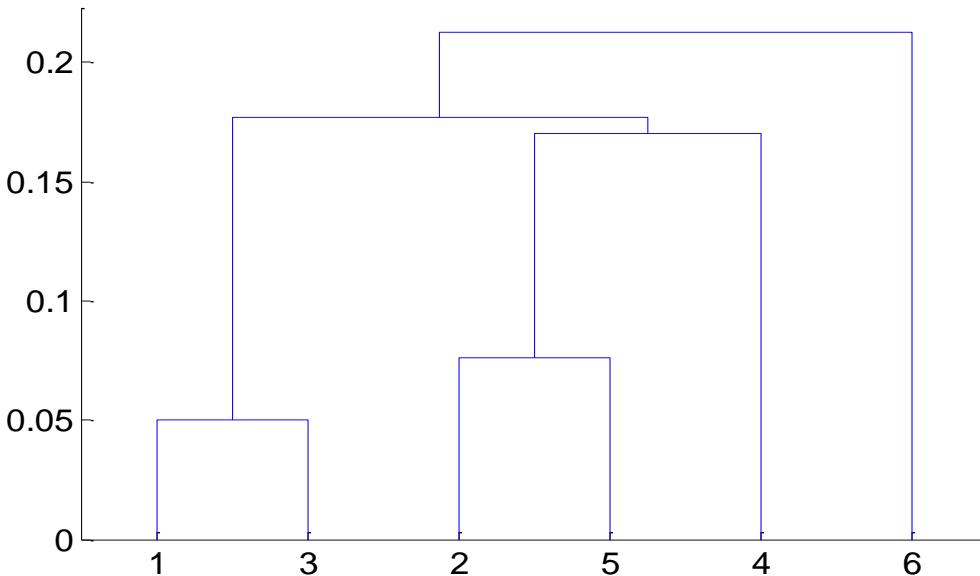
Non-traditional Dendrogram

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
 - A tree like diagram that records the sequences of merges or splits



Hierarchical Clustering: Strength

- Do not have to assume any particular number of clusters
 - Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level



Agglomerative Hierarchical Clustering

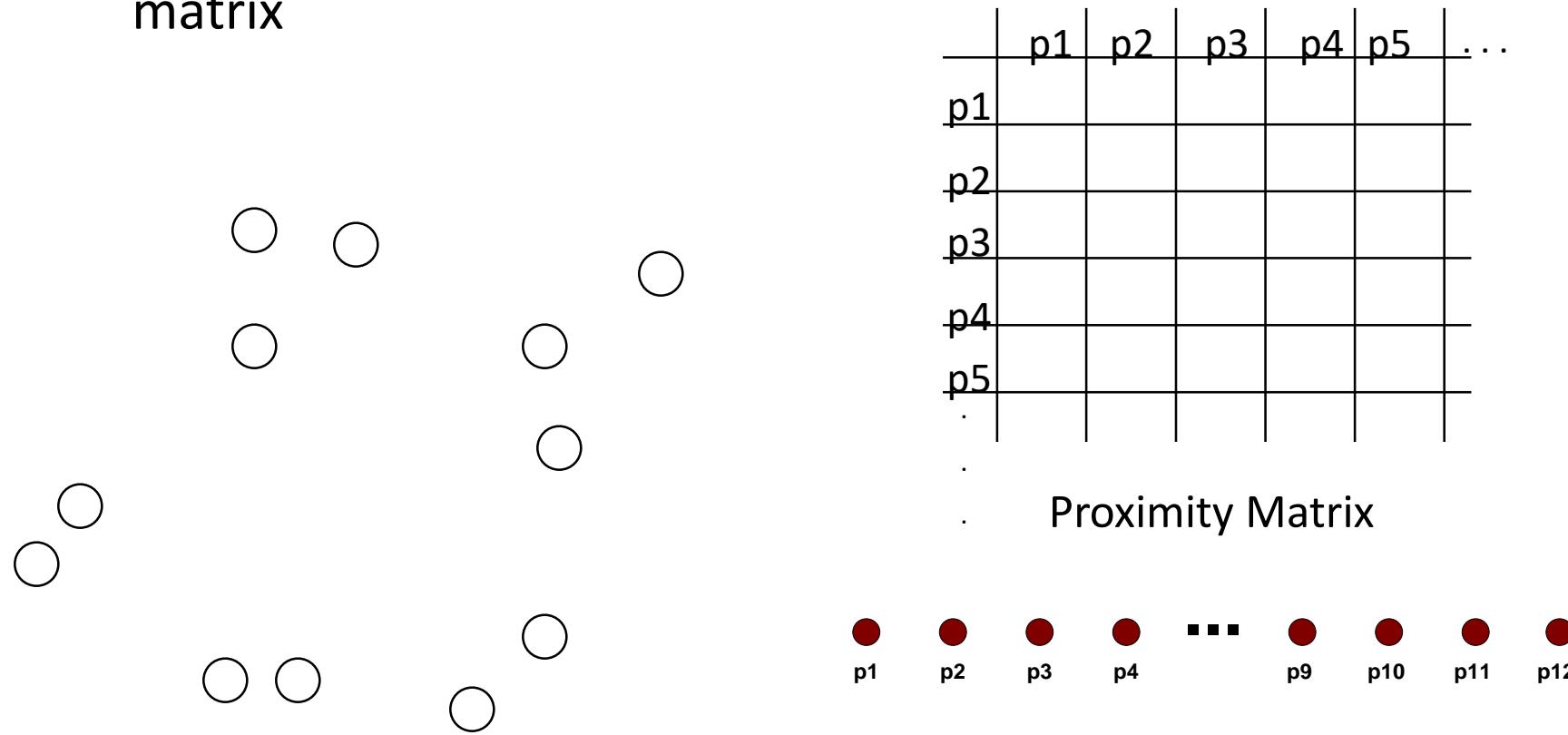
- Popular hierarchical clustering technique
- Basic algorithm is straightforward
 1. Compute the proximity matrix
 2. Let each data point be a cluster
 3. **Repeat**
 4. Merge the two closest clusters
 5. Update the proximity matrix
 6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
 - Different approaches to defining the distance between clusters distinguish the different algorithms

MACHINE INTELLIGENCE

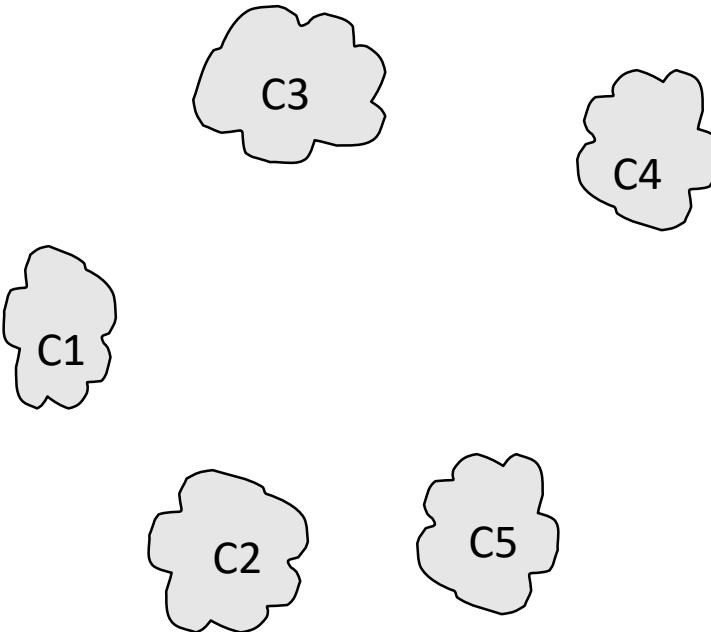
Agglomerative Hierarchical Clustering



- Start with clusters of individual points and a proximity matrix

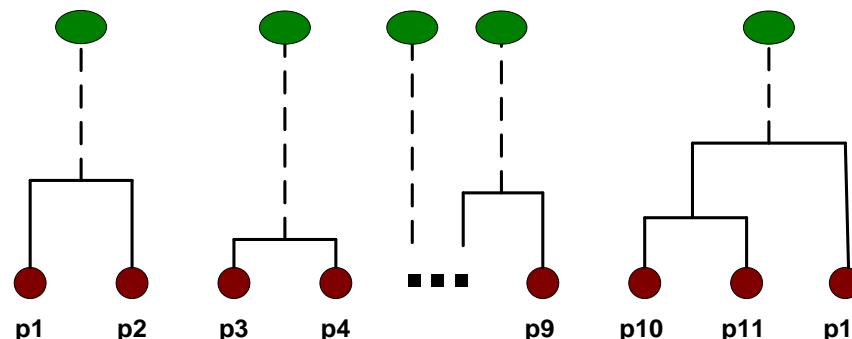


- After some merging steps, we have some clusters

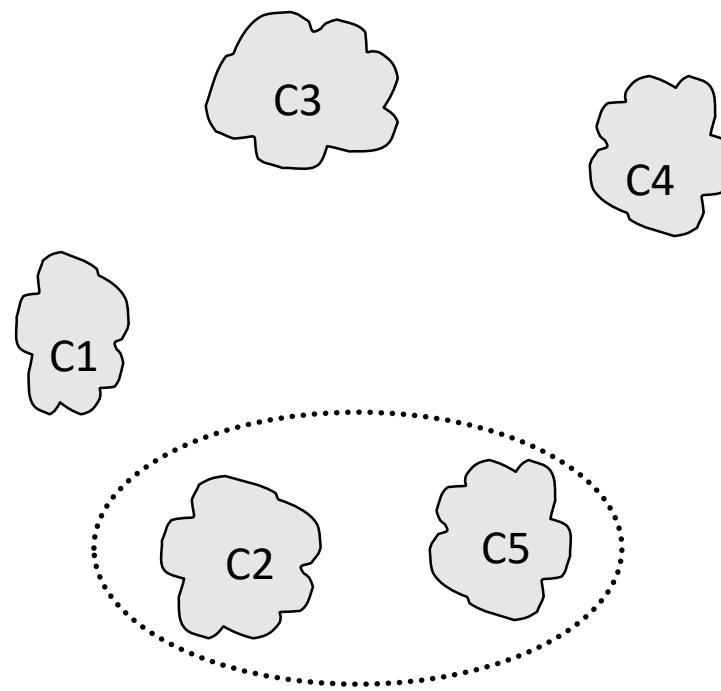


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

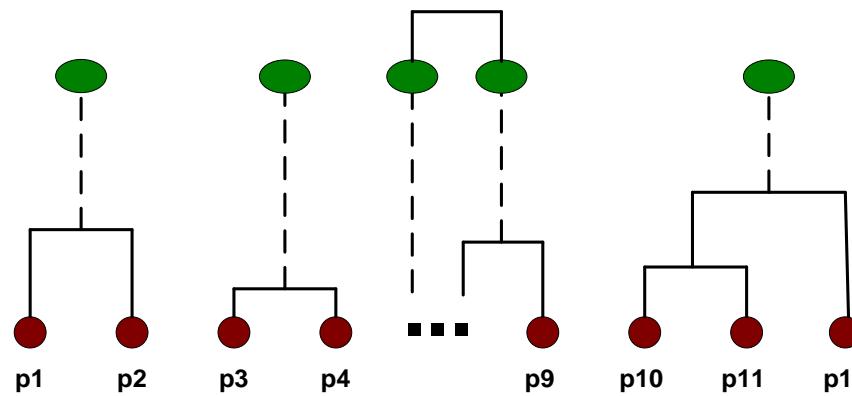


- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



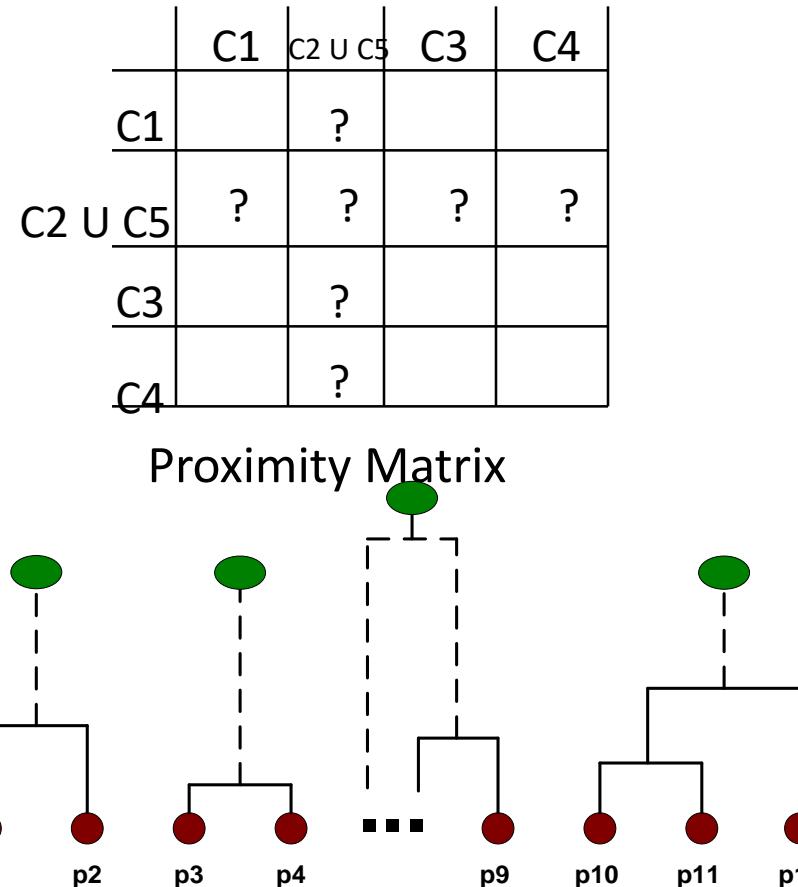
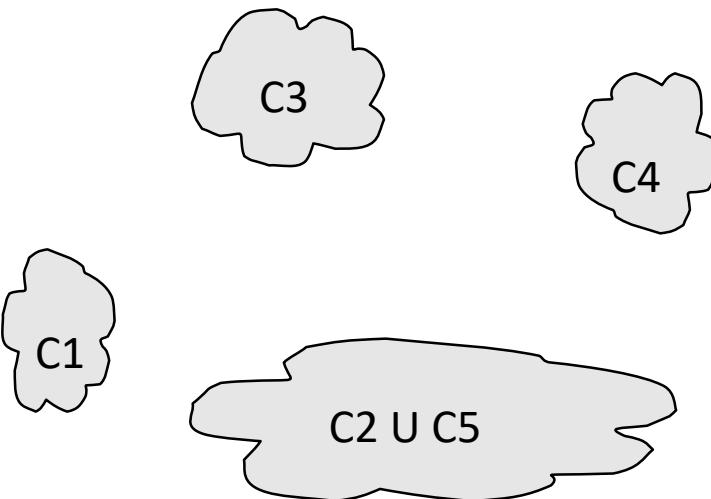
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

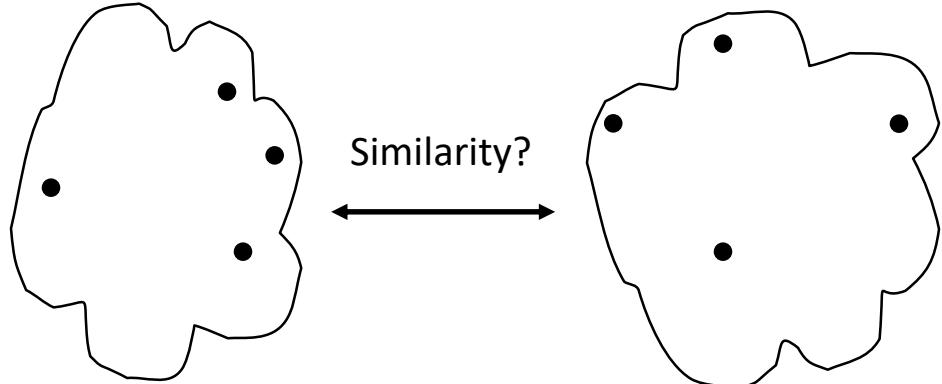


Intermediate Situation: After Merging

- The question is “How do we update the proximity matrix?”



Inter-Cluster Similarity: Methods

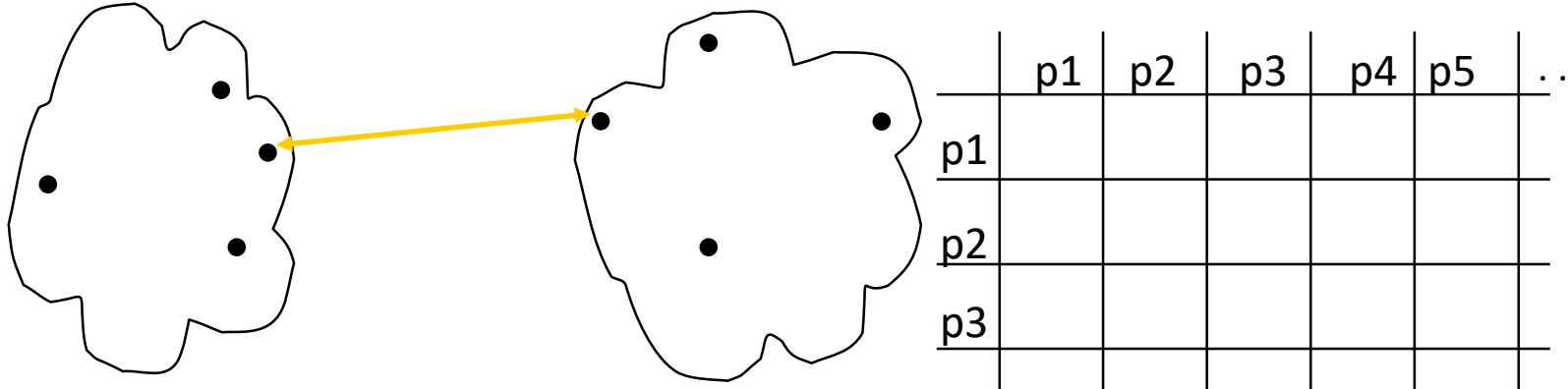


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

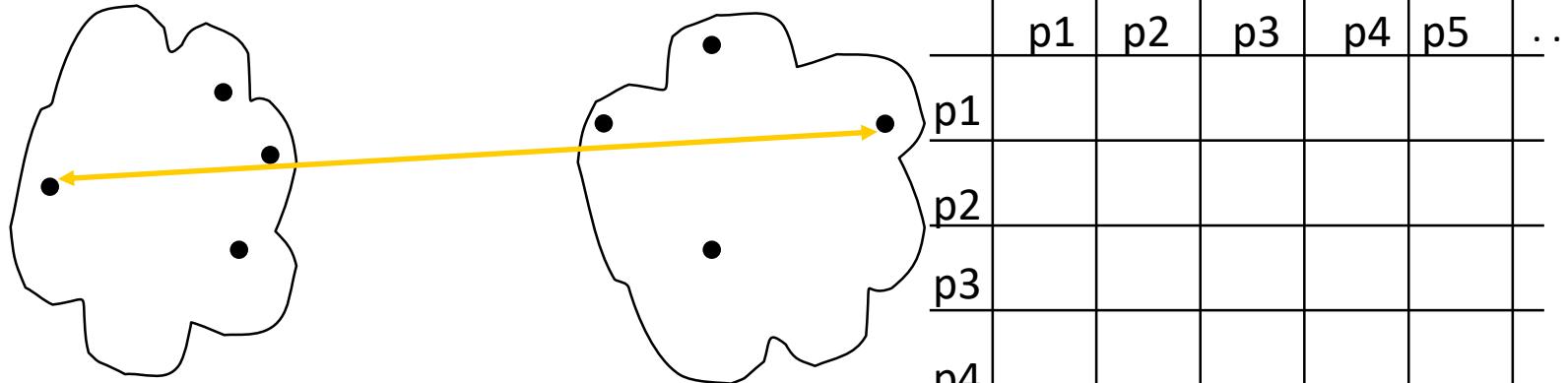
. Proximity Matrix

.



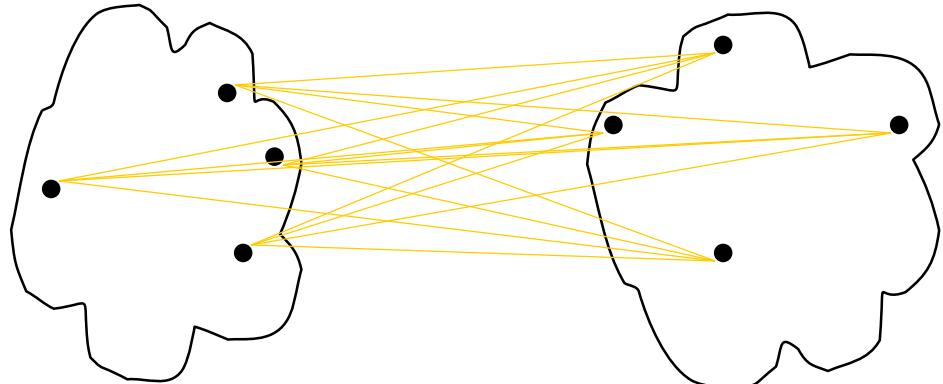
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

. Proximity Matrix



- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

Inter-Cluster Similarity: Methods

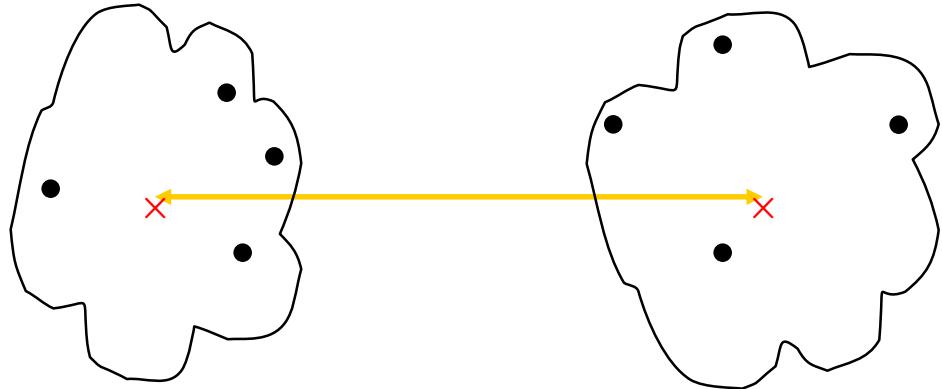


$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

. Proximity Matrix

- MIN
- MAX
- **Group Average**
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error



	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

. Proximity Matrix

- MIN
- MAX
- Group Average
- **Distance Between Centroids**
- Other methods driven by an objective function

Inter-Cluster Similarity: Methods (Example)

assume that there are two clusters: C₁: {a, b} and C₂: {c, d, e}.

	a	b	c	d	e
Feature	1	2	4	5	6

1. Calculate the distance matrix .
2. Calculate three cluster distances between C₁ and C₂.

Single link

$$\begin{aligned}\text{dist}(C_1, C_2) &= \min\{d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)\} \\ &= \min\{3, 4, 5, 2, 3, 4\} = 2\end{aligned}$$

Complete link

$$\begin{aligned}\text{dist}(C_1, C_2) &= \max\{d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)\} \\ &= \max\{3, 4, 5, 2, 3, 4\} = 5\end{aligned}$$

Average

$$\begin{aligned}\text{dist}(C_1, C_2) &= \frac{d(a, c) + d(a, d) + d(a, e) + d(b, c) + d(b, d) + d(b, e)}{6} \\ &= \frac{3 + 4 + 5 + 2 + 3 + 4}{6} = \frac{21}{6} = 3.5\end{aligned}$$

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0



MACHINE INTELLIGENCE

Dr. N MEHALA

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

Dr. N MEHALA

Department of Computer Science and Engineering

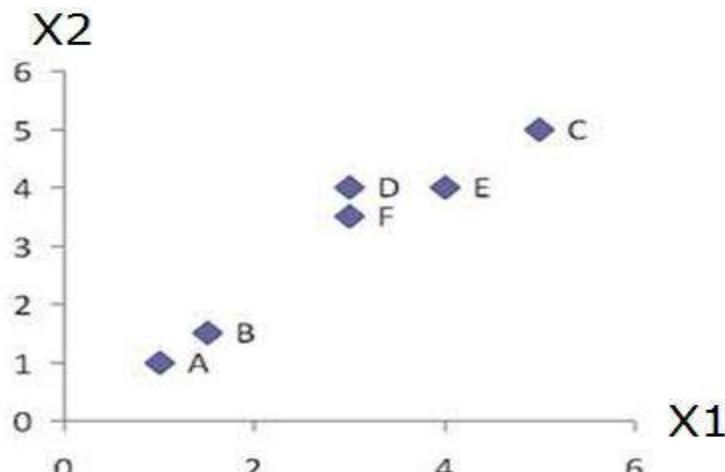
MACHINE INTELLIGENCE

Example1: Hierarchical Clustering



Example1: Hierarchical Clustering

- Problem: clustering analysis with agglomerative algorithm



$$d_{AB} = \sqrt{(1-1.5)^2 + (1-1.5)^2} = \sqrt{\frac{1}{2}} = 0.7071$$

$$d_{DF} = \sqrt{(3-3)^2 + (4-3.5)^2} = 0.5$$

Euclidean distance

data matrix

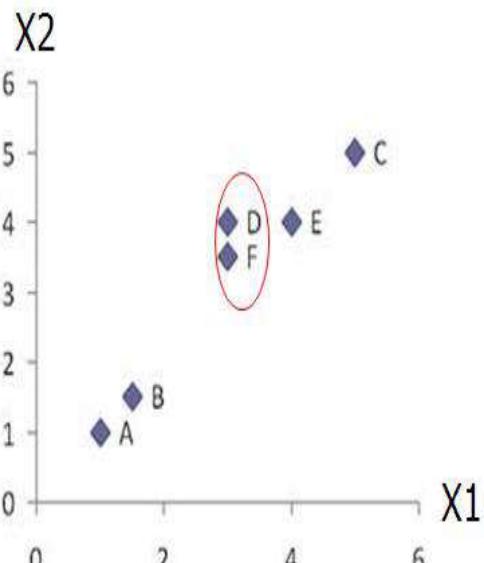
	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

distance matrix

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

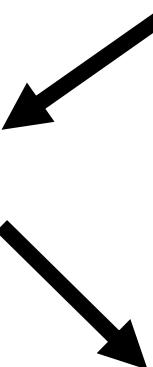
Example1: Hierarchical Clustering

- Merge two closest clusters (iteration 1)



Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	4.95	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00



Example1: Hierarchical Clustering

- Update distance matrix (iteration 1)

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

$$d_{(D,F) \rightarrow A} = \min(d_{DA}, d_{FA}) = \min(3.61, 3.20) = 3.20$$

$$d_{(D,F) \rightarrow B} = \min(d_{DB}, d_{FB}) = \min(2.92, 2.50) = 2.50$$

$$d_{(D,F) \rightarrow C} = \min(d_{DC}, d_{FC}) = \min(2.24, 2.50) = 2.24$$

$$d_{B \rightarrow (D,F)} = \min(d_{BD}, d_{BF}) = \min(1.00, 1.12) = 1.00$$

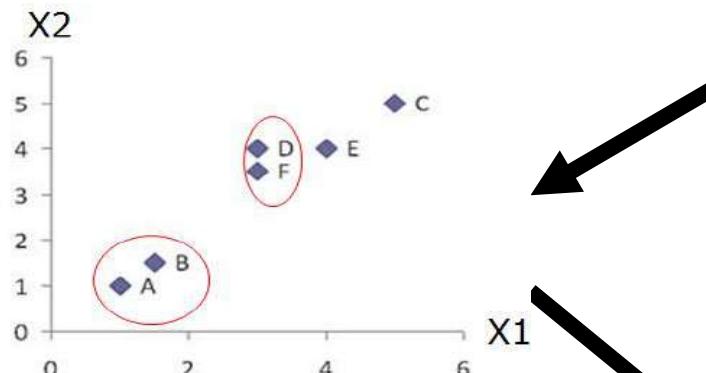
Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

Example1: Hierarchical Clustering

- Merge two closest clusters (iteration 2)



Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Example1: Hierarchical Clustering

- Update distance matrix (iteration 2)

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

$d_{C \rightarrow (A,B)} = \min(d_{CA}, d_{CB}) = \min(5.66, 4.95) = 4.95$
 $d_{(D,F) \rightarrow (A,B)} = \min(d_{DA}, d_{DB}, d_{FA}, d_{FB}) = \min(3.61, 2.92, 3.20, 2.50) = 2.50$
 $d_{E \rightarrow (A,B)} = \min(d_{EA}, d_{EB}) = \min(4.24, 3.54) = 3.54$

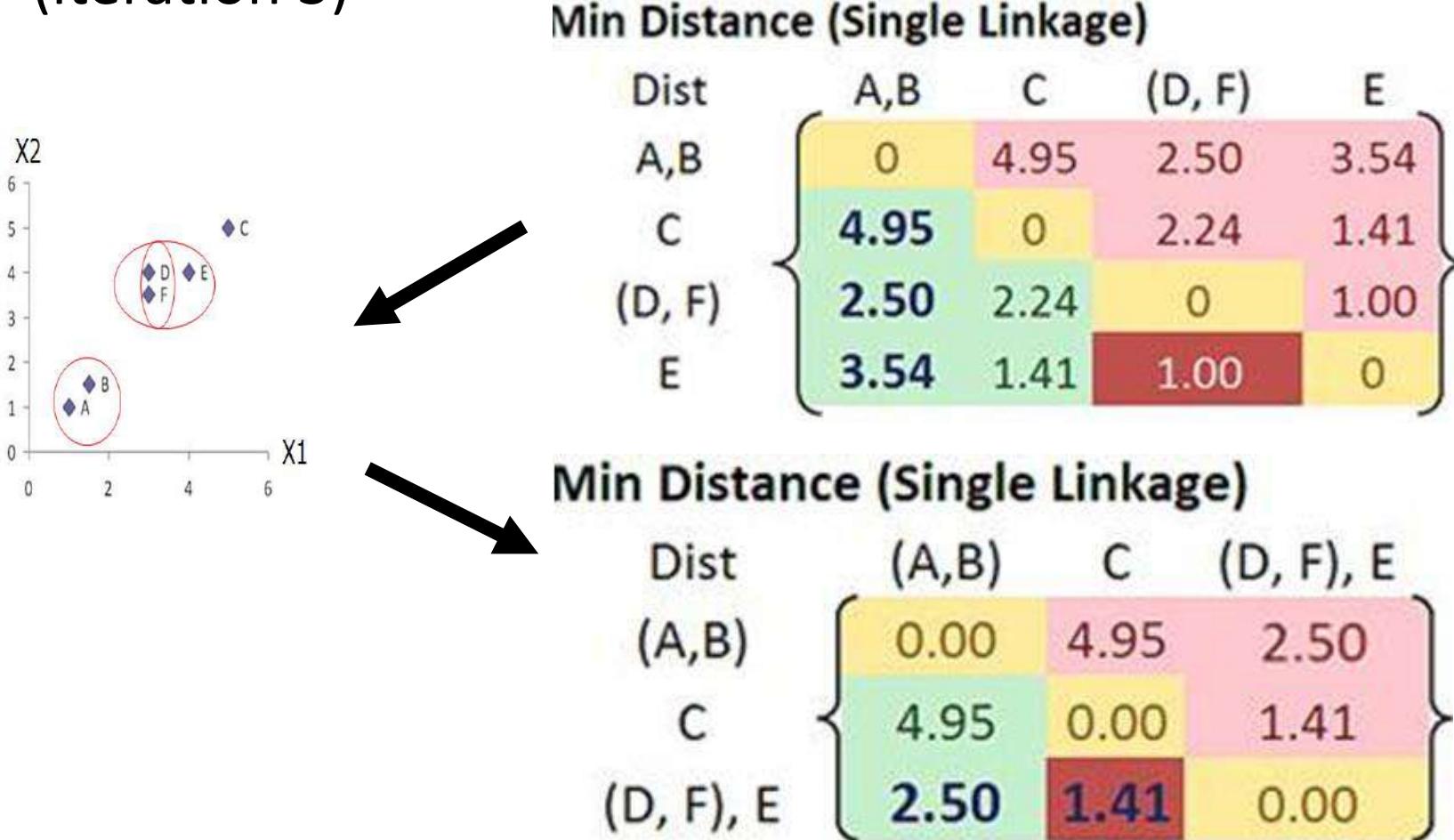
Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D, F)	2.50	2.24	0	1.00
E	3.54	1.41	1.00	0

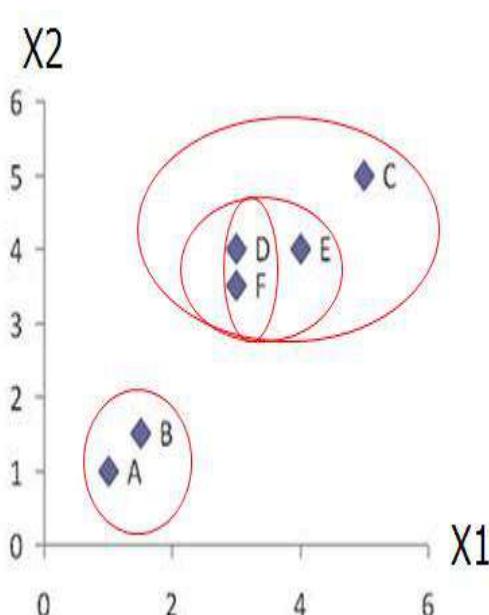
Example1: Hierarchical Clustering

- Merge two closest clusters/update distance matrix (iteration 3)



Example1: Hierarchical Clustering

- Merge two closest clusters/update distance matrix
(iteration 4)

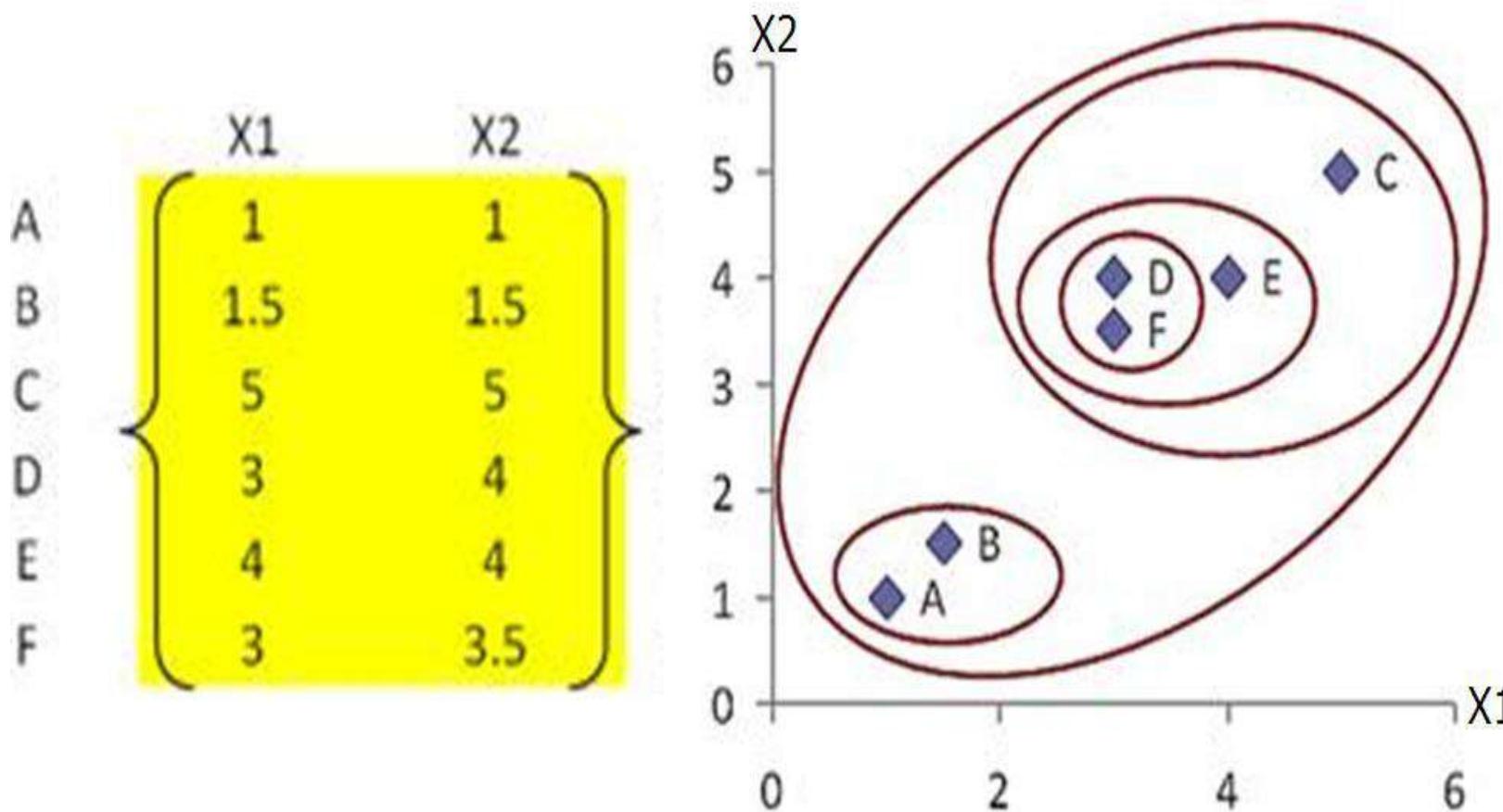


		Min Distance (Single Linkage)			
		Dist	(A,B)	C	(D, F), E
Dist	(A,B)	0.00	4.95	2.50	
	C	4.95	0.00	1.41	
(D, F), E		2.50	1.41	0.00	

		Min Distance (Single Linkage)	
		Dist	(A,B) ((D, F), E),C
Dist	(A,B)	0.00	2.50
	((D, F), E),C	2.50	0.00

Example1: Hierarchical Clustering

- Final result (meeting termination condition)



Hierarchical Clustering: Time and Space Requirement

- $O(N^2)$ space since it uses the proximity matrix.
 - N is the number of points.
- $O(N^3)$ time in many cases
 - There are N steps and at each step the size, N^2 , proximity matrix must be updated and searched
 - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

- Once a decision is made to combine two clusters, it cannot be undone
- Different schemes have problems with one or more of the following:
 - Sensitivity to noise and outliers
 - Difficulty handling different sized clusters and convex shapes
 - Breaking large clusters

- Supervised Vs Unsupervised Learning
- Clustering
- Clustering Types
- Partitional Vs Hierarchical Clustering
- Agglomerative Vs Divisive Clustering
- Agglomerative Hierarchical Clustering: Variants
- Example: Agglomerative Hierarchical Clustering Variants

MACHINE INTELLIGENCE

Resources

- [http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine Learning in Action.pdf](http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine%20Learning%20in%20Action.pdf)
- <http://wwwusers.cs.umn.edu/~kumar/dmbook/.>
- <ftp://ftp.aw.com/cseng/authors/tan>
- <http://web.ccsu.edu/datamining/resources.html>



THANK YOU

Dr. N MEHALA

Department of Computer Science and Engineering

mehala@pes.edu



PES
UNIVERSITY
ONLINE

MACHINE INTELLIGENCE

Dr. N MEHALA

Department of Computer Science
and Engineering

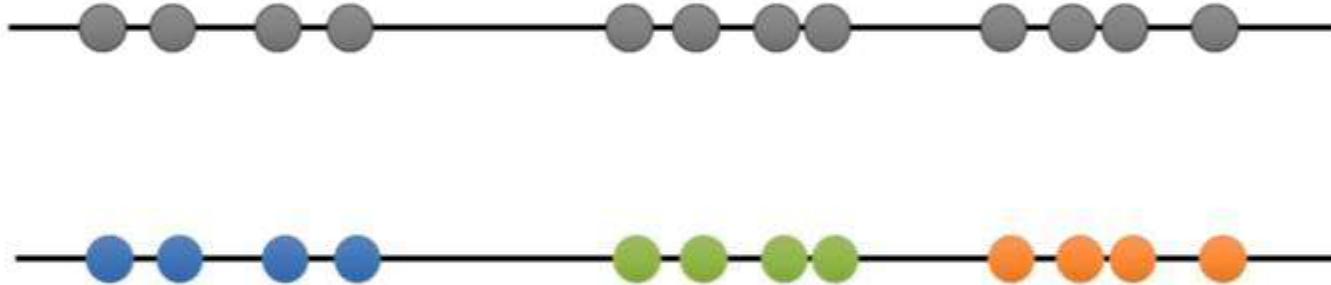
MACHINE INTELLIGENCE

Agenda

- K-Means Clustering Algorithm

MACHINE INTELLIGENCE

K-Means Clustering Algorithm



Initialization Step

a. Input Objects



b. Input K Value: Let K=3

c. Randomly choose K Cluster Centriod Objects



MACHINE INTELLIGENCE

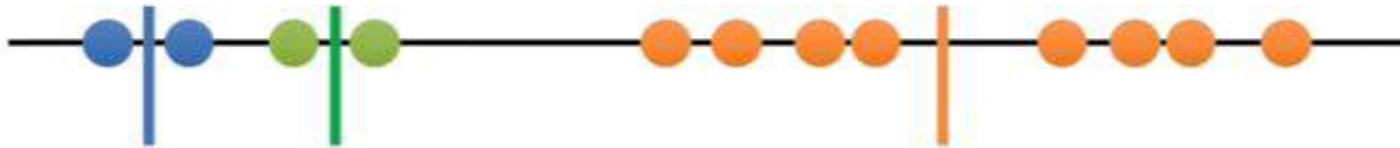
K-Means Clustering Algorithm



ASSIGNMENT STEP

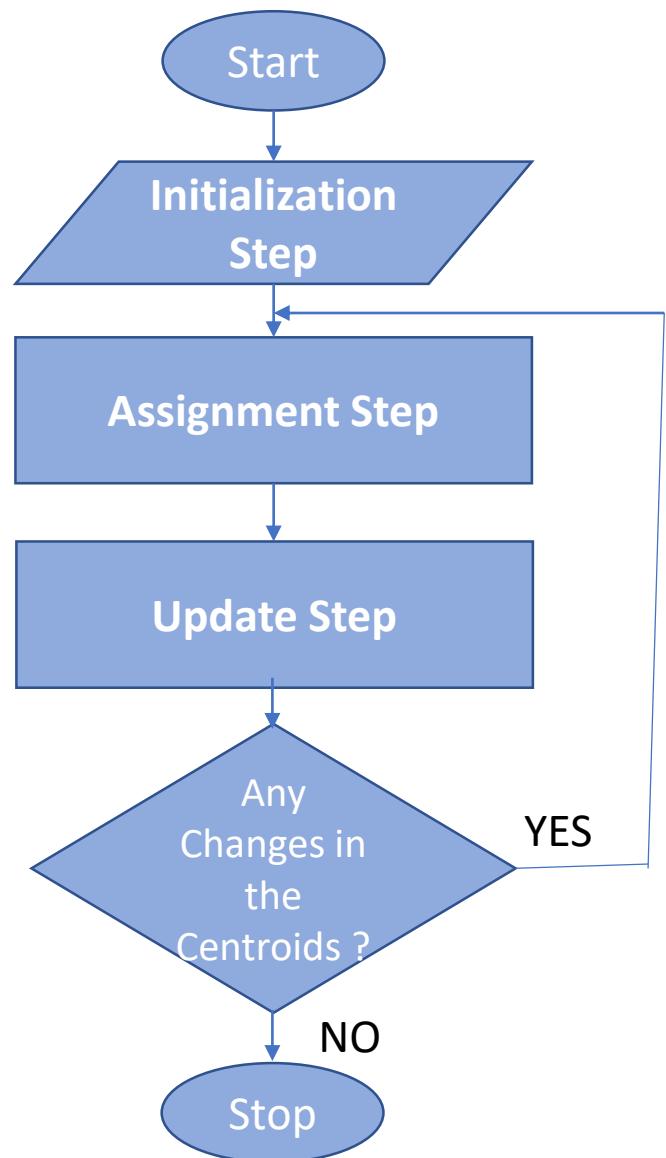
ASSIGNMENT STEP

Since the clustering did not change at all during the last iteration, we're done...



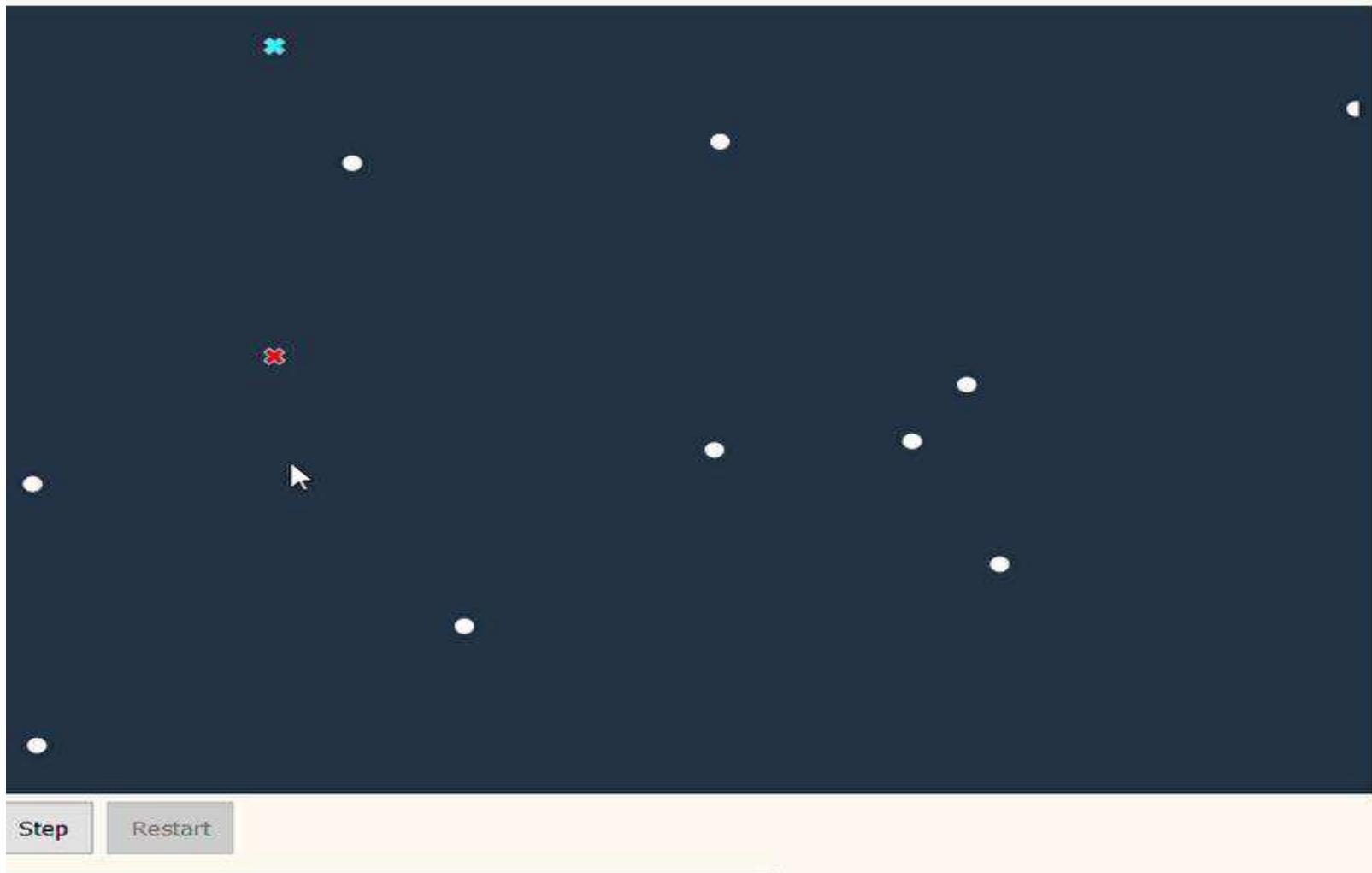
MACHINE INTELLIGENCE

K-Means Clustering Algorithm



MACHINE INTELLIGENCE

K-Means Clustering Algorithm



MACHINE INTELLIGENCE

Resources

- [http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine Learning in Action.pdf](http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine%20Learning%20in%20Action.pdf)
- <http://wwwusers.cs.umn.edu/~kumar/dmbook/.>
- <ftp://ftp.aw.com/cseng/authors/tan>
- <http://web.ccsu.edu/datamining/resources.html>



THANK YOU

Dr. N MEHALA

Department of Computer
Science and Engineering

mehala@pes.edu



PES
UNIVERSITY
ONLINE

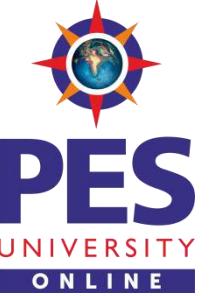
MACHINE INTELLIGENCE

Dr. N MEHALA

Department of Computer Science
and Engineering

MACHINE INTELLIGENCE

Agenda



- K-Means Clustering Algorithm : Example

- 1: Select K points as the initial centroids.
 - 2: **repeat**
 - 3: Form K clusters by assigning all points to the closest centroid.
 - 4: Recompute the centroid of each cluster.
 - 5: **until** The centroids don't change
-

K-Means Clustering: Example

- Suppose we have 4 types of medicines and each medicine/object have two attributes or features as shown in the table. Our goal is to group these objects into K=2 group of medicine based on the two features (pH and weight index).

Object	Attribute 1 (X): weight index	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Step1. Initial value of centroids: Suppose we use medicine A and medicine B as the initial centroids. Let c_1 and c_2 denote the coordinate of the centroids, then $c_1 = (1,1)$ and $c_2 = (2,1)$

Object	Attribute 1 (X): weight index	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

K-Means Clustering: Example

Step2a. Objects-Centroids distance: we calculate the distance between cluster centroid to each object. Let us use **Euclidean distance**, then we have distance matrix at iteration 0 is

$$\begin{matrix} A & B & C & D \\ \left[\begin{matrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{matrix} \right] & X \\ Y \end{matrix} \quad \mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) & group-1 \\ c_2 = (2,1) & group-2 \end{matrix}$$

- First row of the distance matrix corresponds to the distance of each object to the first centroid
- Second row is the distance of each object to the second centroid

Distance from the Medicine $C = (4,3)$ to the first centroid $C_1 = (1,1)$:

$$\sqrt{(4-1)^2 + (3-1)^2} = 3.61$$

Distance from the Medicine $C = (4,3)$ to second centroid $C_2 = (2,1)$:

$$\sqrt{(4-2)^2 + (3-1)^2} = 2.83$$

K-Means Clustering: Example

Step2b. Objects clustering: we assign each object based on minimum distance. Thus medicineA is assigned to group1, medicineB is assigned to group2, medicineC is assigned to group2 and medicineD is assigned to group2. The element of Group matrix is 1 if and only if the object is assigned to that group.

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad c_1 = (1,1) \text{ group-1} \quad c_2 = (2,1) \text{ group-2}$$
$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \begin{matrix} \text{group-1} \\ \text{group-2} \end{matrix}$$

A B C D

K-Means Clustering: Example

Step3. Recompute Centroids: Knowing the members of each group, now we compute the new centroid of each group based on these new memberships. Group1 only has one member thus the centroid remains in $C_1=(1,1)$. Group 2 now has three members, thus the centroid is the average coordinate among the three members: $C_2=\left(\left(\frac{2+4+5}{3}\right), \left(\frac{1+3+4}{3}\right)\right)=\left(\frac{11}{3}\right), \left(\frac{8}{3}\right)$

$$\begin{matrix} A & B & C & D \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} & X \\ Y \end{matrix}$$

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \begin{matrix} \text{group-1} \\ \text{group-2} \end{matrix}$$

$A \quad B \quad C \quad D$

MACHINE INTELLIGENCE

K-Means Clustering: Example (Iteration1)



<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	
1	2	4	5	<i>X</i>
1	1	3	4	<i>Y</i>

Step2a. Objects-Centroids distance

$$\mathbf{D}^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \mathbf{c}_1 = (1,1) \quad \text{group-1}$$

$$\\ \mathbf{c}_2 = \left(\frac{11}{3}, \frac{8}{3}\right) \quad \text{group-2}$$

Step2b. Objects clustering

$$\mathbf{G}^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \text{group-1}$$

$$\\ \text{group-2}$$

$$\begin{array}{cccc} A & B & C & D \end{array}$$

Step3. Recompute Centroids

$$\mathbf{c}_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = \left(1\frac{1}{2}, 1\right) \text{ and } \mathbf{c}_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = \left(4\frac{1}{2}, 3\frac{1}{2}\right)$$

K-Means Clustering: Example (Iteration2)

Step2a. Objects-Centroids distance

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \mathbf{c}_1 = (1\frac{1}{2}, 1) \quad \text{group - 1}$$

$$c_2 = (4\frac{1}{2}, 3\frac{1}{2}) \quad \text{group - 2}$$

Step2b. Objects clustering

$$\mathbf{G}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \hline A & B & C & D \end{bmatrix} \quad \text{group - 1}$$

$$group - 2$$

$$\mathbf{G}^2 = \mathbf{G}^1$$

Object	Feature 1 (X): weight index	Feature 2 (Y): pH	Group (result)
Medicine A	1	1	1
Medicine B	2	1	1
Medicine C	4	3	2
Medicine D	5	4	2

K-Means Clustering: Evaluation

- Most common measure is Sum of Squared Error (SSE)
 - For each point, the error is the distance to the nearest cluster
 - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
 - can show that m_i corresponds to the center (mean) of the cluster
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K, the number of clusters
 - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

K-Means Clustering: Summary

- Pros: Easy to implement
- Cons: Can converge at local minima; slow on very large datasets
- Works with: Numeric values

General approach to k-means clustering

1. Collect: Any method.
2. Prepare: Numeric values are needed for a distance calculation, and nominal values can be mapped into binary values for distance calculations.
3. Analyze: Any method.
4. Train: Doesn't apply to unsupervised learning.
5. Test: Apply the clustering algorithm and inspect the results. Quantitative error measurements such as sum of squared error can be used.
6. Use: Anything you wish. Often, the clusters centers can be treated as representative data of the whole cluster to make decisions

K-Means Clustering: Summary

- Initial centroids are often chosen randomly
 - Clusters produced vary from one run to another
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation etc.
- Most of the convergence happens in the first few iterations
 - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Let, n = number of points, K = number of clusters,
 I = number of iterations, d = number of attributes
 - Complexity is $O(n * K * I * d)$

- K-Means Clustering (Partitional Clustering)
- Example: K-Means Clustering
- K-Means Clustering: Evaluation

MACHINE INTELLIGENCE

Resources

- [http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine Learning in Action.pdf](http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine%20Learning%20in%20Action.pdf)
- <http://wwwusers.cs.umn.edu/~kumar/dmbook/.>
- <ftp://ftp.aw.com/cseng/authors/tan>
- <http://web.ccsu.edu/datamining/resources.html>



THANK YOU

Dr. N MEHALA

Department of Computer Science and Engineering

mehala@pes.edu



MACHINE INTELLIGENCE

Srinivas K.S

Department of Computer Science
and Engineering

MACHINE INTELLIGENCE

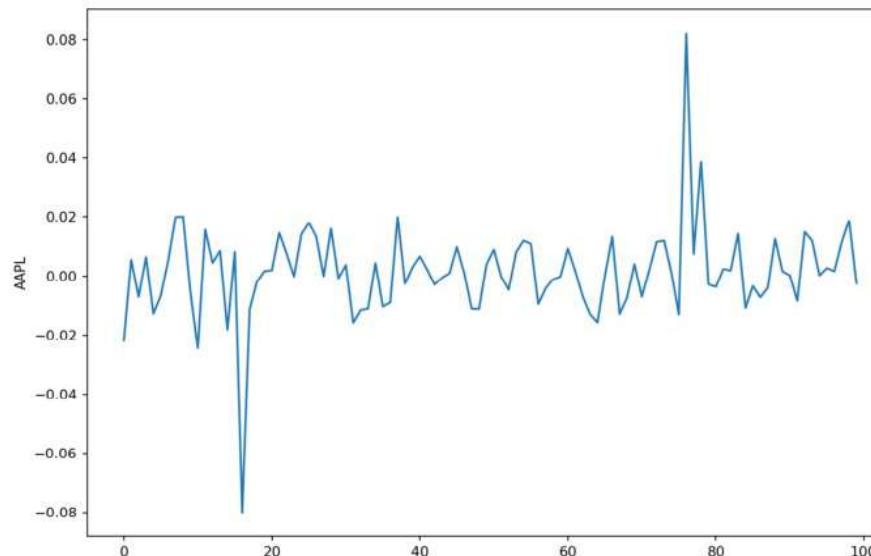
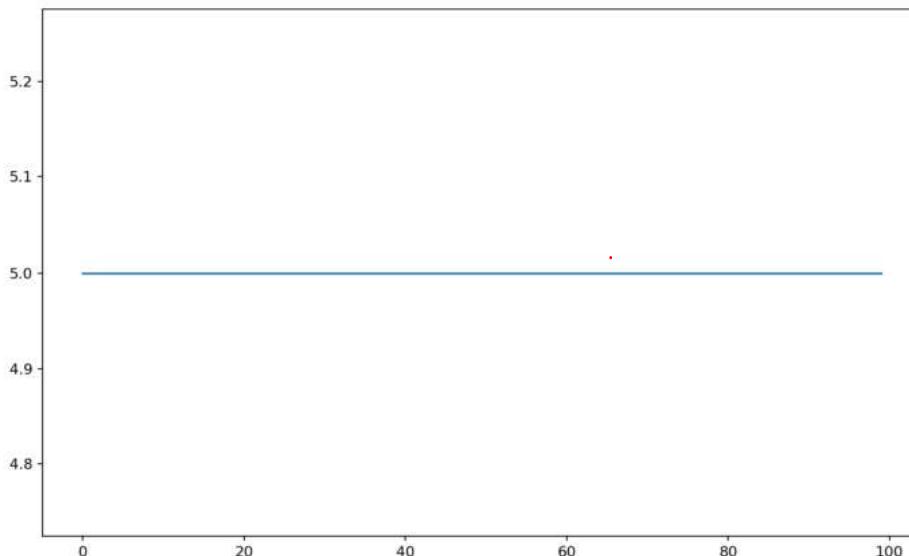
Module 4 [Unsupervised Learning]

SRINIVAS K S

Department of Computer Science and Engineering

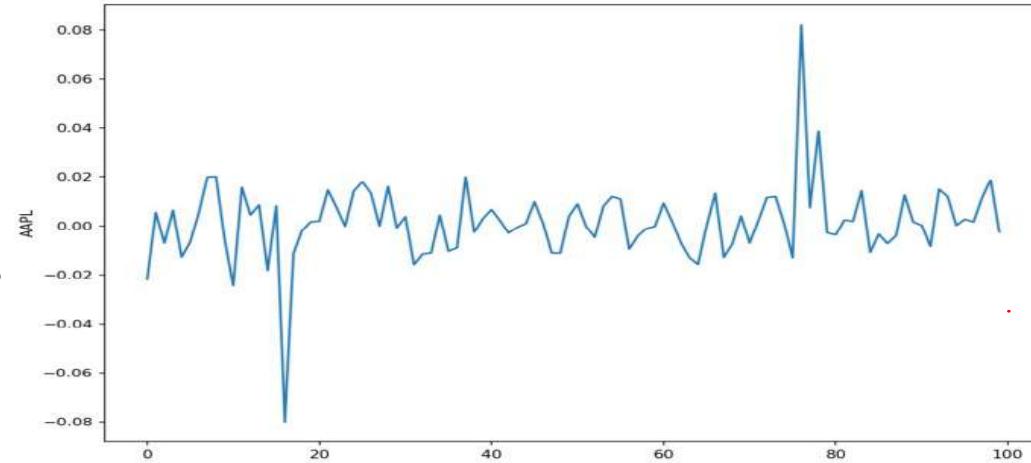
Variance is Both Our Enemy and Our Friend

- Which of the data patterns do you find more interesting?
- Data with no variance has **no uncertainty**, so there is **nothing** for us to **predict** or explain



Need For Principal Component Analysis (PCA)

- Variance is basically how much the data bounces up and down by. The more bouncy it is, the harder it is to predict or explain
- But amidst all that noise, bouncy data also contains signal (a.k.a. information)
- bouncy data is both infinitely more interesting as the target variable of your model and infinitely more useful as a feature variable inside your model



Need For Principal Component Analysis (PCA)

- Enemy: **Target variable variance** makes it harder to **predict** as the target variable creates **uncertainty**
- Friend: To build a good model we need features with signals and when **features have non-zero co-variances we have features with signals**

- I currently working on molecular data to predict toxicity. We have over 1024 features and less than 700 samples. Such a model would **absolutely overfit**
- We are living in an age of information overload

Need For Principal Component Analysis (PCA)



- Now if only there were an algorithm that could transform our 1024 features into an ideal set of feature
- Ideal:
 - High Variance - Signals
 - Uncorrelated
 - Few Features - Overfit
- Uncorrelated: Imagine a room full of stock traders all working hand in glove. If all of them are wrong the stock market would tank

- I currently working on molecular data to predict toxicity. We have over 1024 features and less than 700 samples. Such a model would absolutely overfit

- PCA (Principal Components Analysis) gives us our ideal set of features

Need For Principal Component Analysis (PCA)

Curse of Dimensionality:

Data Analysis become difficult as the dimensionality of the data increases.

As dimensionality increases, the data becomes increasingly sparse in the space.

For Classification, high dimensional data that is sparse in space means there is not enough examples for training/creating the model appropriately so that it can assign labels to new examples properly.

For Clustering, the density and distance between the data objects become less meaningful in high dimension.

That is the reason why Classification and Clustering Algorithms have problem with high dimensionality.

Food for the thought:

- *If you have 4 data points and 10 features each, is it a high dimensional dataset?*

- *If you have 100K features and 10M samples, is it a high dimensional dataset?*

Need For Principal Component Analysis (PCA)

- So in order to reduce dimensionality of the data, not only **PCA** but **SVD** is also used.
- PCA and SVD are two linear algebra techniques (particularly for continuous data) *to project data from high dimensional space to low dimensional space.*
- There are other Dimensionality reduction techniques also like
 1. ISOMAP
 2. Backward feature elimination
 3. F/w feature extraction
 4. Linear Discriminant Analysis
 5. Neural Autoencoder
 6. t-distributed stochastic neighbour embedding
 7. Missing value ratios etc.

Need For Principal Component Analysis (PCA)

PCA creates a set of **principal components**

- **Rank ordered by variance** (the first component has higher variance than the second, and so on)
- **Uncorrelated**
- **Low in number** (we can throw away the lower ranked components as they contain little signal).
- **Linear Combination** of original attributes.
- Are **orthogonal** to each other.
- **Capture maximum variation** in the data.
- Trying to make sense of a *large number of feature variables for prediction* is very hard
- So we do dimensionality reduction – reducing feature space
 - Feature Extraction
 - Feature Elimination

Feature Elimination:

- Drop ones that don't contribute to the prediction. Prevents overfitting and easy to interpret but signals from features that are dropped are lost.

Retained variables contain most valuable parts of the old variables – thus some amount of signal is retained

Feature Extraction:

- Create new variables (may be the same number of them) where **each of these new variables are a combination of the old variables**
- Each of these new variables also contain a signal on ***how well they predict the target variable***
- We ***order (rank) the new variables in their order of how well they predict the target variable***
- Less ***Significant ones can be chopped off***

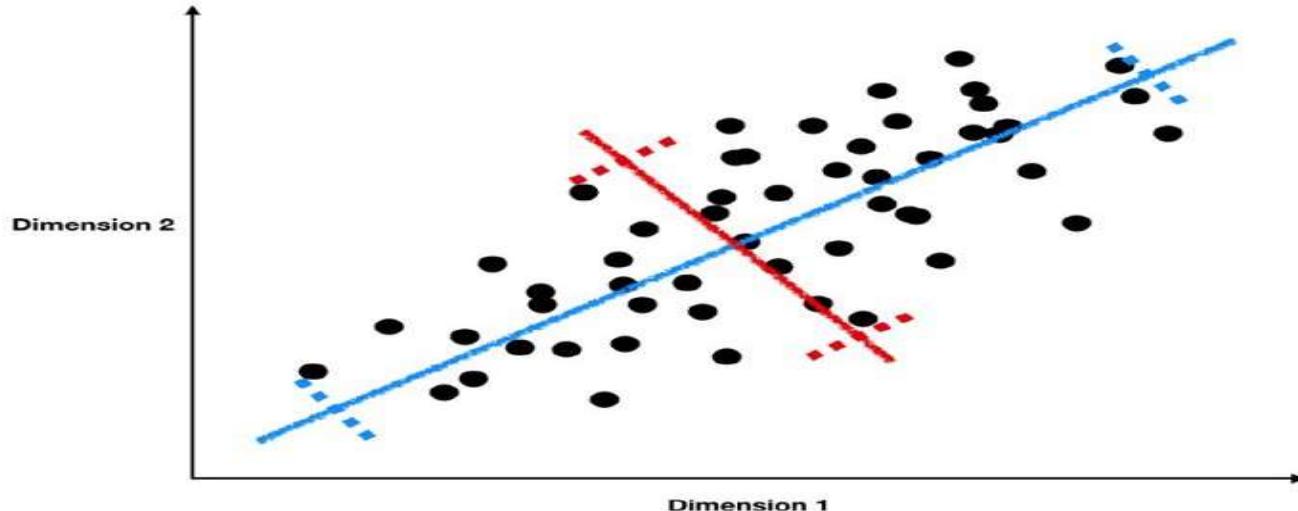
Need For Principal Component Analysis (PCA)

Principal component analysis is a technique of feature extraction.

- It combines the input variables in a specific way, then the “least important” variables are dropped.
- Still retaining the most valuable parts of all of the variables!
- The assumptions of the model gives us the additional benefit of linear independence
- **When should I use PCA?**
- Do you want to **reduce the number of variables**, but aren't able to identify variables to completely remove from consideration?
- Do you want to ensure your variables **are independent** of one another?
- Are you comfortable making your independent variables **less interpretable**?

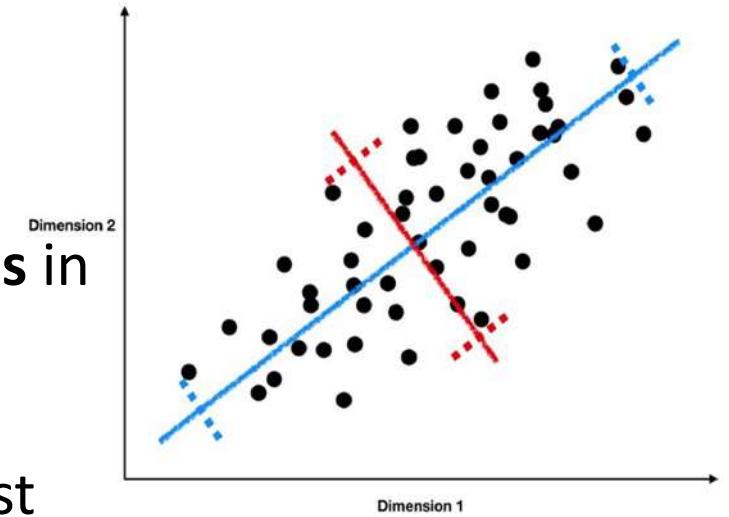
Principal Component Analysis (PCA) – Intuitive understanding

- At the very start of the process, PCA asks what is the strongest underlying trend in the feature set (we will call this component 1). *The first principal component is the direction in space along which projections have the largest variance.*
- Next PCA asks what is the second strongest underlying trend in the feature set that also happens to be **uncorrelated with component 1** (we will call it component 2)?
The second principal component is the direction which maximizes variance among all directions orthogonal to the first.
- Then PCA asks what is the third strongest underlying trend in the feature set that also happens to be **uncorrelated with both components 1 and 2 (we will call it component 3)**?
- And so on...*The kth component is the variance-maximizing direction orthogonal to the previous k - 1 components.*



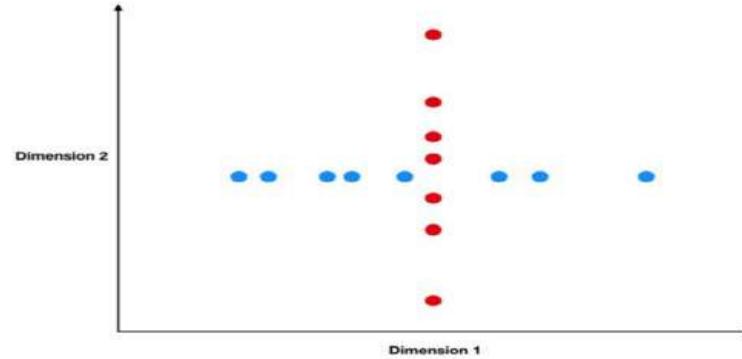
- In the above picture does the blue line indicate a better trend than the red line ?
- If you looked at it as a linear regression problem you would say Yes!
- The Blue line has more variance than the red line so given D1 it is easier to predict D2

- So our blue line is the Principal Component 1
- The Principal component is a linear combination of the weighted combination of the features
- The weighted sums that **best express the underlying trends** in our feature set – High Variance
- Now for component 2, we want to find the second strongest underlying trend with the *added condition that it is uncorrelated to component 1.*
- In statistics, trends and data that are orthogonal (a.k.a. perpendicular) to each other are uncorrelated.



Need For Principal Component Analysis (PCA)

- In the plot on the right
- All of the variation in the blue feature is horizontal and all the variation in the red one is vertical.
- Thus, as the blue feature changes (horizontally), the red feature stays completely constant as it can only change vertically.

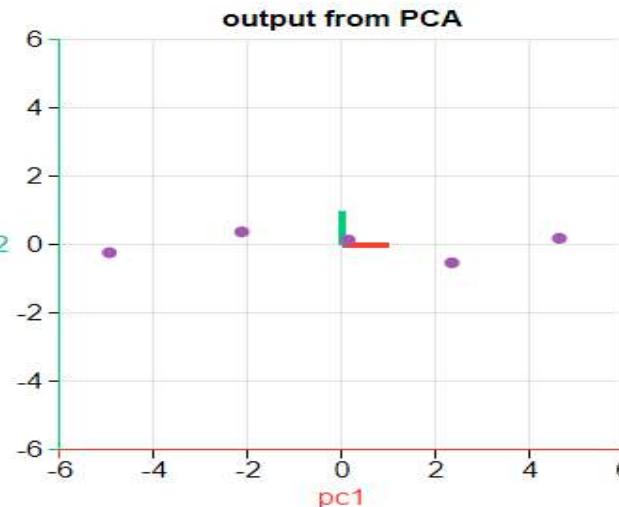
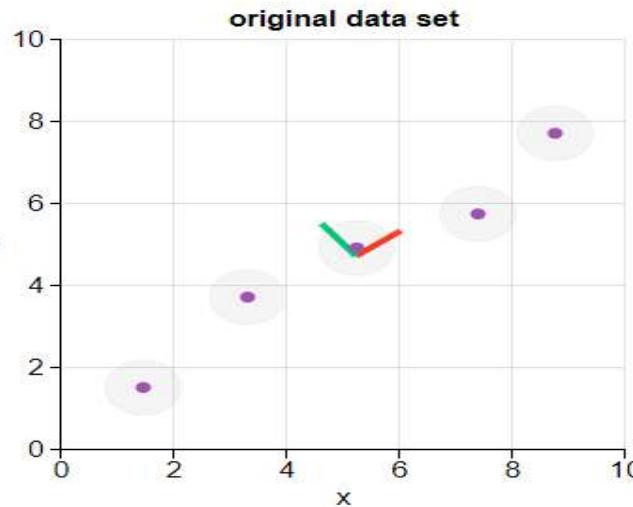


Need for Eigen Vectors and the proofs

So far we have established that we would need to change the basis vectors and use basis vectors that have the highest degree of variance.

PCA a formal 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.

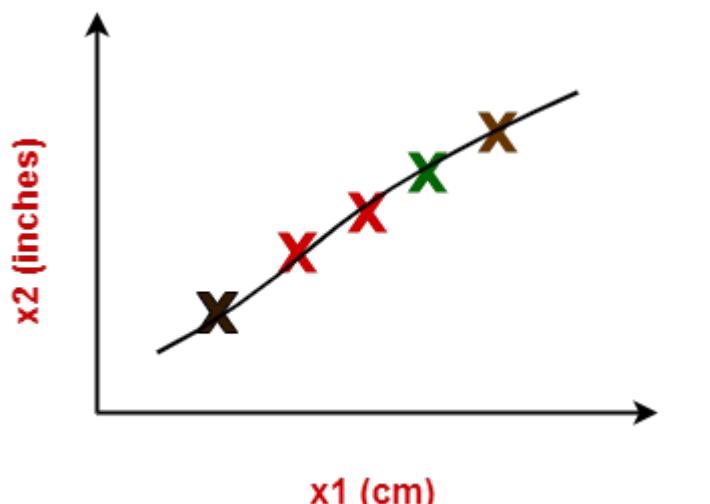


Dimension Reduction

- Process of converting a data set having vast dimensions into a data set with lesser dimensions.
- Ensures that the converted data set conveys similar information concisely.

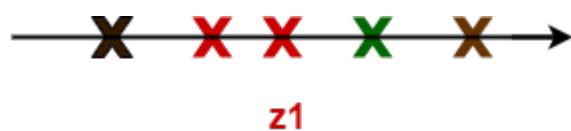
Example

- Consider the following graph shows dimensions x_1 and x_2
- x_1 represents the measurement of several objects in cm
- x_2 represents the measurement of several objects in inches



Using dimension reduction techniques-

- We convert the dimensions of data from 2 dimensions (x_1 and x_2) to 1 dimension (z_1).
- It makes the data relatively easier to explain.

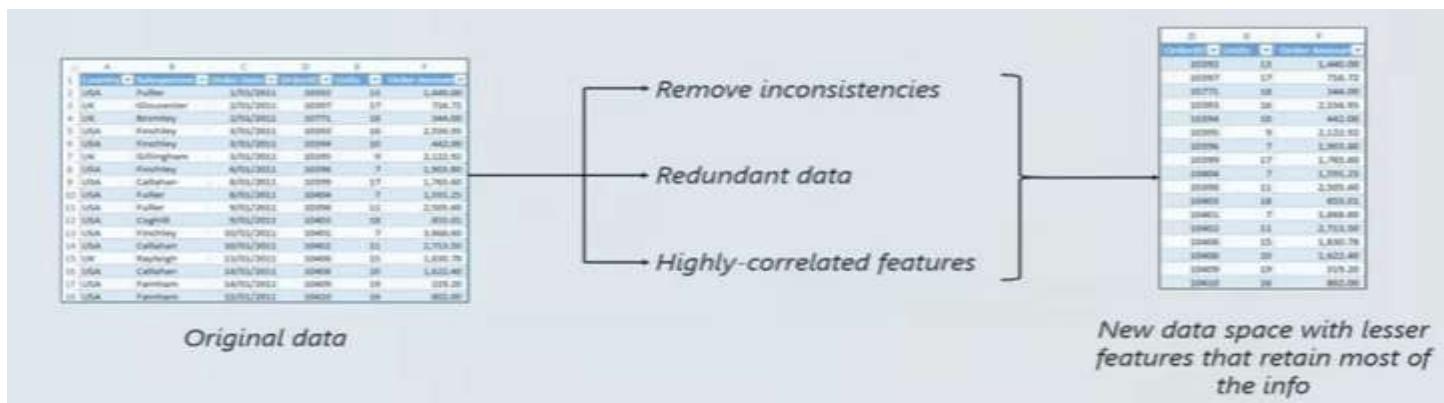


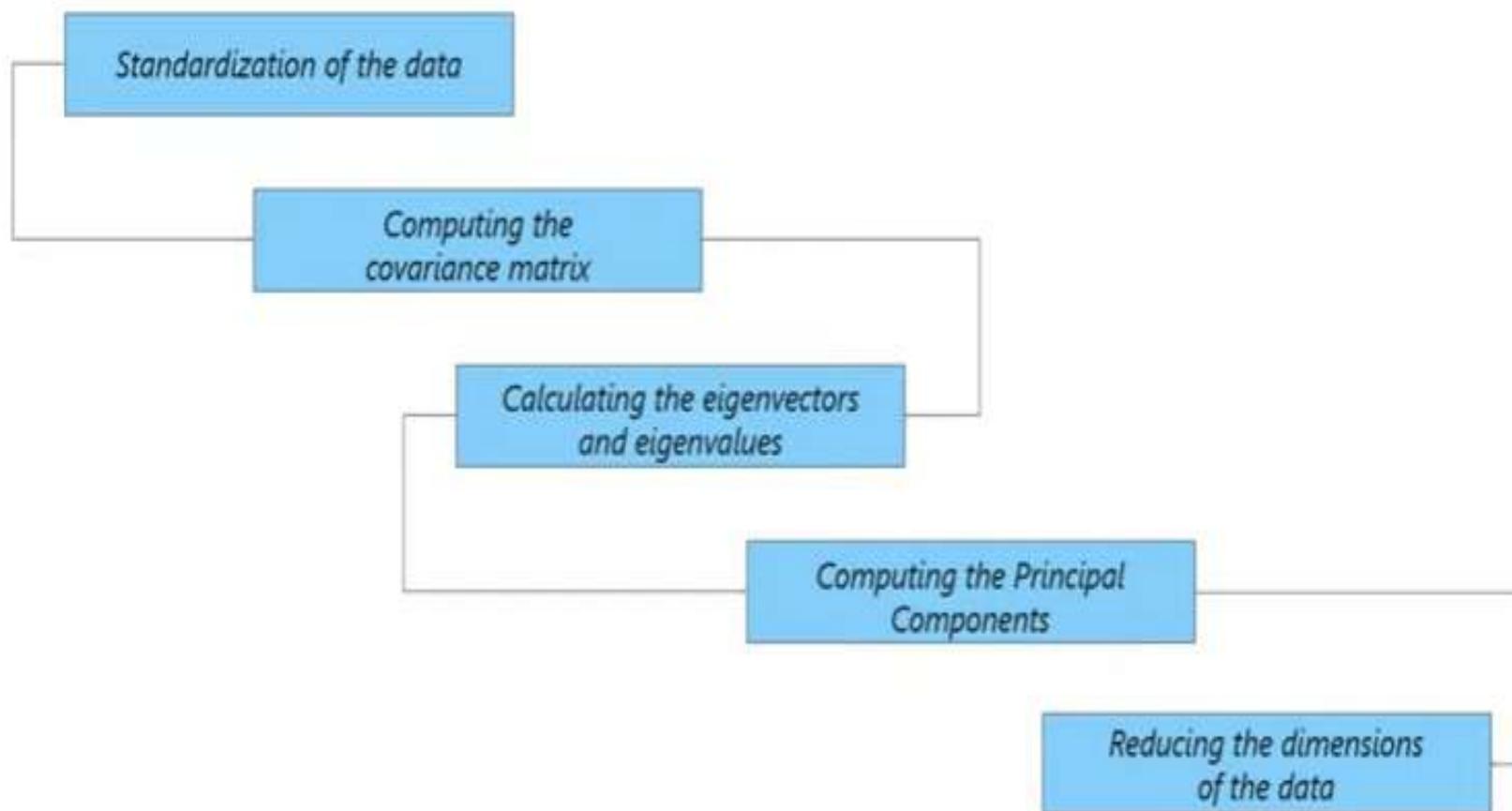
Benefits of Dimensionality Reduction

- Compresses the data and thus reduces the storage space requirements.
- Reduces the time required for computation since less dimensions require less computation.
- Eliminates the redundant features.
- Improves the model performance.

Dimensionality Reduction Technique : PCA

- Well-known dimension reduction technique
- Transforms the variables into a new set of variables called as principal components.
- The first principal component (PC1) accounts for most of the possible variation of original data
- The second principal component (PC2) does its best to capture the variance in the data.
- There can be only two principal components for a two-dimensional data set.





Step1: Standardization of Data

- Scaling your data in such a way that all the variables and their values lie within a similar range.

Rating	# of downloads
5	1383
3	668
2	763
5	839
1	342

Rating feature ranges between 0-5

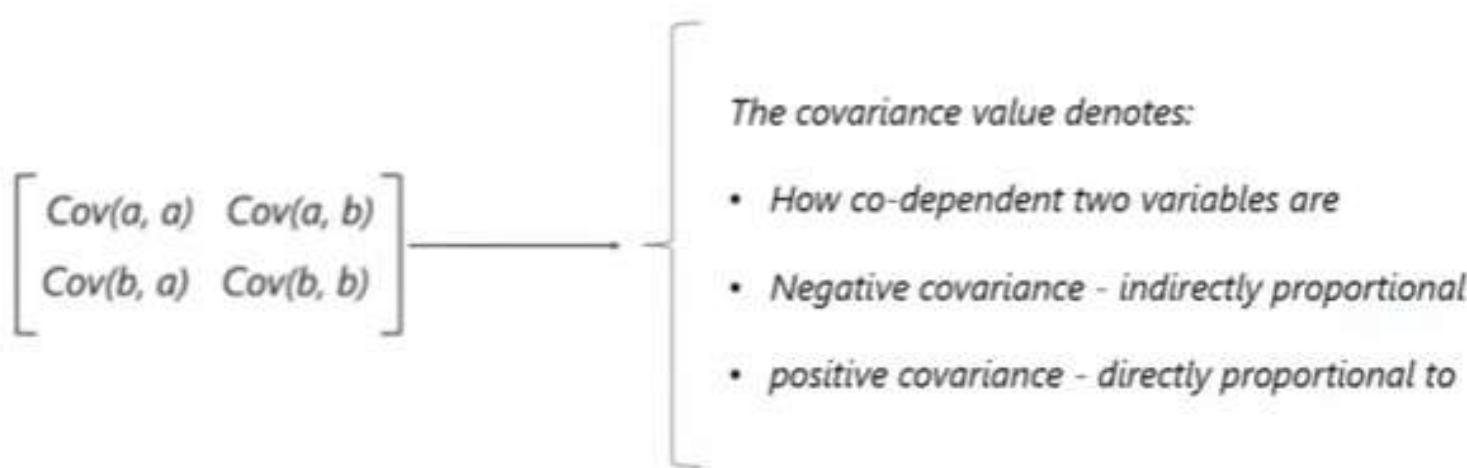
of downloads ranges between 100-5000

$$x_j^{(i)} = \frac{x_j^{(i)} - \bar{x}_j}{\sigma_j} \quad \forall j$$

Step2: Computing Covariance Matrix

- Expresses the correlation between the different variables in the data set.
- Essential to identify **heavily dependent variables** because they contain **biased and redundant information** which reduces the **overall performance** of the model

$$\Sigma = \frac{1}{m} \sum_i^m (x_{(i)})(x_{(i)})^T, \quad \Sigma \in \mathbb{R}^{n \times n}$$



Step3: Calculating the EigenVectors and EigenValues

- Eigenvectors and Eigenvalues are the mathematical constructs that must be computed **from the covariance matrix** in order to determine the principal components of the data set.
- principal components are the new set of variables that are obtained from the initial set of variables. They compress and possess most of the useful information that was scattered among the initial variables

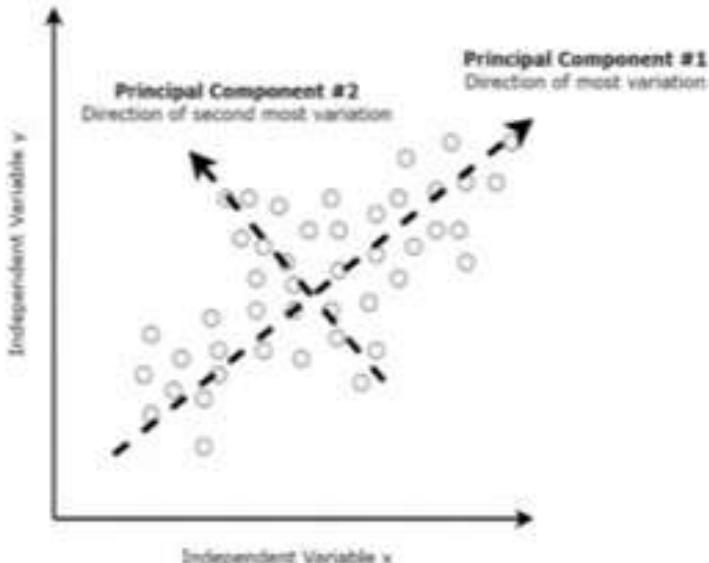
$$u^T \Sigma = \lambda u \quad (3)$$

$$U = \begin{pmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{pmatrix}, \quad u_i \in \mathbb{R}^n \quad (4)$$

- *Eigenvectors are those vectors when a linear transformation is performed on them then their direction does not change.*
- *Eigenvalues simply denote the scalars of the respective eigenvectors*

Step4:Compute Principal Components

- PC1 is the most significant and stores the maximum possible information.
- PC2 is the second most significant PC and stores remaining maximum info and so on



$$x_{\text{new}}^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k \quad (5)$$

- Once we have computed the Eigenvectors and Eigenvalues, all we have to do is order them in the descending order, where the eigenvector with the highest eigenvalue is the most significant and thus forms the first principal component.

- Step5: Reducing the dimensions of the Dataset

movieId	title	genres	userId	movieId	rating	timestamp
1	Toy Story (1995)	Animation Children	1	1	5	8.47E+08
2	Jumanji (1995)	Adventure Children Fantasy	1	2	3	8.48E+08
3	Grumpier Old Men (2004)	Comedy Romance	1	30	3	8.48E+08
4	Waiting to Exhale (1995)	Drama Romance	1	32	4	8.48E+08
5	Father of the Bride (1995)	Comedy	1	34	4	8.48E+08
6	Heat (1995)	Action Crime Thriller	1	47	3	8.48E+08
7	Sabrina (1995)	Comedy Romance	1	50	4	8.48E+08
8	Tom and Huck (1995)	Adventure Children	1	62	4	8.48E+08
9	Sudden Death (1995)	Action	1	150	4	8.47E+08
10	GoldenEye (1995)	Action Adventure Thriller	1	153	3	8.47E+08
11	American Pie (1999)	Comedy Drama Romance	1	160	3	8.48E+08
12	Dracula (1992)	Comedy Horror	1	161	4	8.48E+08
13	Bathtime (1999)	Adventure Animation Children	1	165	4	8.47E+08
14	Nixon (1995)	Drama	1	185	3	8.48E+08
15	Cutthroat Island (1995)	Action Adventure Romance	1	208	3	8.48E+08
16	Casino (1995)	Crime Drama	1	253	3	8.48E+08
17	Sense and Sensibility (1995)	Romance	1	265	3	8.48E+08
18	Four Room (2002)	Comedy				

Original data

userId	movieId	rating	timestamp
1	1	5	8.47E+08
1	2	3	8.48E+08
1	30	3	8.48E+08
1	32	4	8.48E+08
1	34	4	8.48E+08
1	47	3	8.48E+08
1	50	4	8.48E+08
1	62	4	8.48E+08
1	150	4	8.47E+08
1	153	3	8.47E+08
1	160	3	8.48E+08
1	161	4	8.48E+08
1	165	4	8.47E+08
1	185	3	8.48E+08
1	208	3	8.48E+08
1	253	3	8.48E+08
1	265	3	8.48E+08

Reduced data

$$x_{new}^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k \quad (5)$$

- To re-arrange the original data with the final principal components which represent the maximum and the most significant information of the data set.

- Step5: Reducing the dimensions of the Dataset

movieId	title	genres	userId	movieId	rating	timestamp
1	Toy Story (1995)	Animation Children	1	1	5	8.47E+08
2	Jumanji (1995)	Adventure Children Fantasy	1	2	3	8.48E+08
3	Grumpier Old Men (2004)	Comedy Romance	1	30	3	8.48E+08
4	Waiting to Exhale (1995)	Drama Romance	1	32	4	8.48E+08
5	Father of the Bride (1995)	Comedy	1	34	4	8.48E+08
6	Heat (1995)	Action Crime Thriller	1	47	3	8.48E+08
7	Sabrina (1995)	Comedy Romance	1	50	4	8.48E+08
8	Tom and Huck (1995)	Adventure Children	1	62	4	8.48E+08
9	Sudden Death (1995)	Action	1	150	4	8.47E+08
10	GoldenEye (1995)	Action Adventure Thriller	1	153	3	8.47E+08
11	American Pie (1999)	Comedy Drama Romance	1	160	3	8.48E+08
12	Dracula (1992)	Comedy Horror	1	161	4	8.48E+08
13	Bathtime (1999)	Adventure Animation Children	1	165	4	8.47E+08
14	Nixon (1995)	Drama	1	185	3	8.48E+08
15	Cutthroat Island (1985)	Action Adventure Romance	1	208	3	8.48E+08
16	Casino (1995)	Crime Drama	1	253	3	8.48E+08
17	Sense and Sensibility (1995)	Romance	1	265	3	8.48E+08
18	Four Room (2002)	Comedy				

Original data

userId	movieId	rating	timestamp
1	1	5	8.47E+08
1	2	3	8.48E+08
1	30	3	8.48E+08
1	32	4	8.48E+08
1	34	4	8.48E+08
1	47	3	8.48E+08
1	50	4	8.48E+08
1	62	4	8.48E+08
1	150	4	8.47E+08
1	153	3	8.47E+08
1	160	3	8.48E+08
1	161	4	8.48E+08
1	165	4	8.47E+08
1	185	3	8.48E+08
1	208	3	8.48E+08
1	253	3	8.48E+08
1	265	3	8.48E+08

Reduced data

$$x_{new}^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k \quad (5)$$

- To re-arrange the original data with the final principal components which represent the maximum and the most significant information of the data set.

PCA : Why do Eigen vectors have the highest co-variance

$$x_{new}^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k$$

This is the projection of the raw data data

- (5) The length (variance) of the projection of x onto u is given by $x^T u$. i.e., if $x^{(i)}$ is a point in the dataset, then its projection onto u is distance $x^T u$ from the origin. Hence to maximize the variance, we can formulate it as an optimization problem:

$$\max \quad \frac{1}{m} \sum_{i=1}^m (x^{(i)T} u)^2 \quad (6)$$

$$s.t. \quad \|u\| = 1 \quad (7)$$

PCA : Why do Eigen vectors have the highest co-variance

$$x_{new}^{(i)} = \begin{bmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{bmatrix} \in \mathbb{R}^k \quad (5)$$

This is the projection of the raw data data

The length (variance) of the projection of x onto u is given by $x^T u$. i.e., if $x^{(i)}$ is a point in the dataset, then its projection onto u is distance $x^T u$ from the origin. Hence to maximize the variance, we can formulate it as an optimization problem:

$$\max \quad \frac{1}{m} \sum_{i=1}^m (x^{(i)T} u)^2 \quad (6)$$

$$s.t. \quad \|u\| = 1 \quad (7)$$

PCA : Why do Eigen vectors have the highest co-variance

$$\max \quad \frac{1}{m} \sum_{i=1}^m (x^{(i)T} u)^2 \quad (6)$$

$$s.t. \quad \|u\| = 1 \quad (7)$$

This is a maximization problem with constraint. It can be reformulated as:

$$\max \quad u^T \left(\frac{1}{m} \sum_{i=1}^m x^{(i)} x^{(i)T} \right) u = u^T \Sigma u \quad (8)$$

$$u^T u = 1 \quad (9)$$

To get the (local) optimum of the constrained problem, we use the method of Lagrange multipliers:

$$L(u, \lambda) = u^T \Sigma u - \lambda(u^T u - 1) \quad (10)$$

PCA : Why do Eigen vectors have the highest co-variance

To get the (local) optimum of the constrained problem, we use the method of Lagrange multipliers:

$$L(u, \lambda) = u^T \Sigma u - \lambda(u^T u - 1) \quad (10)$$

Note that λ is the Lagrange multipliers not the eigenvalue of covariance matrix. Now let us take the partial derivatives of formula (10) with respect to u and λ . Then let the derivatives equal to zero:

$$\frac{\partial L}{\partial u} = u^T \Sigma - \lambda u = 0 \quad (11)$$

$$\frac{\partial L}{\partial \lambda} = u^T u - 1 = 0 \quad (12)$$

We can see that the formula (11) is equivalent to formula (3). Thus choosing the new basis of lower space is equivalent to getting the top-k eigenvector of covariance matrix of raw data.

$$u^T \Sigma = \lambda u \quad (3)$$

$$U = \begin{pmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{pmatrix}, \quad u_i \in \mathbb{R}^n \quad (4)$$

Thus choosing the new basis of lower space is equivalent to getting the top-k eigenvector of covariance matrix of raw data.

Step-01a: Get data.

Step-01b: Compute the mean vector (μ).

Step-01c: Subtract mean from the given data.

Step-02: Calculate the covariance matrix.

Step-03: Calculate the eigen vectors and eigen values of the covariance matrix.

Step-04: Choosing components and forming a feature vector.

Step-05: Deriving the new data set.

Step 01a: Consider the two dimensional patterns (2, 1), (3, 5),
(4, 3), (5, 6), (6, 7), (7, 8).

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Step 01b: Calculate the mean vector (μ)

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Mean vector (μ)

$$= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)$$
$$= (4.5, 5)$$

Mean vector (μ) = $\begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$

Step 01c: Subtract mean vector (μ) from the given feature vectors

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

- $x_1 - \mu = (2 - 4.5, 1 - 5) = (-2.5, -4)$
- $x_2 - \mu = (3 - 4.5, 5 - 5) = (-1.5, 0)$
- $x_3 - \mu = (4 - 4.5, 3 - 5) = (-0.5, -2)$
- $x_4 - \mu = (5 - 4.5, 6 - 5) = (0.5, 1)$
- $x_5 - \mu = (6 - 4.5, 7 - 5) = (1.5, 2)$
- $x_6 - \mu = (7 - 4.5, 8 - 5) = (2.5, 3)$

Mean vector (μ) = $\begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$

Feature vectors (x_i) after subtracting mean vector (μ)

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Step 02: Calculate the covariance matrix

$$\text{Covariance Matrix} = \frac{\Sigma (x_i - \mu)(x_i - \mu)^t}{n}$$

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Step 02: Calculate the covariance matrix

$$m_1 = (x_1 - \mu)(x_1 - \mu)^t = \begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -2.5 & -4 \end{bmatrix} = \begin{bmatrix} 6.25 & 10 \\ 10 & 16 \end{bmatrix}$$

$$m_2 = (x_2 - \mu)(x_2 - \mu)^t = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -1.5 & 0 \end{bmatrix} = \begin{bmatrix} 2.25 & 0 \\ 0 & 0 \end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)^t = \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} -0.5 & -2 \end{bmatrix} = \begin{bmatrix} 0.25 & 1 \\ 1 & 4 \end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)^t = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)^t = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 1.5 & 2 \end{bmatrix} = \begin{bmatrix} 2.25 & 3 \\ 3 & 4 \end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)^t = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \begin{bmatrix} 2.5 & 3 \end{bmatrix} = \begin{bmatrix} 6.25 & 7.5 \\ 7.5 & 9 \end{bmatrix}$$

Covariance Matrix = $\frac{\sum (x_i - \mu)(x_i - \mu)^t}{n}$

Covariance matrix
 $= (m_1 + m_2 + m_3 + m_4 + m_5 + m_6) / 6$

On adding the above matrices
 and dividing by 6, we get-

Covariance Matrix = $\frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$

Covariance Matrix = $\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$

Step 03: Calculate the eigen values and eigen vectors of the covariance matrix

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

Covariance Matrix = $\frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$

Covariance Matrix = $\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

From here,

$$(2.92 - \lambda)(5.67 - \lambda) - (3.67 \times 3.67) = 0$$

$$16.56 - 2.92\lambda - 5.67\lambda + \lambda^2 - 13.47 = 0$$

$$\lambda^2 - 8.59\lambda + 3.09 = 0$$

Solving this quadratic equation,
we get $\lambda = 8.22, 0.38$
Thus, two eigen values are
 $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

Step 03: Calculate the eigen values and eigen vectors of the covariance matrix

- Eigen vector corresponding to the greatest eigen value is the principal component (PC1) for the given data set.
- So, we find the eigen vector corresponding to eigen value λ_1 .
- We use the following equation to find the eigen vector-
 - $MX = \lambda X$
- where

- M = Covariance Matrix

- X = Eigen vector

- λ = Eigen value

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = 8.22 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

- Substituting the values in the above equation, we get-

Step 03: Calculate the eigen values and eigen vectors of the covariance matrix

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = 8.22 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

Solving these, we get-

$$2.92X_1 + 3.67X_2 = 8.22X_1$$

$$3.67X_1 + 5.67X_2 = 8.22X_2$$

On simplification, we get-

$$5.3X_1 = 3.67X_2 \quad \dots\dots\dots(1)$$

$$3.67X_1 = 2.55X_2 \quad \dots\dots\dots(2)$$

From (1) and (2), $X_1 = 0.69X_2$

From (2), the eigen vector is-

Eigen Vector :

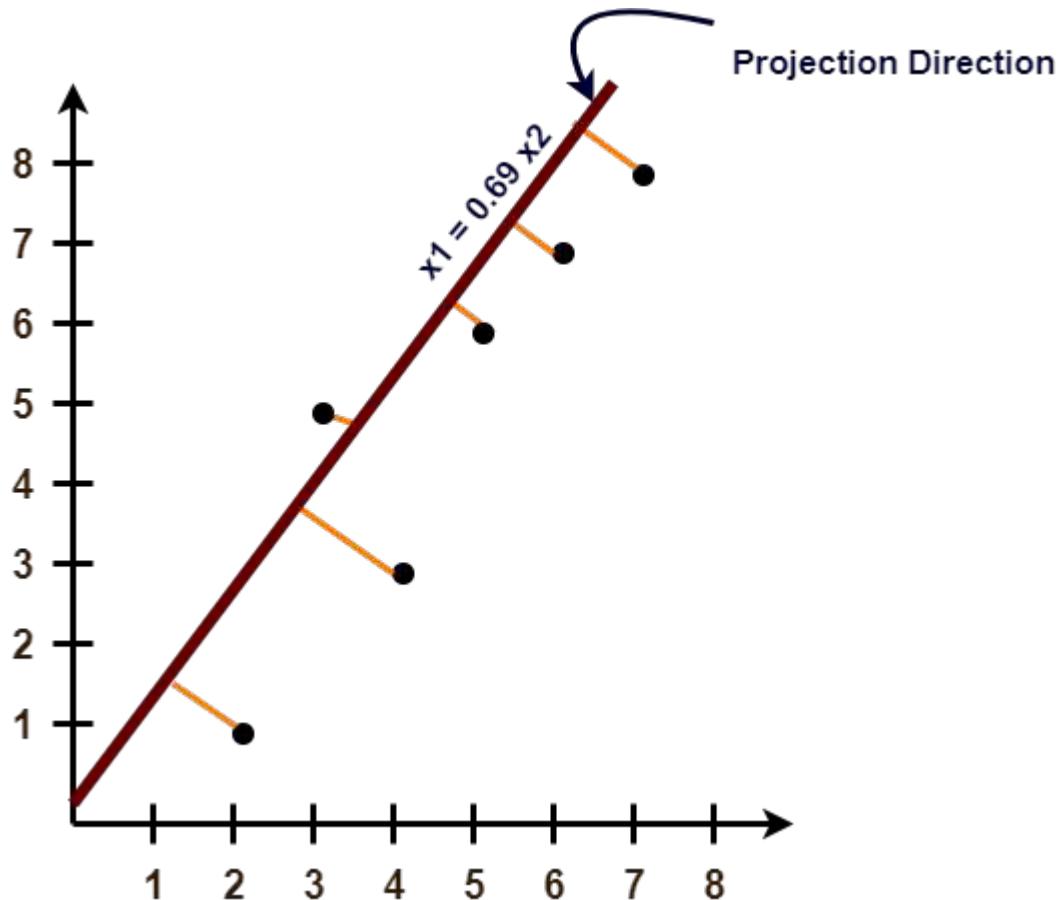
$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Step 03: Calculate the eigen values and eigen vectors of the covariance matrix

Principal Component :

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Step 04: Projecting the data points onto the new subspace



Use PCA Algorithm to transform the pattern (2, 1) onto the eigen vector in the previous Example.

Given Feature Vector :

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

The feature vector gets transformed to

= Transpose of Eigen vector \times (Feature Vector – Mean Vector)

$$= \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}^T \times \left(\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 4.5 \\ 5 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 2.55 & 3.67 \end{bmatrix} \times \begin{bmatrix} -2.5 \\ -4 \end{bmatrix}$$

$$= -21.055$$

Try Using the PCA() class from the matplotlib.mlab library

1. Breast Cancer dataset

The Breast Cancer data set is a real-valued multivariate data that consists of two classes, where each class signifies whether a patient has breast cancer or not. The two categories are: malignant and benign.

The malignant class has 212 samples, whereas the benign class has 357 samples.

It has 30 features shared across all classes: radius, texture, perimeter, area, smoothness, fractal dimension, etc.

You can download the breast cancer dataset from kdnuggets.com, or rather an easy way is by loading it with the help of the sklearn library.

2. CIFAR - 10

The CIFAR-10 (Canadian Institute For Advanced Research) dataset consists of 60000 images each of 32x32x3 color images having ten classes, with 6000 images per category.

The dataset consists of 50000 training images and 10000 test images.

The classes in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

You can download the CIFAR dataset from kdnuggets.com, or you can also load it on the fly with the help of a deep learning library like Keras.

MACHINE INTELLIGENCE

Resources

- [http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine Learning in Action.pdf](http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine%20Learning%20in%20Action.pdf)
- <http://wwwusers.cs.umn.edu/~kumar/dmbook/>.
- <ftp://ftp.aw.com/cseng/authors/tan>
- <http://web.ccsu.edu/datamining/resources.html>



THANK YOU

Srinivas KS

Department of Computer Science and Engineering

srinivasks@pes.edu



MACHINE INTELLIGENCE

Dr. N MEHALA

Department of Computer Science
and Engineering

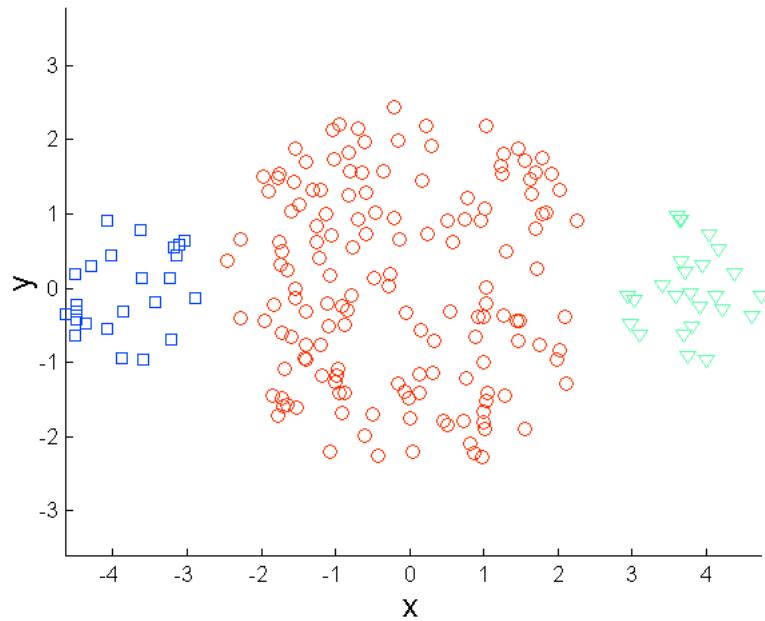
MACHINE INTELLIGENCE

Module 4 [Unsupervised Learning]

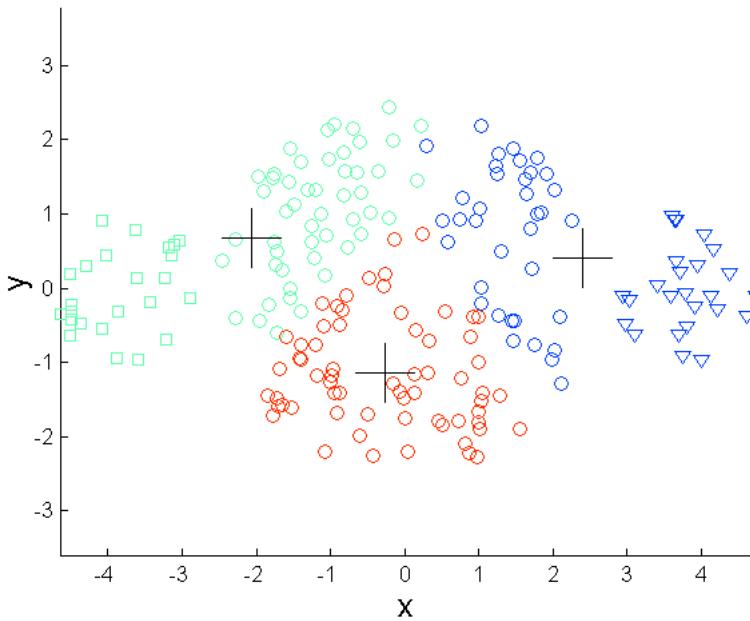
Dr. N MEHALA

Department of Computer Science and Engineering

Limitations of K-means: Differing Sizes



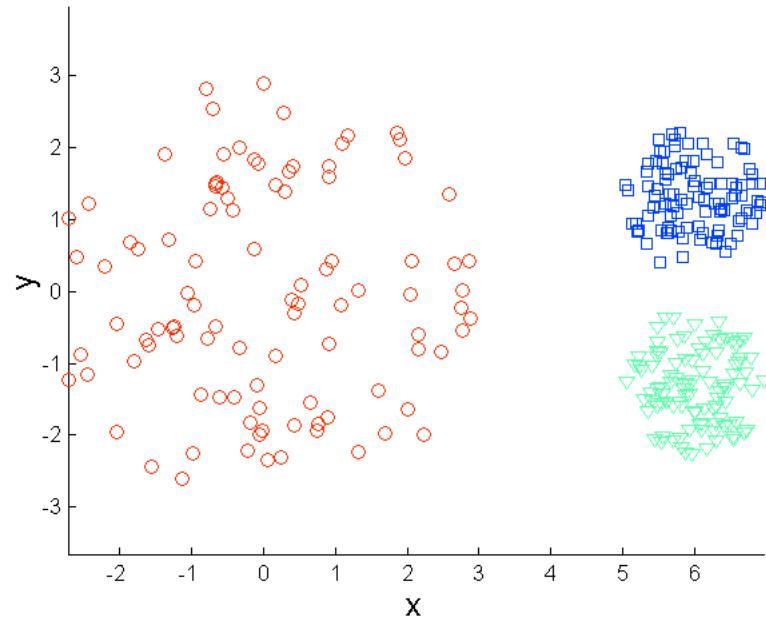
Original Points



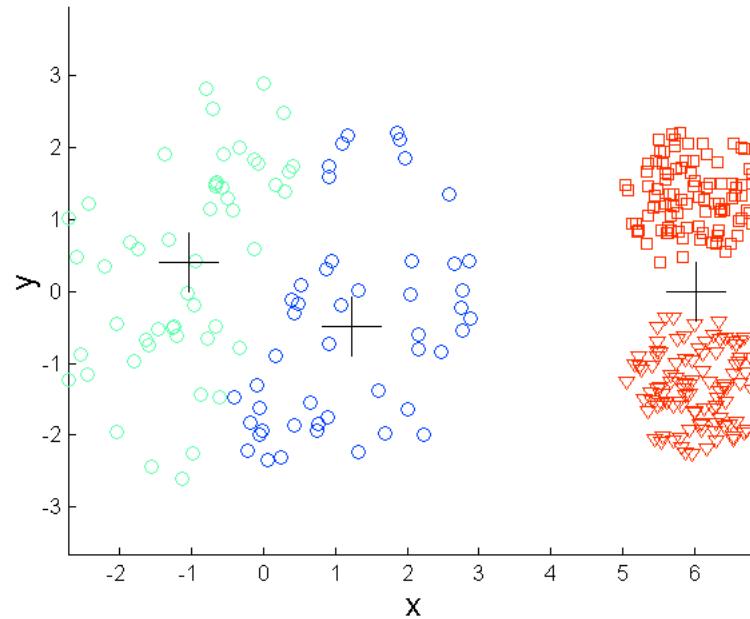
K-means (3 Clusters)

K-Means Clustering: Limitations

Limitations of K-means: Differing Density

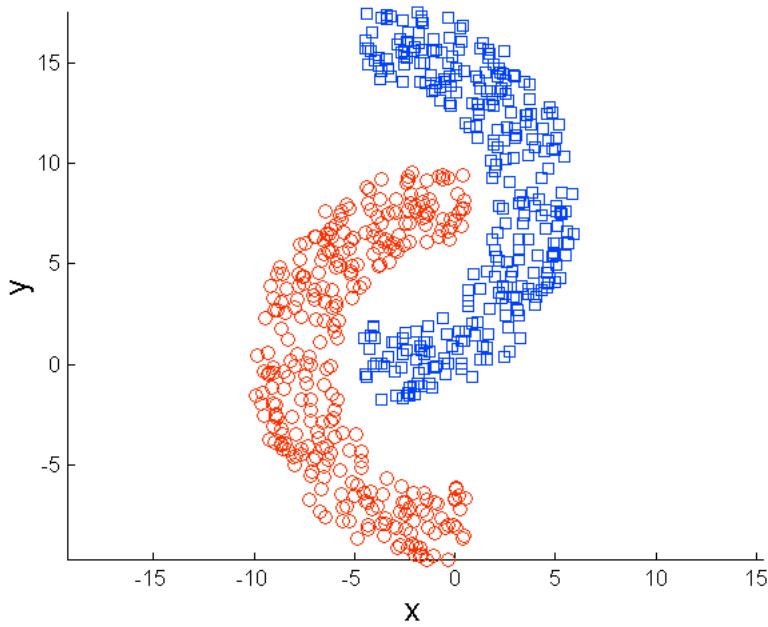


Original Points

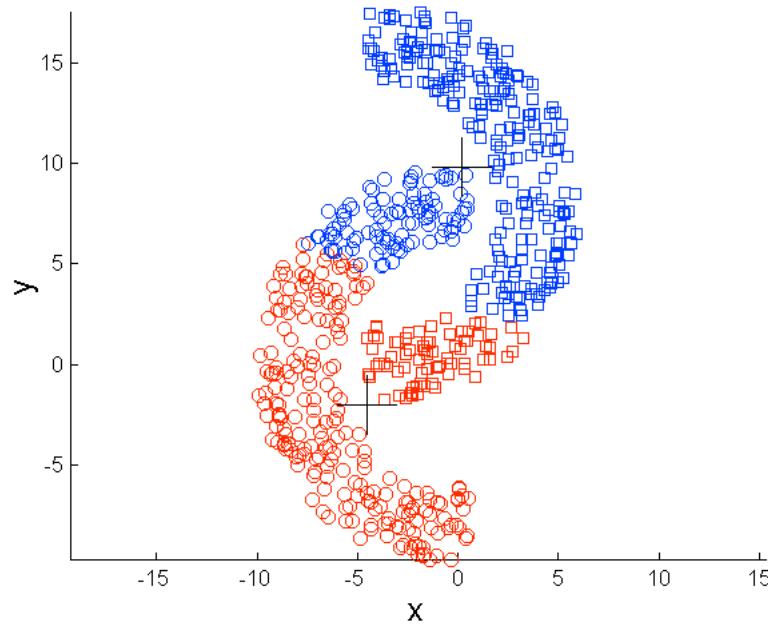


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

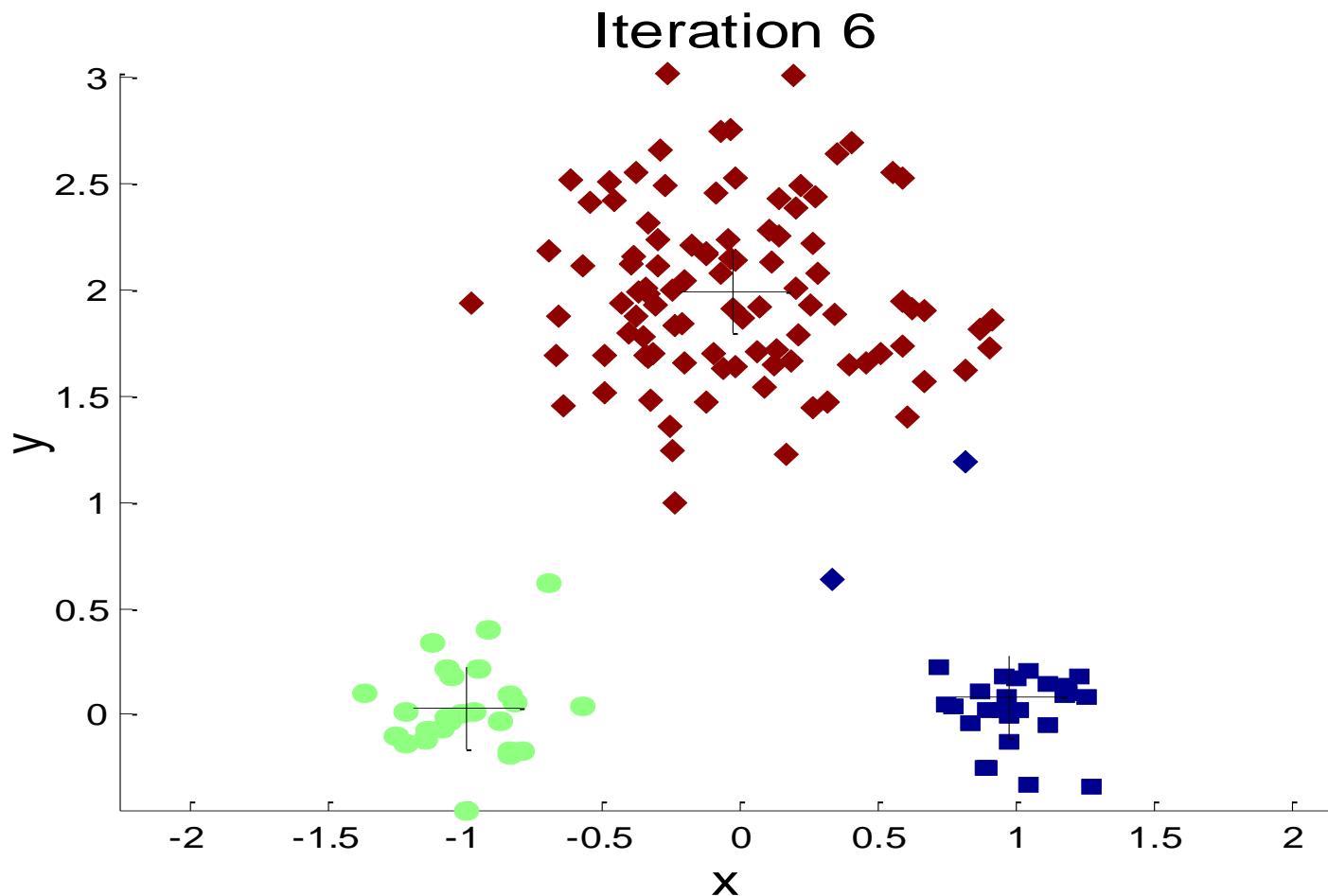


Original Points

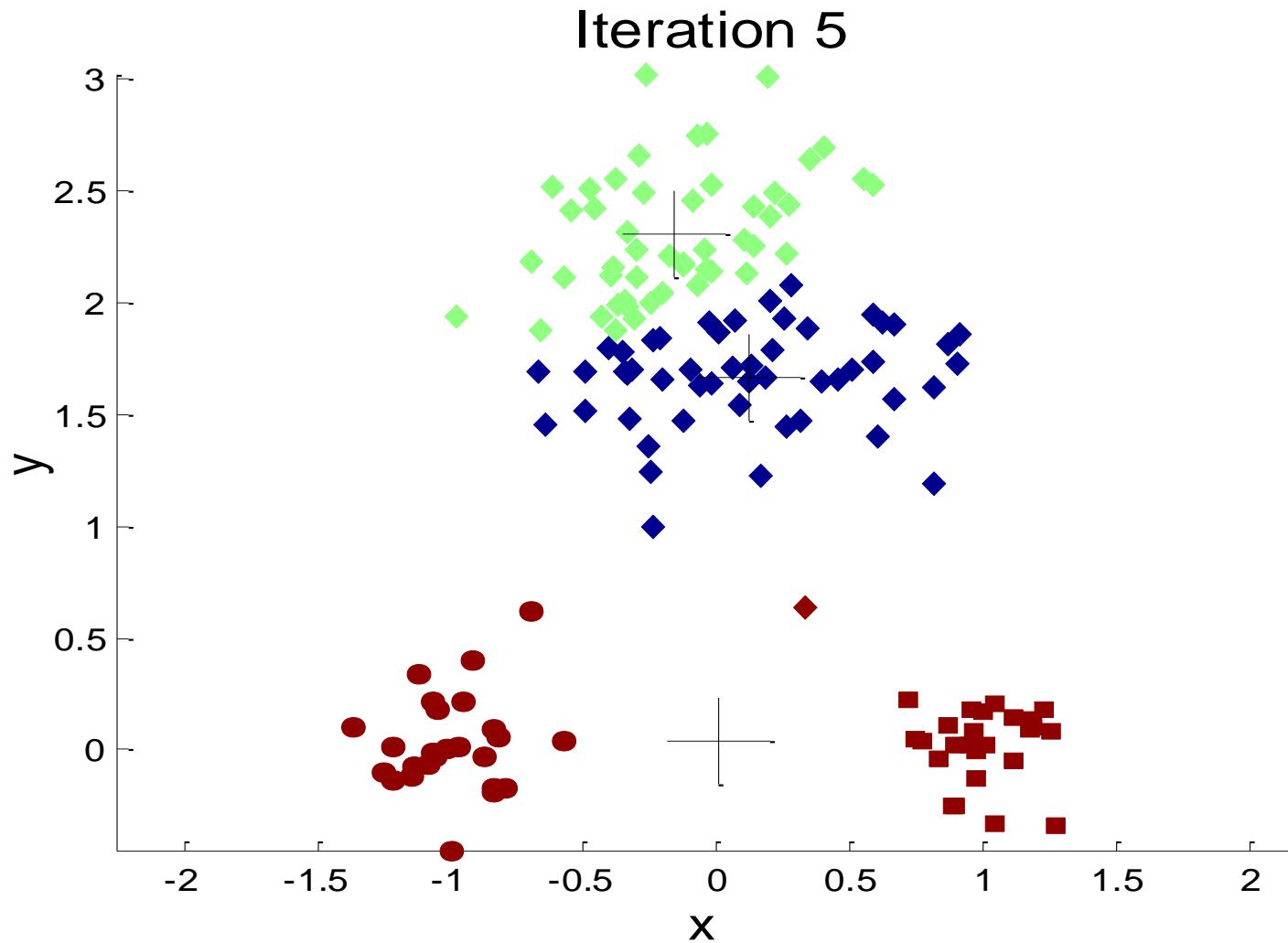


K-means (2 Clusters)

Importance of Choosing Initial Centroids



Importance of Choosing Initial Centroids



Problems with Selecting Initial Points

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
 - Chance is relatively small when K is large
 - If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
- Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't

Solutions to Initial Centroids Problem

- Multiple runs
 - Helps, but probability is not on our side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
 - Select most widely separated
- Postprocessing
- Generate a larger number of clusters and then perform a hierarchical clustering
- Bisecting K-means
 - Not as susceptible to initialization issues

Bisection K-Means Clustering Algorithm

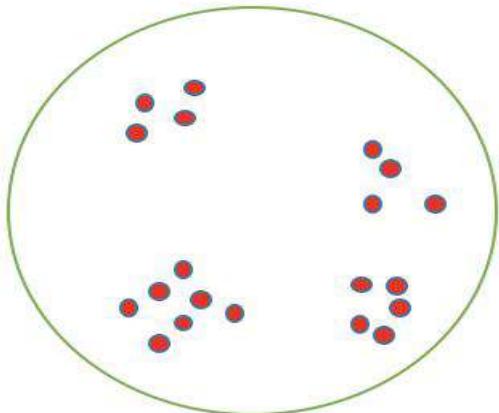
- To overcome the problem of poor clusters because of k-means getting caught in a local minimum
- Variant of K-means that can produce a partitional or a hierarchical clustering
- Instead of partitioning the data set into K clusters in each iteration, bisection k-means algorithm splits one cluster into two sub clusters at each bisecting step (by using k-means) until k clusters are obtained.
- Hybrid approach between Divisive Hierarchical Clustering (top down clustering) and K-means Clustering.

Note: A local minimum means that the result is good but not necessarily the best possible. A global minimum is the best possible

How it Works?

Step1: Set K to define the number of cluster

Step2: Set all data as a single cluster

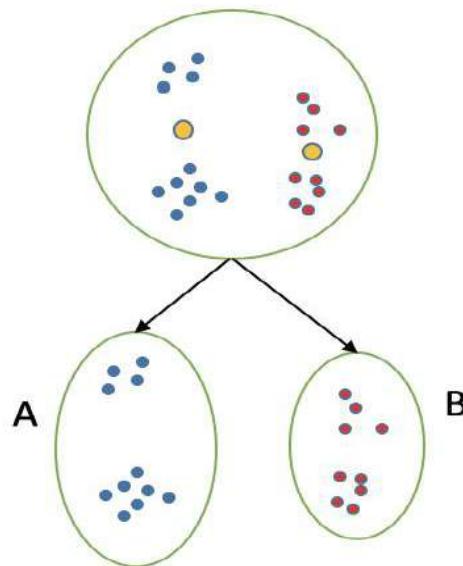


How it Works?

Step1: Set K to define the number of cluster

Step2: Set all data as a single cluster

Step3: Use K-means with K=2 to split the cluster



How it Works?

Step1: Set K to define the number of cluster

Step2: Set all data as a single cluster

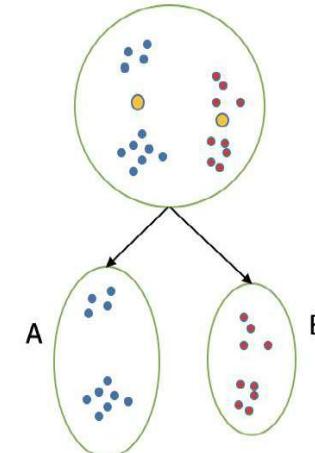
Step3: Use K-means with K=2 to split the cluster

Step4: Measure the distance for each intra cluster

- Sum of square Distance

$$\sum_{i=0}^n (X_i - \bar{X})^2$$

Step5: Select the cluster that have the largest distance and split to 2 cluster using K-means



How it Works?

Step1: Set K to define the number of cluster

Step2: Set all data as a single cluster

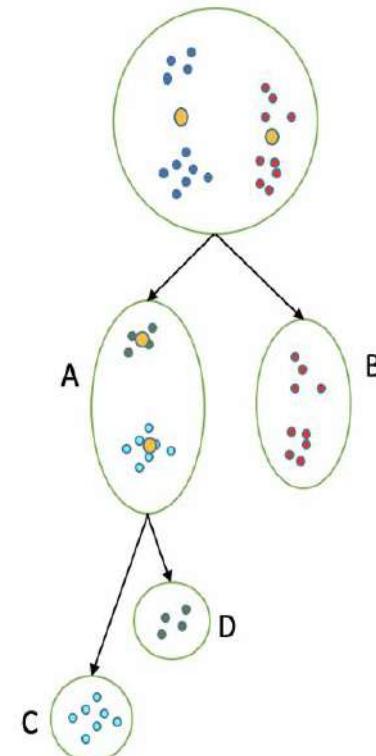
Step3: Use K-means with K=2 to split the cluster

Step4: Measure the distance for each intra cluster

- Sum of square Distance

$$\sum_{i=0}^n (X_i - \bar{X})^2$$

Step5: Select the cluster that have the largest distance and split to 2 cluster using K-means



How it Works?

Step1: Set K to define the number of cluster

Step2: Set all data as a single cluster

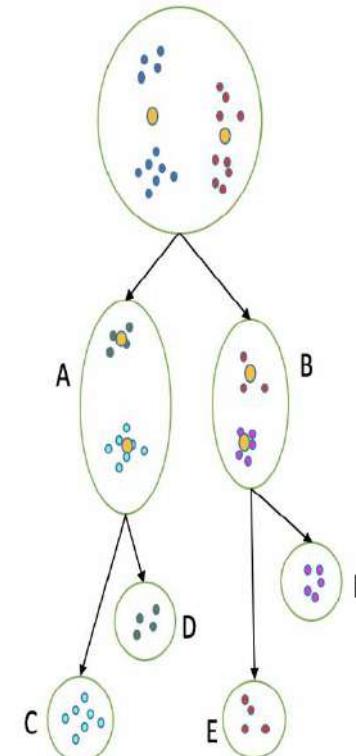
Step3: Use K-means with K=2 to split the cluster

Step4: Measure the distance for each intra cluster

- Sum of square Distance

Step5: Select the cluster that have the largest distance and split to 2 cluster using K-means

Step6: Repeat step 3–5 until the number of leaf cluster = K.



Advantage of Bisecting K-Means over K-Means

- Bisecting k-means is more efficient when **K** is large.
- Bisecting k-means produce clusters of similar sizes, while k-means is known to produce clusters of widely different sizes.

- K-Means Issues
- Bisecting K-Means Clustering

MACHINE INTELLIGENCE

Resources

- [http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine Learning in Action.pdf](http://www2.ift.ulaval.ca/~chaib/IFT-4102-7025/public_html/Fichiers/Machine%20Learning%20in%20Action.pdf)
- <http://wwwusers.cs.umn.edu/~kumar/dmbook/.>
- <ftp://ftp.aw.com/cseng/authors/tan>
- <http://web.ccsu.edu/datamining/resources.html>



THANK YOU

Dr. N MEHALA

Department of Computer Science and Engineering

mehala@pes.edu



MACHINE INTELLIGENCE

Srinivas K S

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

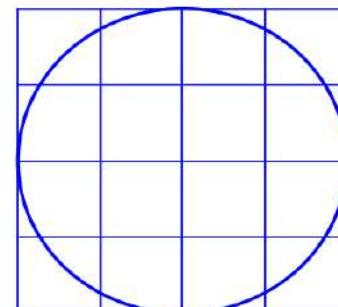
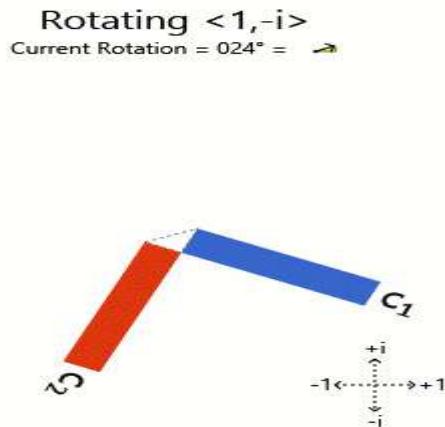
SVD

Srinivas K S

Department of Computer Science and Engineering

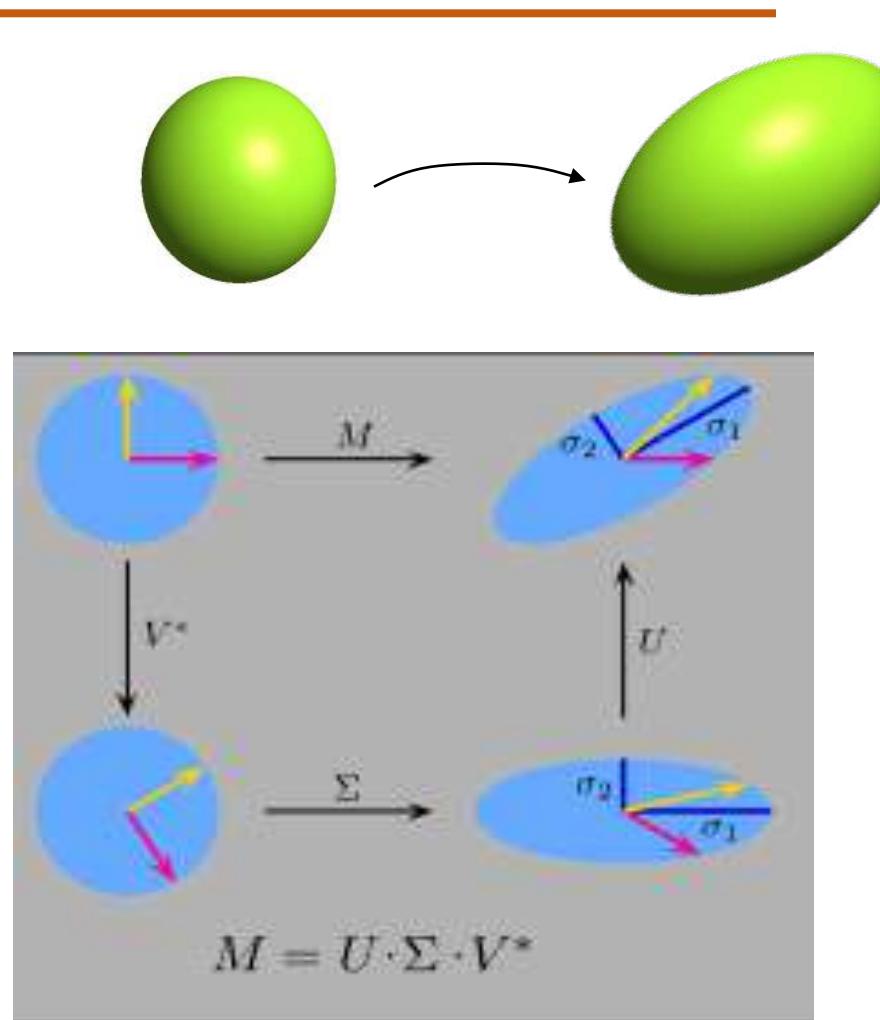
What happens when you multiply matrix with a vector

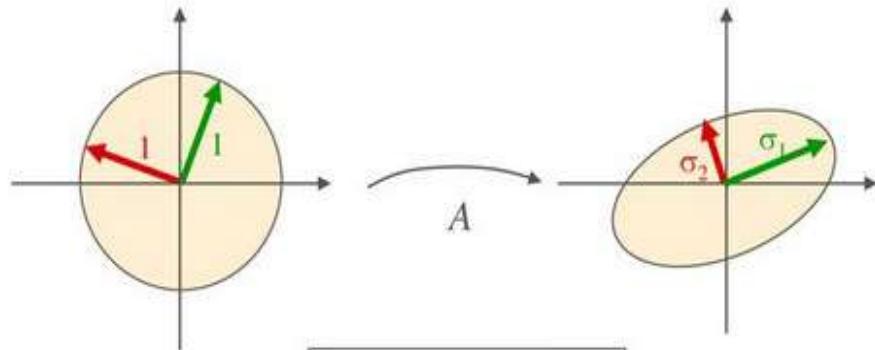
- The Vector Scales
- The Vector Rotates
- The Vector Scales and Rotates



Singular Value Decomposition : SVD

- Matrix multiplied with vector results in scaling and rotation
- For a set of vectors that form a circle – the same thing happens
- That's the sum and substance of singular value decomposition
- Find 2 guys – matrices that rotate and scale.





$$AV = U \Sigma$$

$$A [\text{orthonormal } \mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_n] = [\text{orthonormal } \mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_n] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

A sphere of N-Dimensions

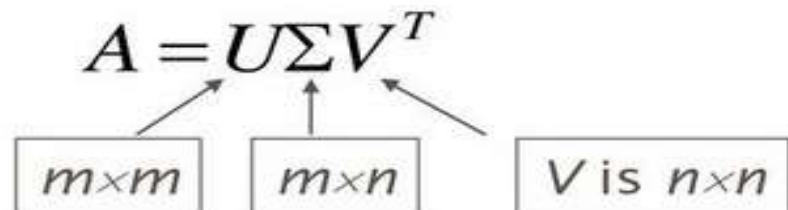
- By multiplying it with a matrix A
- New coordinate system U is called principal axis
- Stretched by a scaling factor α are called singular values

For any matrix $m \times n$

For an $m \times n$ matrix \mathbf{A} of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$

$m \times m$ $m \times n$ $V \text{ is } n \times n$



The columns of \mathbf{U} are orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^T$.

The columns of \mathbf{V} are orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$.

Eigenvalues $\lambda_1 \dots \lambda_r$ of $\mathbf{A}\mathbf{A}^T$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = \text{diag}(\sigma_1 \dots \sigma_r)$$

Singular values.

NOTE:

- The matrix \mathbf{AA}^T and $\mathbf{A}^T\mathbf{A}$ are very special in linear algebra.
- Consider any $m \times n$ matrix \mathbf{A} , we can multiply it with \mathbf{A}^T to form \mathbf{AA}^T and $\mathbf{A}^T\mathbf{A}$ separately.
 - These matrices are
 - symmetrical,
 - square,
 - both matrices have the same positive eigenvalues, and
 - both have the same rank r as \mathbf{A} .

Recollect

$$A \text{ is symmetric} \iff A = A^T.$$

- **AA^T and A^TA are symmetric**
- we can choose its eigenvectors to be orthonormal (perpendicular to each other with unit length)

$\overline{AA^T}$ and $\overline{A^TA}$

A does not need to be square

Eigenvectors can be chosen to be orthonormal

- We name the eigenvectors for AA^T as u_i and A^TA as v_i
- call these sets of eigenvectors u and v the **singular vectors** of A .
- Both matrices have the same positive eigenvalues.
- The square roots of these eigenvalues are called **singular values**.

SO SVD IS ALL ABOUT FINDING 3 MATRICES

$$A_{m \times n} = U_{m \times m} S_{m \times n} V^T_{n \times n}$$

$$\begin{pmatrix} & \mathbf{A} & \\ x_{11} & x_{12} & x_{1n} \\ & \ddots & \\ x_{m1} & & x_{mn} \end{pmatrix}_{m \times n} = \begin{pmatrix} & \mathbf{U} & \\ u_{11} & \dots & u_{m1} \\ u_{1m} & \ddots & u_{mm} \end{pmatrix}_{m \times m} \begin{pmatrix} & \mathbf{S} & \\ \sigma_1 & \dots & 0 \\ 0 & \ddots & 0 \end{pmatrix}_{m \times n} \begin{pmatrix} & \mathbf{V}^T & \\ v_{11} & \dots & v_{1n} \\ v_{n1} & & v_{nn} \end{pmatrix}_{n \times n}$$

The columns of \mathbf{U} are orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^T$.

The columns of \mathbf{V} are orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$.

Eigenvalues $\lambda_1, \dots, \lambda_r$ of $\mathbf{A}\mathbf{A}^T$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$$

 Singular values.

ONE MORE TIME

 A is $m \times n$ $A = (\text{orthogonal}) (\text{diagonal}) (\text{orthogonal})$

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

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} = \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} U \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \sigma_r \end{bmatrix} \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} V^T$$

8)

$$A = U \Sigma V^T \rightarrow A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

- Inner Product
- 2 Vectors are orthogonal if their inner product equals 0 i.e the angle between them is 90

$$(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_{i=1}^n x_i y_i$$

For example, if $\vec{x} = [1, 6, 7, 4]$ and $\vec{y} = [3, 2, 8, 3]$, then

$$\vec{x} \cdot \vec{y} = 1(3) + 6(2) + 7(8) + 3(4) = 83$$

the vectors $[2, 1, -2, 4]$ and $[3, -6, 4, 2]$

are orthogonal because

$$[2, 1, -2, 4] \cdot [3, -6, 4, 2] = 2(3) + 1(-6) - 2(4) + 4(2) = 0$$

NORMAL AND ORTHONORMAL VECTORS

if $\vec{v} = [2, 4, 1, 2]$, then

$$|\vec{v}| = \sqrt{2^2 + 4^2 + 1^2 + 2^2} = \sqrt{25} = 5$$

Then $\vec{u} = [2/5, 4/5, 1/5, 1/5]$ is a normal vector because

$$|\vec{u}| = \sqrt{(2/5)^2 + (4/5)^2 + (1/5)^2 + (2/5)^2} = \sqrt{25/25} = 1$$

.....,

$$\vec{u} = [2/5, 1/5, -2/5, 4/5]$$

and

$$\vec{v} = [3/\sqrt{65}, -6/\sqrt{65}, 4/\sqrt{65}, 2/\sqrt{65}]$$

are orthonormal because

$$|\vec{u}| = \sqrt{(2/5)^2 + (1/5)^2 + (-2/5)^2 + (4/5)^2} = 1$$

$$|\vec{v}| = \sqrt{(3/\sqrt{65})^2 + (-6/\sqrt{65})^2 + (4/\sqrt{65})^2 + (2/\sqrt{65})^2} = 1$$

$$\vec{u} \cdot \vec{v} = \frac{6}{5\sqrt{65}} - \frac{6}{5\sqrt{65}} - \frac{8}{5\sqrt{65}} + \frac{8}{5\sqrt{65}} = 0$$

- \vec{u} and \vec{v}
- Are orthonormal as lengths are 1 and dot product is 0

Gram-Schmidt Orthonormalization Process

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 0 \\ 2 & 3 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\vec{w}_k = \vec{v}_k - \sum_{i=1}^{k-1} \vec{u}_t \cdot \vec{v}_k * \vec{u}_t$$

first normalize $\vec{v}_1 = [1, 0, 2, 1]$:

$$\vec{u}_1 = [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}].$$

Next, let

$$\begin{aligned} \vec{w}_2 &= \vec{v}_2 - \vec{u}_1 \cdot \vec{v}_2 * \vec{u}_1 = [2, 2, 3, 1] - [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}] \cdot [2, 2, 3, 1] * [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}] \\ &= [2, 2, 3, 1] - (\frac{9}{\sqrt{6}}) * [\frac{1}{\sqrt{6}}, 0, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}] \\ &= [2, 2, 3, 1] - [\frac{3}{2}, 0, 3, \frac{3}{2}] \\ &= [\frac{1}{2}, 2, 0, \frac{-1}{2}] \end{aligned}$$

Normalize \vec{w}_2 to get

$$\vec{u}_2 = [\frac{\sqrt{2}}{6}, \frac{2\sqrt{2}}{3}, 0, \frac{-\sqrt{2}}{6}]$$

Singular Value Decomposition : SVD

A matrix A is orthogonal if $AA^T = A^TA = I$. For example,

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix}$$

is orthogonal because

$$A^TA = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Singular Value Decomposition 3 views

- Capturing the essence of moving from higher to lower dimensions
 - method for identifying and ordering the dimensions along which data points exhibit the most variation
 - method for transforming correlated variables into a set of uncorrelated ones that better expose the various relationships
- Let us look at a mxn matrix.
 - This can be expressed as a product of 3 matrices

$$A_{mn} = U_{mm} S_{mn} V_{nn}^T$$

Some properties of this decomposition

- Columns of U are orthonormal eigenvectors of AA^T
- Columns of V are orthonormal eigenvectors of A^TA
- S is a diagonal matrix containing the square roots of Eigen values from U or V in descending order
- Lets work thru an example

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

Lets compute U first

- A^T

$$A^T = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}$$

- AA^T

$$AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

- Eigen Values and Eigen Vector

$$\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$(11 - \lambda)x_1 + x_2 = 0$$

$$x_1 + (11 - \lambda)x_2 = 0$$

Solve for Eigen Value

$$\begin{vmatrix} (11 - \lambda) & 1 \\ 1 & (11 - \lambda) \end{vmatrix} = 0$$

$$(11 - \lambda)(11 - \lambda) - 1 \cdot 1 = 0$$

$$(\lambda - 10)(\lambda - 12) = 0$$

$$\lambda = 10, \lambda = 12$$

Plug this back

For $\lambda = 10$ we get

$$(11 - 10)x_1 + x_2 = 0$$

$$x_1 = -x_2$$

: $x_1 = 1$ and $x_2 = -1$ For $\lambda = 12$ we have

$$(11 - 12)x_1 + x_2 = 0$$

$$x_1 = x_2$$

$$x_1 = 1 \text{ and } x_2 = 1$$

Singular Value Decomposition : SVD

and for the same reason as before we'll take $x_1 = 1$ and $x_2 = 1$. Now, for $\lambda = 12$ we have the eigenvector $[1, 1]$. These eigenvectors become column vectors in a matrix ordered by the size of the corresponding eigenvalue. In other words, the eigenvector of the largest eigenvalue is column one, the eigenvector of the next largest eigenvalue is column two, and so forth and so on until we have the eigenvector of the smallest eigenvalue as the last column of our matrix. In the matrix below, the eigenvector for $\lambda = 12$ is column one, and the eigenvector for $\lambda = 10$ is column two.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

- Now Ortho normalize this by Gram Schmidt
- Similarly do this for V which is $A^T A$

V Calculations

$$A^T A = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$10x_1 + 2x_3 = \lambda x_1$$

$$10x_2 + 4x_3 = \lambda x_2$$

$$2x_1 + 4x_2 + 2x_3 = \lambda x_2$$

$$(10 - \lambda)x_1 + 2x_3 = 0$$

$$(10 - \lambda)x_2 + 4x_3 = 0$$

$$2x_1 + 4x_2 + (2 - \lambda)x_3 = 0$$

$$\begin{vmatrix} (10 - \lambda) & 0 & 2 \\ 0 & (10 - \lambda) & 4 \\ 2 & 4 & (2 - \lambda) \end{vmatrix} = 0$$

$$\lambda(\lambda - 10)(\lambda - 12) = 0$$

Eigen Values and Eigen Vectors

$$\lambda = 12, \vec{v}_1 = [1, 2, 1]. \quad \begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & 0 & 5 \end{bmatrix}$$

$$\lambda = 10, \vec{v}_2 = [2, -1, 0]. \quad V = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

$$V^T = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

Finally S

For S we take the square roots of the non-zero eigenvalues and populate the diagonal with them, putting the largest in s_{11} , the next largest in s_{22} and so on until the smallest value ends up in s_{mm} . The non-zero eigenvalues of U and V are always the same, so that's why it doesn't matter which one we take them from. Because we are doing full SVD, instead of reduced SVD (next section), we have to add a zero column vector to S so that it is of the proper dimensions to allow multiplication between U and V . The diagonal entries in S are the singular values of A , the columns in U are called left singular vectors, and the columns in V are called right singular vectors.

$$S = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

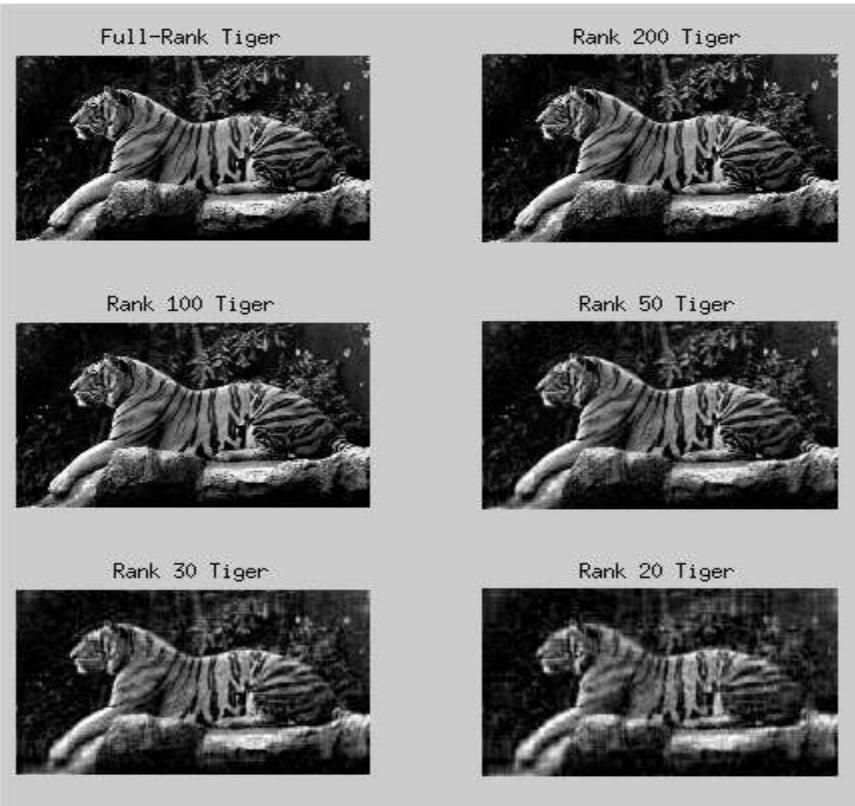
Applications

- Page Ranking
- Recommendation Systems
- Image Compression
- Facial Recognition
- Noise reduction

You My Dear Friends are all Tigers



Just 20% of your high dimensions is enough
to still show you as Tigers





THANK YOU

Srinivas K.S

Department of Computer Science and Engineering

srinivasks@pes.edu



MACHINE INTELLIGENCE EXPECTATION MAXIMIZATION

K.S.Srinivas

Department of Computer Science and Engineering



MACHINE INTELLIGENCE EXPECTATION MAXIMIZATION

K.S.Srinivas

Department of Computer Science
and Engineering

Machine Intelligence

Unit III

Expectation Maximization

Srinivas K.S
Department of Computer Science

Expectation Maximization

- Maximum likelihood estimation is an approach to **density estimation** for a dataset by **searching across probability distributions** and their **parameters**.
- It is a general and effective approach that underlies many machine learning algorithms, although it **requires that the training dataset is complete**, e.g. all relevant **interacting random variables are present**.
- Maximum likelihood becomes **intractable** if there are variables that interact with those in the dataset but were **hidden or not observed**, so-called latent variables.
- **The expectation-maximization algorithm** is an approach for performing **maximum likelihood estimation in the presence of latent variables**

It does this by f

estimating the values for the latent variables, (E)
then optimizing the model(M),

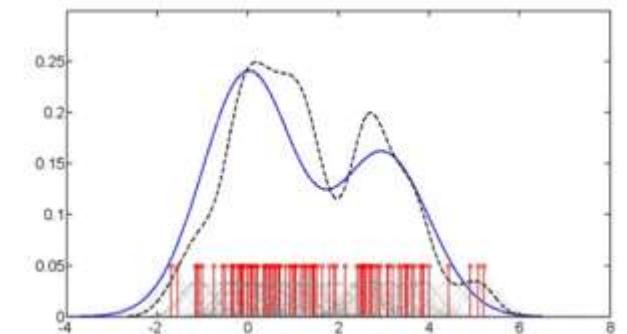
then repeating these two steps until convergence.

Unsupervised Learning and EM

- A central application of unsupervised learning is in the field of density estimation.

unsupervised learning intends to infer an a priori probability distribution $p_X(x)$.

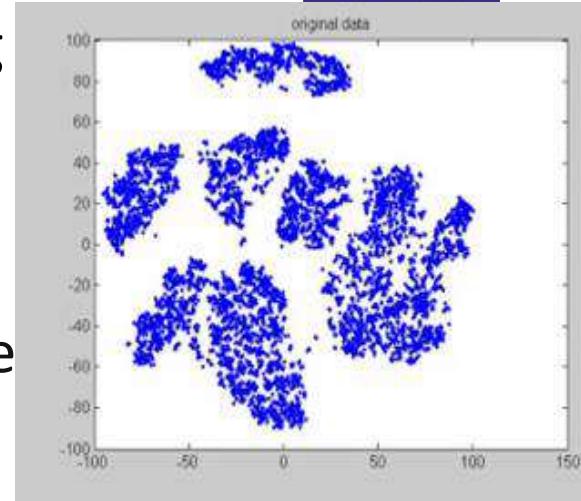
- We will cover unsupervised learning in Unit 4 some 8 hours from now – but lets understand one of the simplest unsupervised learning algorithm to set the context for expectation maximization
- K-Means clustering



- K-means clustering is a simple and elegant approach for partitioning a data set into K **distinct, non-overlapping clusters**.
- To perform K-means clustering, we must first specify the desired number of clusters K
- K-means algorithm will assign each observation to exactly one of the K clusters

Algorithm 10.1 K-Means Clustering

1. Randomly assign a number, from 1 to K , to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
 - (a) For each of the K clusters, compute the cluster *centroid*. The k th cluster centroid is the vector of the p feature means for the observations in the k th cluster.
 - (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).



Let C_1, \dots, C_K denote sets containing the indices of the observations in each cluster

They must satisfy 2 properties

1. $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_{k'} = \emptyset$ for all $k = k'$. In other words, the clusters are nonoverlapping: no observation belongs to more than one cluster.

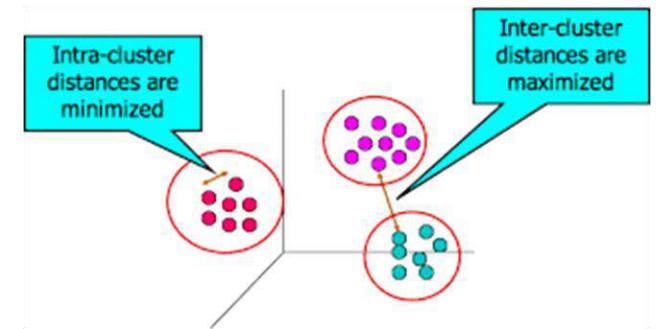
K-Means Clustering

- The idea behind K -means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.
- The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other.

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K W(C_k) \right\}.$$

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible
- The intra-cluster distance is measured using the Euclidian distance between pair wise instances in the cluster

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$$



Expectation Maximization of K-Means

- The E-step is assigning the data points to the closest cluster.
- The M-step is computing the centroid of each cluster.
- Lets prove that convergence is guaranteed
- **E-Step**
- where $w_{ik}=1$ for data point x_i if it belongs to cluster k ; otherwise, $w_{ik}=0$

$$J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \|x^i - \mu_k\|^2 \quad (1)$$

$$\begin{aligned} \frac{\partial J}{\partial w_{ik}} &= \sum_{i=1}^m \sum_{k=1}^K \|x^i - \mu_k\|^2 \\ \Rightarrow w_{ik} &= \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- **M-Step**

$$\begin{aligned} \frac{\partial J}{\partial \mu_k} &= 2 \sum_{i=1}^m w_{ik} (x^i - \mu_k) = 0 \\ \Rightarrow \mu_k &= \frac{\sum_{i=1}^m w_{ik} x^i}{\sum_{i=1}^m w_{ik}} \end{aligned} \quad (3)$$

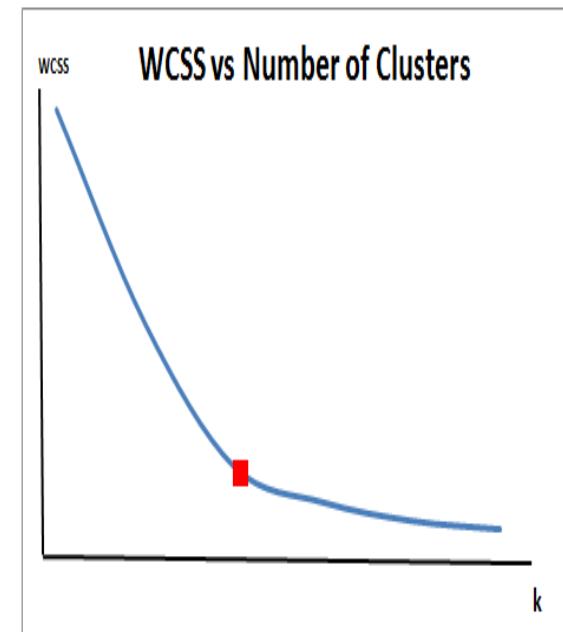
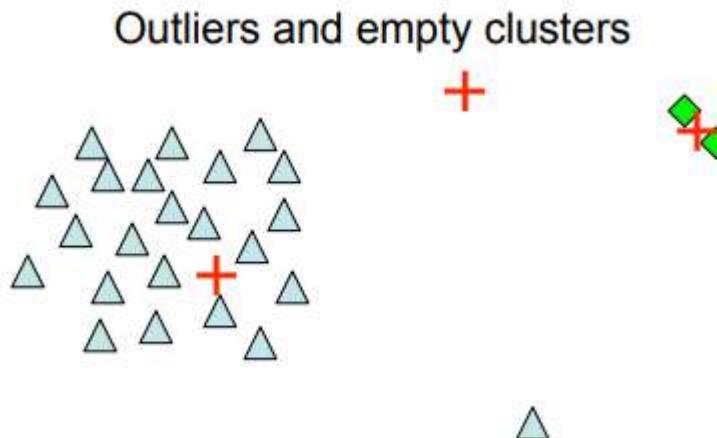
Closing Notes on K-Means

A normal distribution such that the mean $\mu = 0$ and standard deviation $\sigma = 1$ for your data

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} (-\infty < x < \infty).$$

K-Means algorithm converges to a local minimum:

- Can try multiple random restarts



Expectation Maximization

In the expectation, or E-step, **the missing data are estimated** given the

- observed data and
- current estimate of the model parameters

In the maximization or the M-step, **the likelihood function is maximized**

- under the assumption that the **missing data are known**

The estimate of the missing data from the E-step are used in lieu of the actual missing data.



THANK YOU

Srinivas K S

Department of Computer Science & Engineering

srinivasks@pes.edu



MACHINE INTELLIGENCE

Introduction to Genetic Algorithms

Dr. Arti Arya

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Introduction to Genetic Algorithms

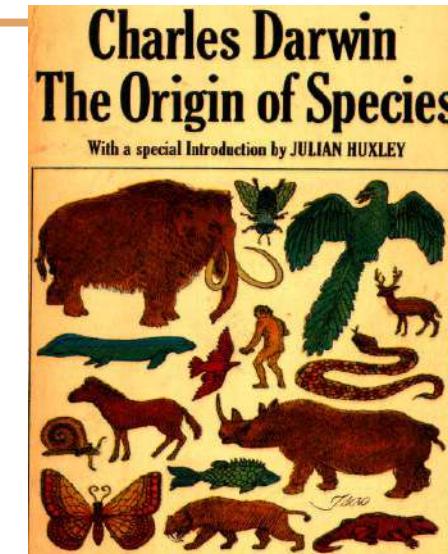
Dr. Arti Arya

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Genetic Algorithms- Introduction

- Genetic Algorithm (GA) is a metaheuristic search-based optimization technique inspired by the principles of **Genetics and Natural Selection** (*Darwin's theory of evolution: survival of the fittest*).
- Idea was introduced by applying to problem solving by Professor John Holland in 1965.
- It is frequently used to find *optimal or near-optimal solutions* to difficult problems which otherwise would take a lifetime to solve.
- Multiple points in the solution space are simultaneously considered for optimality, ie **global perspective of the solution space**. This avoids local optima.
- For detailed information you can further refer to [2].





- The nucleus contains the genetic information.

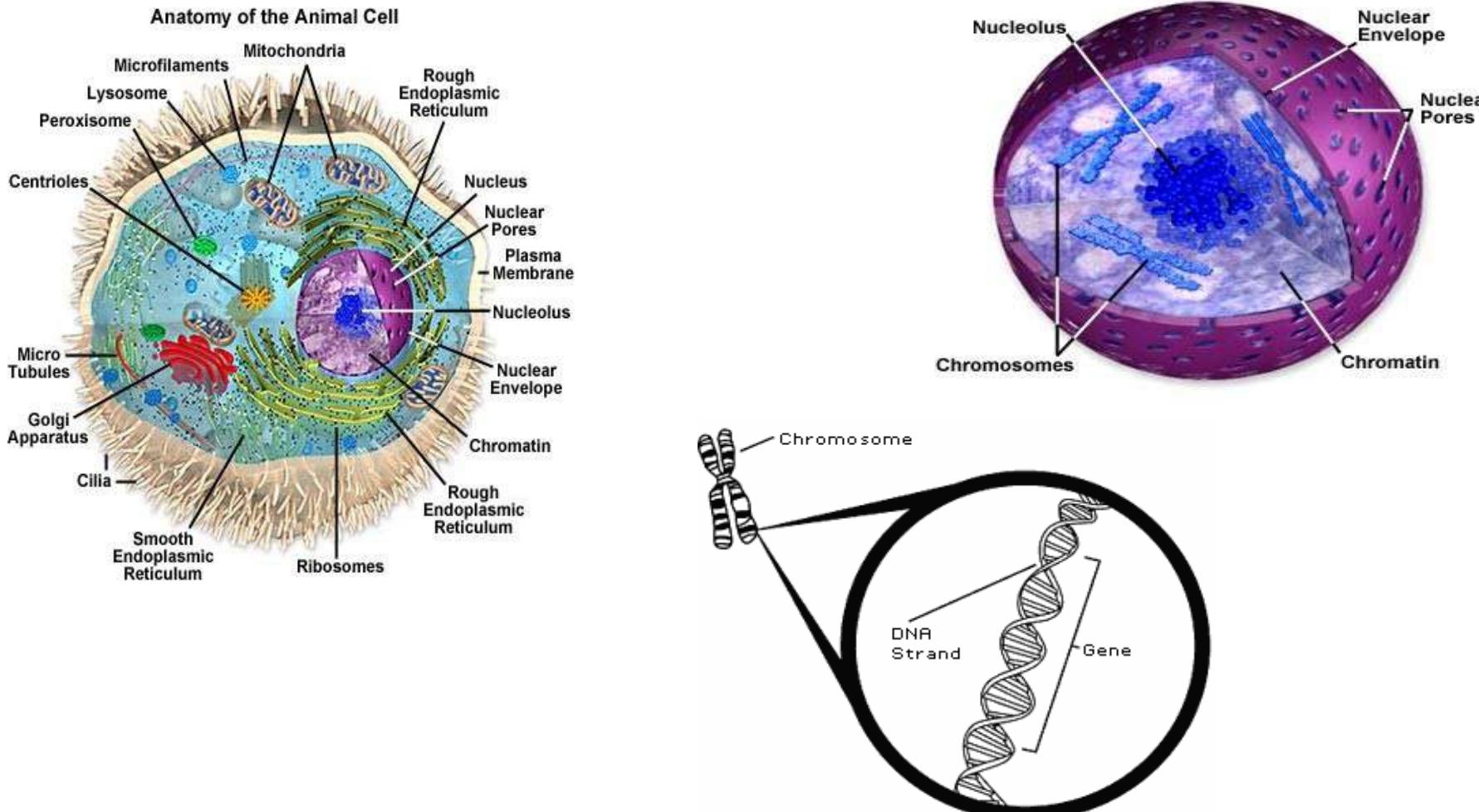


Figure from "Soft and Evolutionary Computing" by N P Padhy.

Connection Of GA with Optimization

- The set of all possible solutions or values of the problem, make up the search space.
- This search space has a point or a set of points which gives the optimal solution.
- The aim of optimization is to find that point or set of points in the search space and *so is the objective of GAs*.
- GA use only objective function information, not derivatives or other auxiliary knowledge.
- Thus GA can deal with the non-smooth, non-continuous and non-differentiable functions which actually exist in a practical optimization problem.

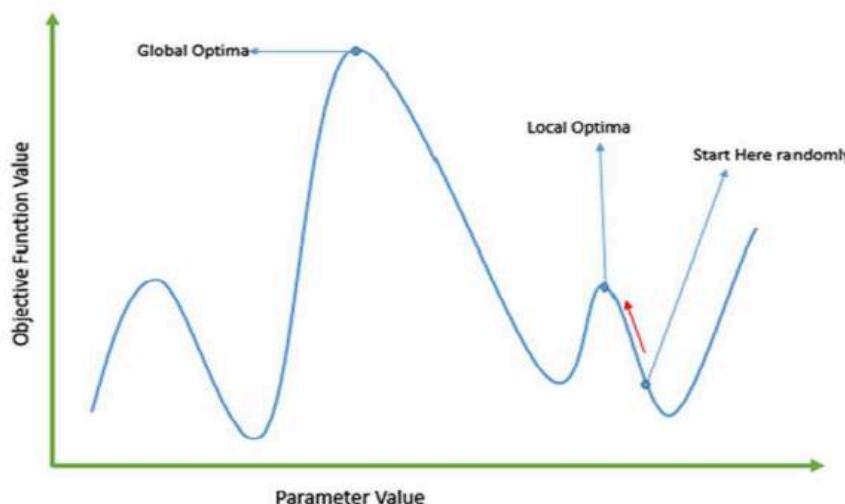
- Because Genetic Algorithms tend to find **globally optimal solutions**[1], which makes genetic algorithms attractive for solving optimization problems.
- **Need of GAs**

1. Solving Difficult Problems

- In computer science, there is a large set of problems, which are **NP-Hard**. For solving such problems, even the most powerful computing systems take a very long time (even years!) to solve that problem.

Genetic Algorithms- Motivation

- Failure of Gradient Based Methods
- Traditional techniques efficient and works very well for **single-peaked (unimodal) objective functions** like the cost function in linear regression.
- More complex problems having **multi-peaked (multimodal)objective functions** come across, which causes such methods to fail, as they suffer from an inherent tendency of getting stuck at the local optima.



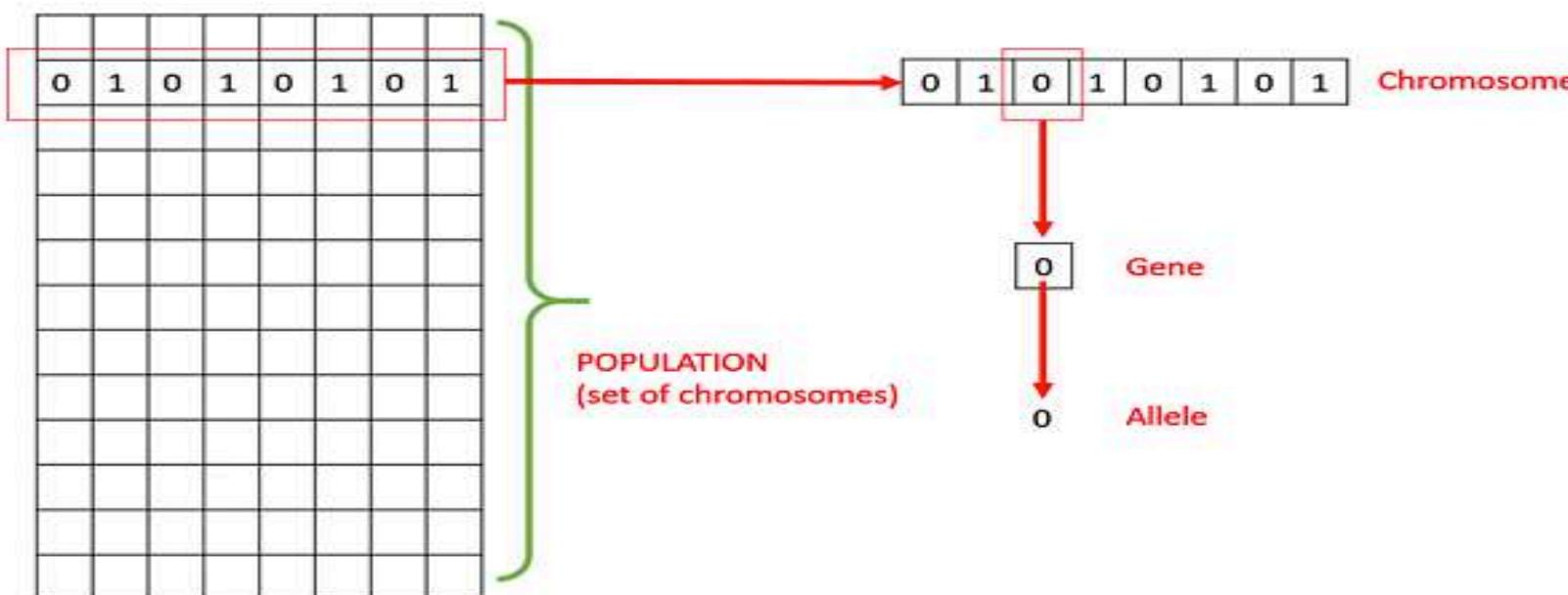
- **Getting a Good Solution Fast**

Some difficult problems like the Travelling Salesperson Problem (TSP), have real-world applications like path finding and VLSI Design.

MACHINE INTELLIGENCE

GA- Basic Terminologies

- **Population** – Subset of all the possible (encoded) solutions.
- **Chromosomes** – A chromosome is one such solution to the given problem.
- **Gene** – A gene is one element position of a chromosome.
- **Allele** – It is the value a gene takes for a particular chromosome.

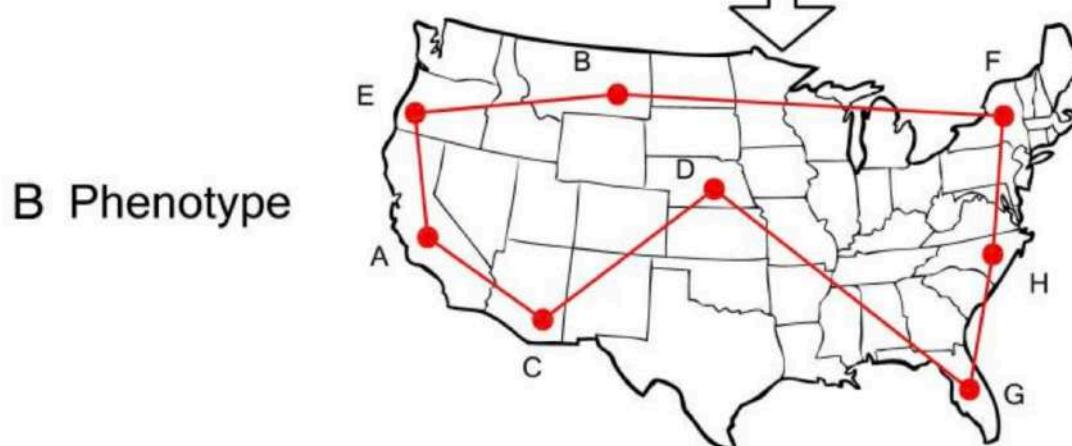


- **Genotype** – The set of genes representing the chromosome in computation space.
- **Phenotype** – is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

A Genotype

City E	City A	City C	City D	City G	City H	City F	City B
--------	--------	--------	--------	--------	--------	--------	--------

Mapping



Decoding and Encoding- Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space.

•**Fitness Function** – A fitness function takes the **solution as input** and produces the **suitability of the solution as the output**.

•**Genetic Operators** – These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

GA ()

 initialize population

 find fitness of population

while (termination criteria is reached) do

 parent selection

 crossover with probability pc

 mutation with probability pm

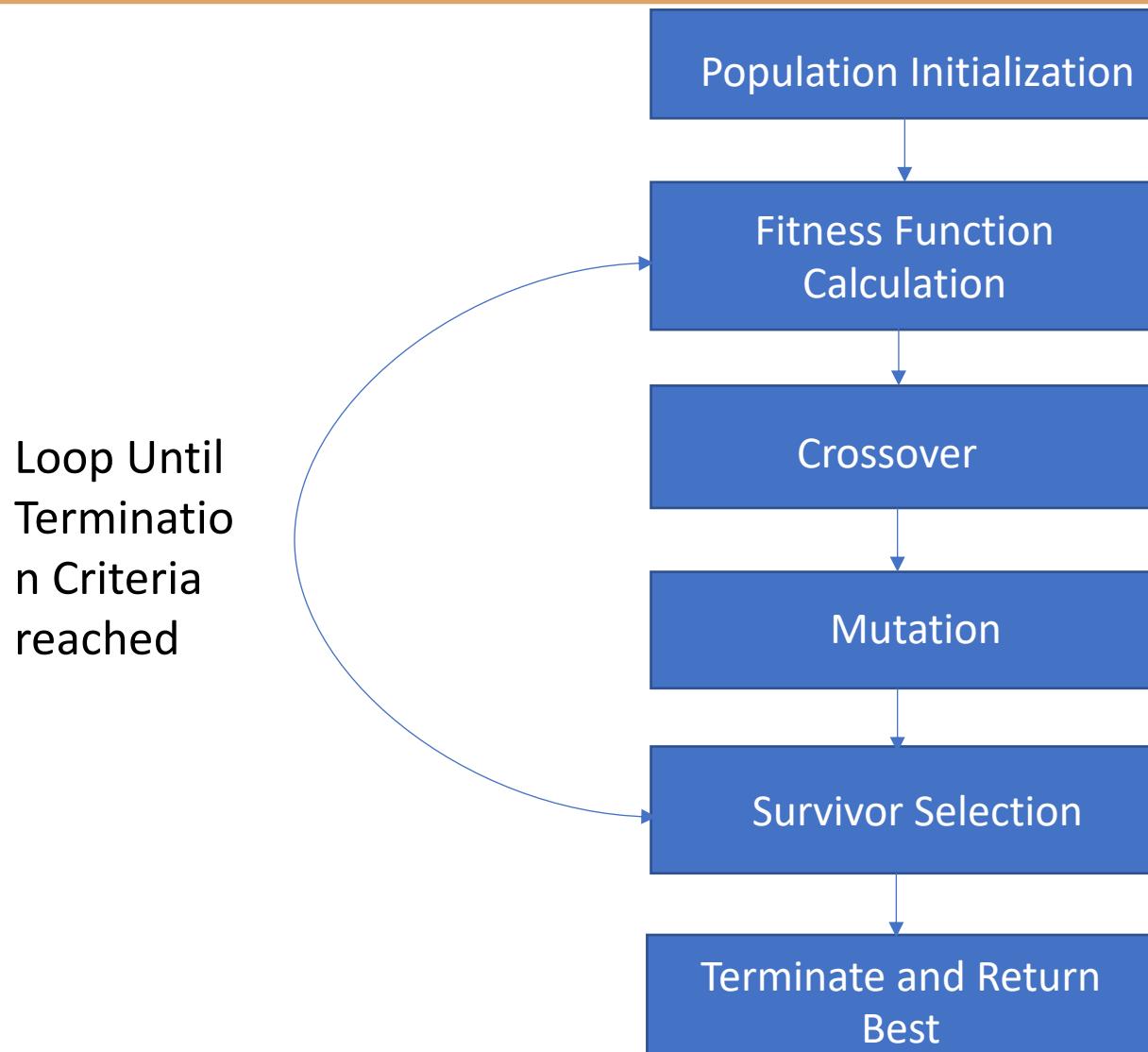
 decode and fitness calculation

 survivor selection

 find best

return best

- In GAs,
 - A pool or a population of possible solutions to the given problem is considered.
 - These solutions then undergo recombination and mutation (like in natural genetics), producing new offsprings.
 - And the process is repeated over various generations.
 - Each individual (or candidate solution) is assigned a fitness value and the fitter individuals will get a higher chance to mate and yield more “fitter” individuals.
 - In this way solutions “evolving” over generations, till a stopping criterion is met.



Genetic Representation(Encoding)

- *Encoding* is a process of representing chromosomes and thus genes.
- GAs algorithms do not deal with the solutions directly but with encoded representation of it.
- Chromosomal representation of a problem can be done by any of the following ways:
 - Binary string representation
 - Real number coding
 - Integer coding
 - Permutation Representation
 - Random key Representation
 - Representing genes in arrays, trees, lists and other objects

- Most of the time hypotheses in GAs are represented by **bit strings**.
- Bit strings are used for encoding bcoz it becomes easy to use genetic operators like **crossover and mutation**.
- Just as a simple example how bit strings would be used:
- Consider the classical example of “ Play Tennis” where Outlook takes 3 values ‘rainy’, ‘sunny’ and ‘overcast’.
- Say Outlook attribute is represented by a binary string of length 3 then
 - 100 represents Outlook has value “**rainy**”.
 - 110 represents Outlook can acquire “**rainy or sunny**”.

- Consider a rule's precondition
 $(\text{Outlook} = \text{Overcast} \vee \text{Rain}) \wedge (\text{wind} = \text{strong})$
- This can be encoded as
101 10 (How?)

Consider a rule:

If. Wind=Strong Then PlayTennis=yes

Encoding:

111 10 10 (Pls Explain how?)

Binary Encoding

The equivalent value for any n-bit string can be obtained by

$$X_i = X_i^l + \frac{X_i^u - X_i^l}{(2^{n_i} - 1)} \times (\text{decoded value of string})$$

Each variable has both the upper and lower limit which can be put in the form as

$$X_i^l \leq X_i \leq X_i^u$$

An 'n'-bit string can be represented by an integer between 0 to $2^n - 1$, which consists of 2^n integers

$$n = 4, \quad X_i^l = 4, \quad X_i^u = 25, \quad \text{string} = 1010$$

$$\text{Decoded value of string} = 1010 = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 0 = 10$$

The decoded value of 4 bit string is

$$X_i = 4 + \frac{21}{15} * 10 = 18$$

If 'p' is the precision required for a continuous variable then the string length 'S_i' should be equal to

$$S_i = \log_2 \left[\frac{X^u - X^l}{p} \right]$$

Example : Knapsack problem [4]

The problem: There are things with given value and size. The knapsack has a given capacity. Select things to maximize the value of things in knapsack, but do not extend knapsack capacity.

Encoding: Each bit says, if the corresponding thing is in knapsack.

Real Valued Representation

For problems where we want to define the genes using *continuous* rather than discrete variables, the real valued representation is the most natural[3]. The precision of these real valued or floating point numbers is however limited to the computer.

0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

Integer Representation

For discrete valued genes, we cannot always limit the *solution space to binary 'yes' or 'no'*. For example, if we want to encode the four distances – North, South, East and West, we can encode them as {1,2,3,4}. In such cases, integer representation is desirable.

1	2	3	4	3	2	4	1	2	1
---	---	---	---	---	---	---	---	---	---

Permutation representation (*Path Representation or Order representation*)

For example, a tour of 8-city Travelling Salesman Problem(TSP)

3-5-8-1-4-2-6-7 can be represented as [3 5 8 1 4 2 6 7]

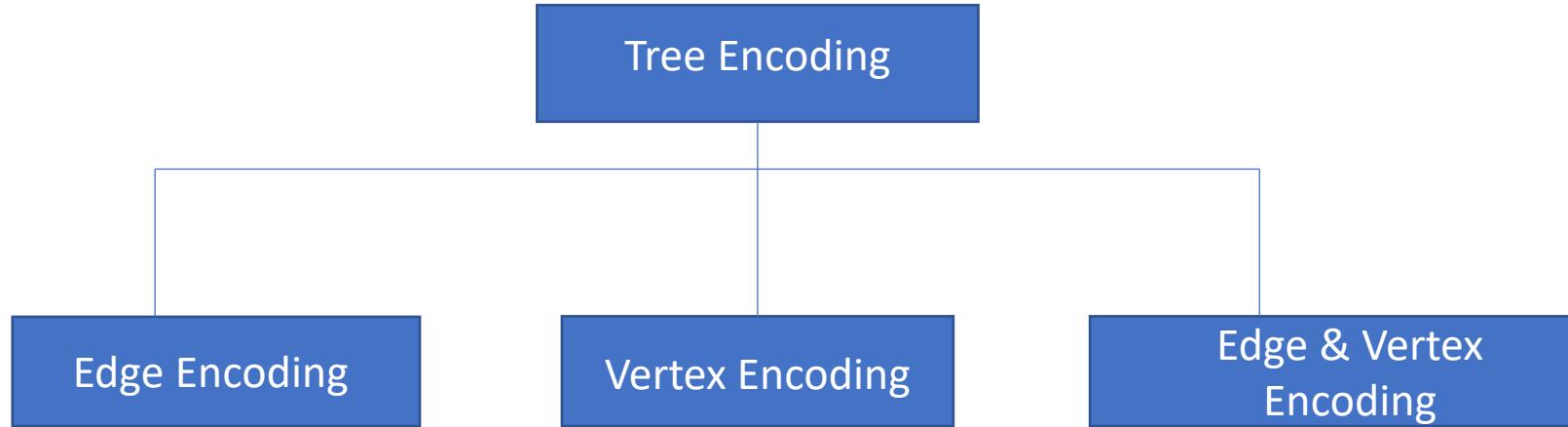
Random keys representation

Solution is encoded with random numbers from (0, 1) - 8 city TSP

[0.45 0.68 0.91 0.11 0.62 0.34 0.74 0.89] can be represented as 3-5-8-1-4-2-6-7

(machine scheduling- resource allocation- vehicle routing-quadratic assignment problem)

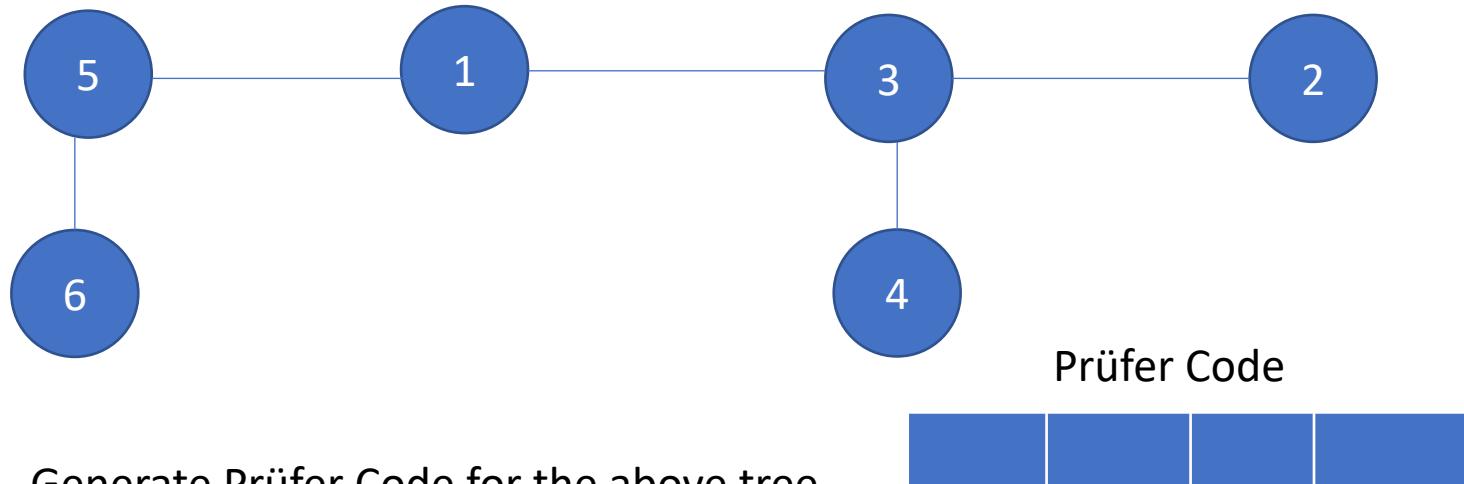
Tree Representation



For GA's, one method of **encoding input** is through **Tree Representation** i.e. a chromosome may be represented through a tree.

Vertex Encoding

Prufer's Code: Represents a tree in a particular way.

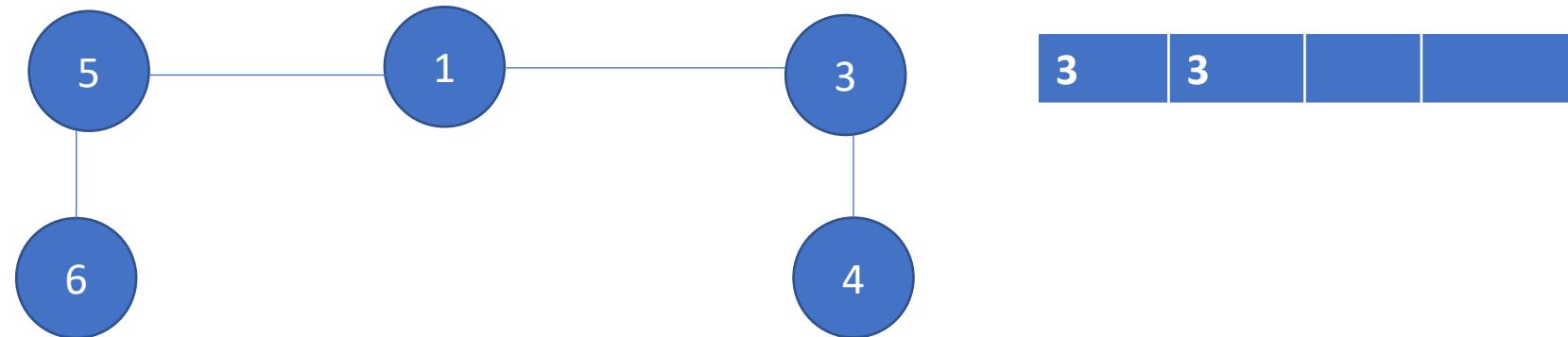


Generate Prüfer Code for the above tree.

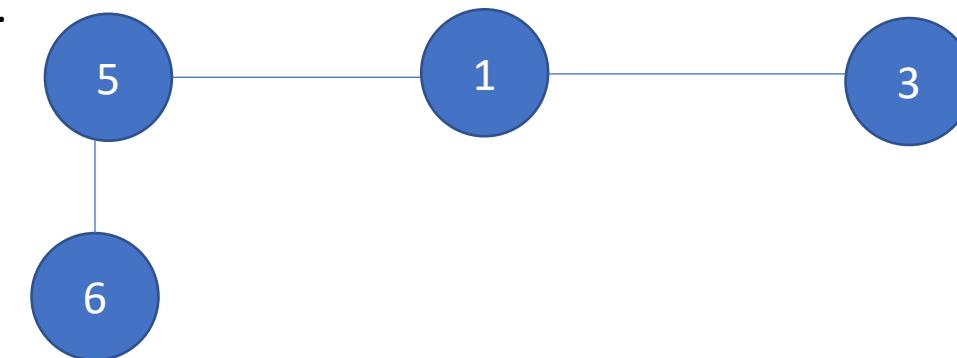
The length of Prüfer's Code is given by **n-2**, where n is number of nodes/vertices in the tree.

Step 1: Look for the leaf nodes with **the smallest label**. So, its **2** and edge incident from **node 2 is to node 3**. So **remove node 2** from the tree and **add 3** at the first position in the Prüfer code.

3			
---	--	--	--



Step 2: Now look for the leaf nodes with the **smallest label** again and its **node 4** and **edge incident from node 4 is to node 3**. So **remove 4** from the tree and add 3 at the second position in the Prüfer code.



Continuing like this till just one edge is left and we will have Prüfer's Code of length 4.

So,

3	3	1	5
---	---	---	---

 is the final Prüfer's Code corresponding to given tree.

MACHINE INTELLIGENCE

Representing Hypotheses

- Hypothesis in GAs are often represented by **bit strings**, which are easily manipulated by **crossover** and **mutation**.
- Consider sets of **if-then rules** that can easily be represented by bit strings, by choosing an **encoding of rules** that allocate specific substrings for each rule precondition and postcondition.

Representing Hypotheses- Example

- Attribute: **Outlook=Sunny, Rainy, Overcast**
 - Use a bit string of length 3, where each position corresponds to one of the values. Placing a 1 in some position indicates that the attribute is allowed to take on the corresponding value.
 - So 001 represents **Outlook = Overcast** and 011 represents **Outlook = Overcast or Rainy**

Id code	outlook	temp	humidity	windy	play
a	Sunny	high	high	strong	no
b	s	h	h	weak	n
c	overcast	h	h	s	y
d	Rainy	mild	h	s	y
e	r	cool	normal	s	y

More Representations

- **Conjunctions of constraints** can be represented by concatenation.

Eg. 011 10 represents **Outlook = Rainy or Overcast and Wind = Strong**

- Postconditions can be represented in the same way **111 10 10** represents **If Wind = Strong then PlayTennis = Yes**. Notice that **111** represents the “don’t care” condition on Outlook
- **Sets of rules** can be represented by concatenating single rules, but may not be fixed length!
- Some GAs represent hypothesis as symbolic descriptions rather than bit strings.

- [1] <https://www.solver.com/products-overview>
- [2] Holland JH. Genetic algorithms. *Scientific american*. 1992 Jul 1;267(1):66-73.
- [3] https://www.tutorialspoint.com/genetic_algorithms/
- [4] Hristakeva M, Shrestha D. Solving the 0-1 knapsack problem with genetic algorithms. In Midwest instruction and computing symposium 2004 Apr 16.



THANK YOU

Dr. Arti Arya

Department of Computer Science

artarya@pes.edu

+91 9972032451 Extn 029



Machine Intelligence Fitness, Selection and Genetic Operators

Dr. Arti Arya

Department of Computer Science

artiarya@pes.edu

+080-66186629 Extn 6629

MACHINE INTELLIGENCE

Fitness, Selection and Genetic Operators

Dr. Arti Arya

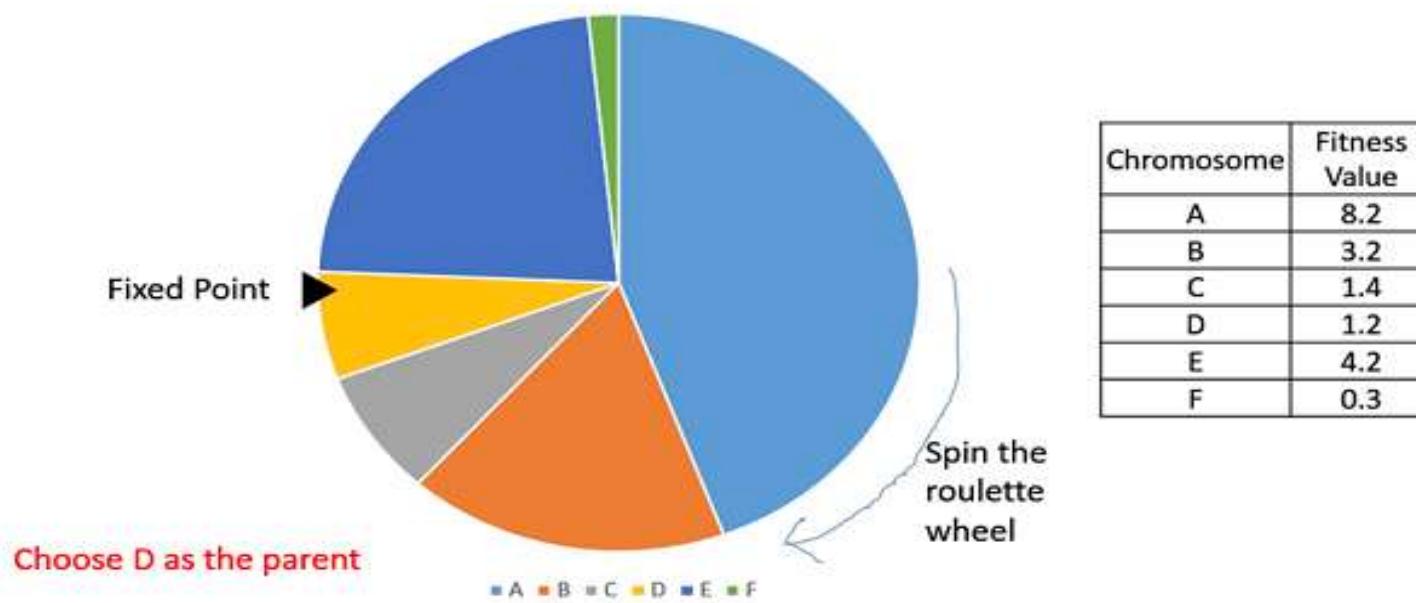
Department of Computer Science and Engineering

The **fitness function defines the criterion for ranking potential hypotheses** and for probabilistically selecting them for inclusion in the next generation population.

- **Fitness proportionate selection (*Roulette wheel*)** : Ratio of fitness to the fitness of other members of the current population.

Roulette Wheel Selection

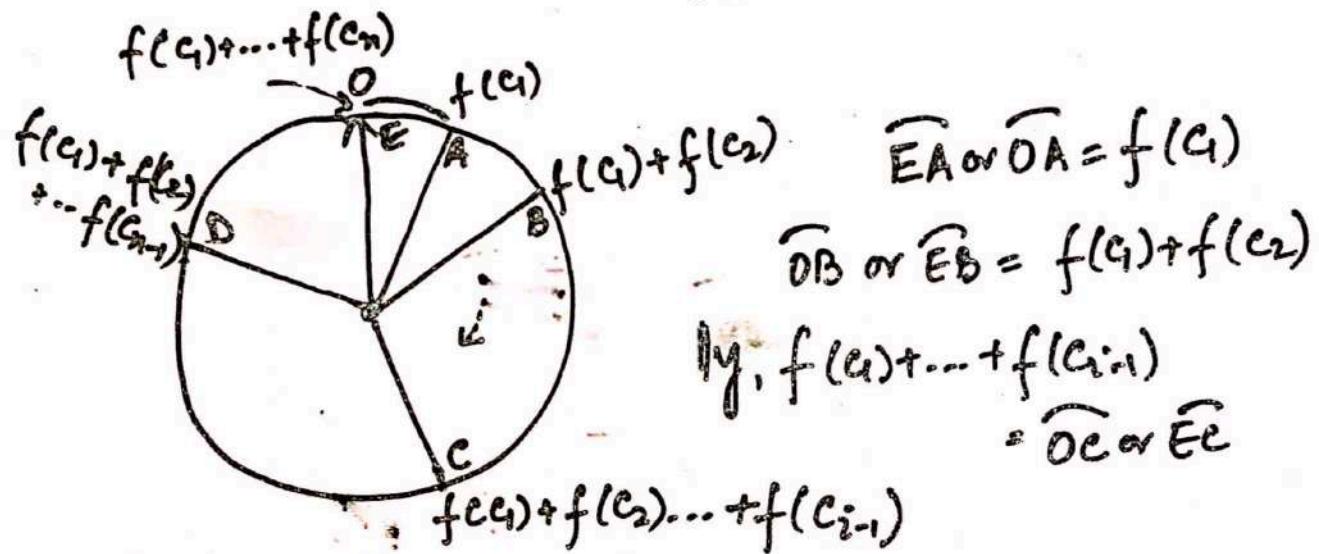
- In a roulette wheel selection, the circular wheel is divided as described below. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated.
- The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.



Consider a population of size n .

Let c_1, c_2, \dots, c_n be n chromosomes with fitness values $f(c_1), f(c_2), \dots, f(c_n)$.

Suppose there is a disc of circumference = $\sum_{i=1}^n f(c_i)$





In order to select a chromosome to be included in the mating pool from the current population,

a random no. is generated bet "
 $0 \text{ & } \sum_{i=1}^n f(c_i)$ "

Let one such random no. be 'x'.

Now locate a point Y on the circumference of the wheel which is at a distance 'x' from the starting point O or E.

The chromosome within whose limits Y is located will be selected for the mating pool.

This process is repeated 'n' times as population size=n.

Roulette Wheel Selection-Example

Keeping a TSP for 5 cities in mind. Consider the following set of chromosomes(possible solutions) in an initial population:

S.No	Chromosome	Fitness
1	d-e-a-b-c	193
2	c-e-b-d-a	172
3	e-a-b-c-d	193
4	d-e-a-b-c	193
5	a-d-e-b-c	141
6	c-e-d-a-b	197
7	e-a-d-b-c	222
8	c-d-e-a-b	193
9	e-a-b-d-c	224
10	a-d-e-b-c	141

$$\sum f(c_i) = 1869, i=1, \dots, 10$$

Let a random no. is generated between $(0, 1869)$, say **1279**

(it can be any number between $(0, 1869)$)

Since 1279 lies between $\sum f(c_i) = 1089, i=1 \dots 6$.
and $\sum f(c_i) = 1311, i=1 \dots 7$.

As $1089 < 1279 < 1311$

So, **7th Chromosome** is selected.

Pls refer [2] in References
for a simple mathematical
problem using GA

- **Tournament Selection:**
 - Two hypothesis (chromosomes) chosen at random, with some predefined probability, p , the more fit is selected.
- **Rank Selection:**
 - Sorted by fitness. The probability that a hypothesis will be selected is then proportional to its rank.
- **Elitist Selection (Elitism):**
 - Ensures that the best chromosomes are retained in the next generation, if not selected by any other method.



THANK YOU

Dr. Arti Arya

Department of Computer Science

artiarya@pes.edu

+91 9972032451 Extn 029



Machine Intelligence Genetic Operators

Dr. Arti Arya

Department of Computer Science

artarya@pes.edu

+080-66186629 Extn 6629

MACHINE INTELLIGENCE

Genetic Operators

Dr. Arti Arya

Department of Computer Science and Engineering

Broadly 2 categories of Operators

Crossover (recombination)

Exchanging genetic material between the individuals of the chromosomal population

Mutation

A genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next

Note: A genetic operator which works well for one problem may not work well for another problem. [1]

- Crossover point is randomly selected.

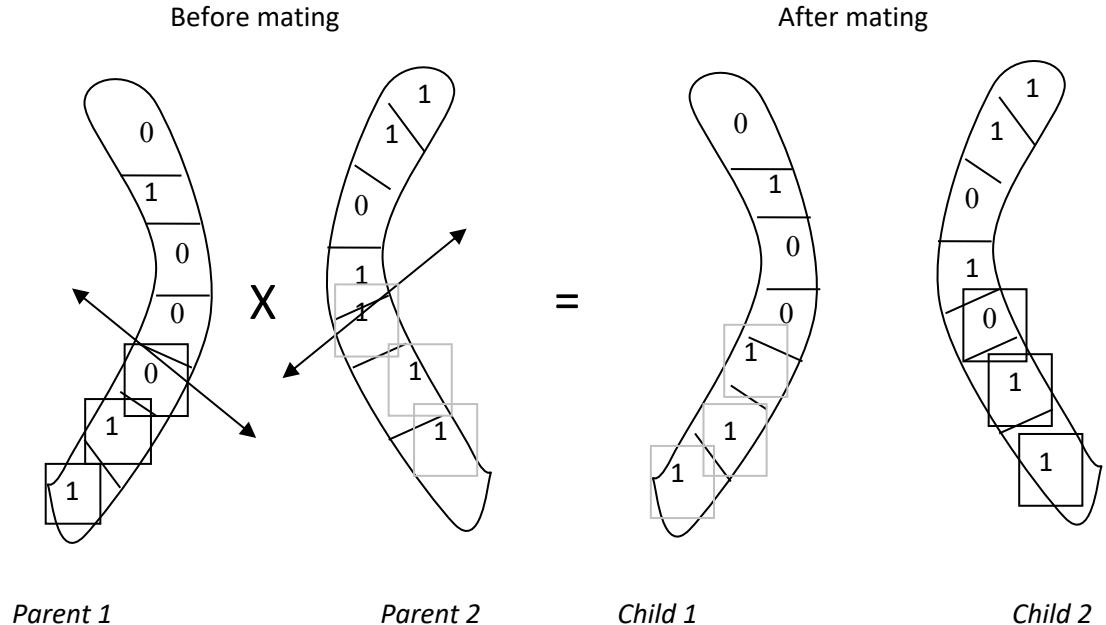
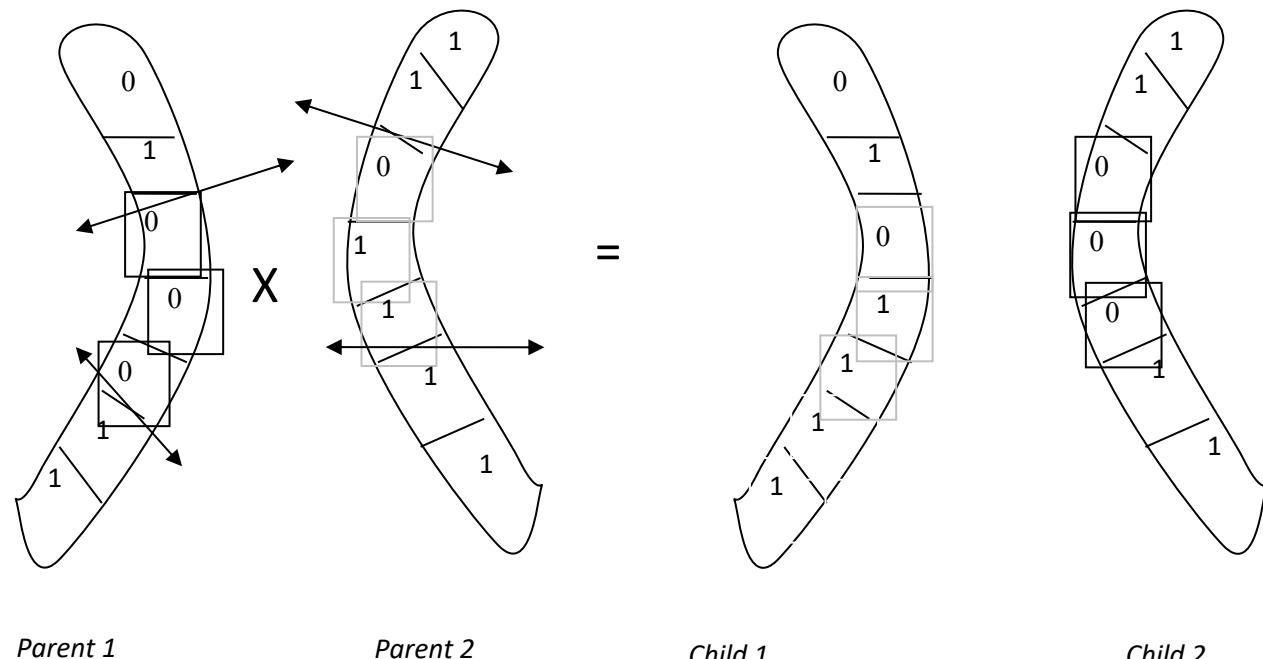


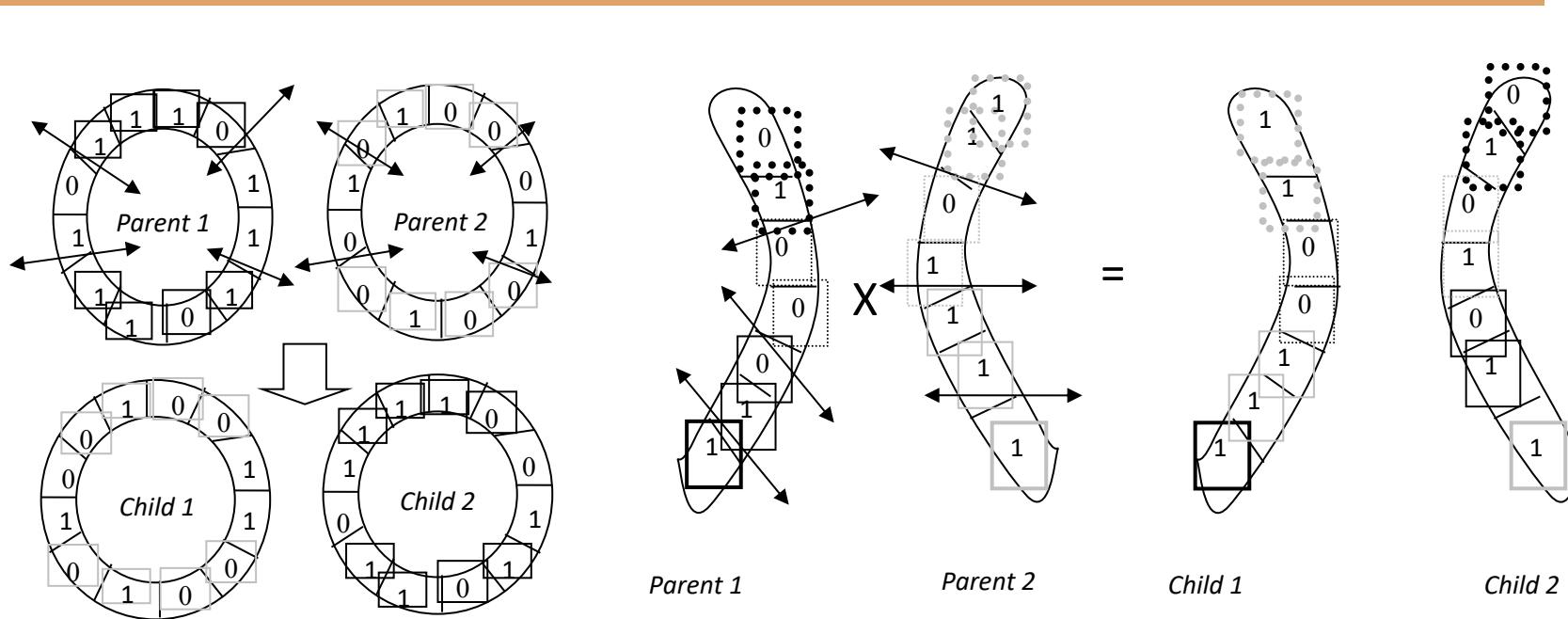
Figure Courtesy: N.P Padhy, "Soft Computing"

- Two random sites are chosen.
- The bits in between the two points are swapped between the parent chromosomes.



MACHINE INTELLIGENCE

GA- Multipoint Crossover



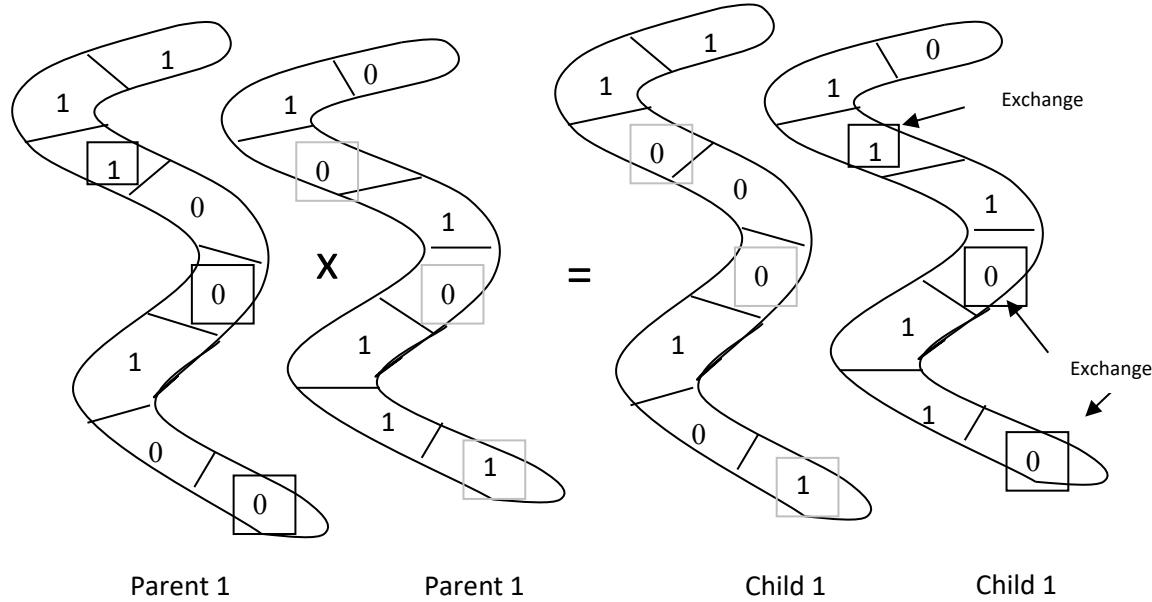
Even cross-site multi - point cross over

Figure Courtesy: N.P Padhy, "Soft Computing"

- Multi point crossover is a generalization of the **one-point crossover** wherein **alternating segments** are swapped to get new off-springs.
- One-point and N-point crossover have a tendency to exhibit **positional bias** and **inherent bias** which have proved to be experimentally evident.

MACHINE INTELLIGENCE

GA- Uniform Crossover



Exchanging bits from two individual parental chromosomes by maintaining a probability of 0.5 to produce offsprings.

GA- Masking Based Uniform Crossover

Uniform crossover does not exhibit positional bias but do exhibit distributional bias due to which uniform crossover has a strong tendency towards transmitting 50% of the genes from each parent.

P1	1	1	0	0	0	1	0	1
Mask	1	0	1	1	0	0	0	1
P2	1	0	1	0	1	0	0	1
C1	1	0	0	0	1	0	0	1
C2	1	1	1	0	0	1	0	1

P1=1, P2=0
P1=0, P2=1

MACHINE INTELLIGENCE

GA- Matrix Crossover

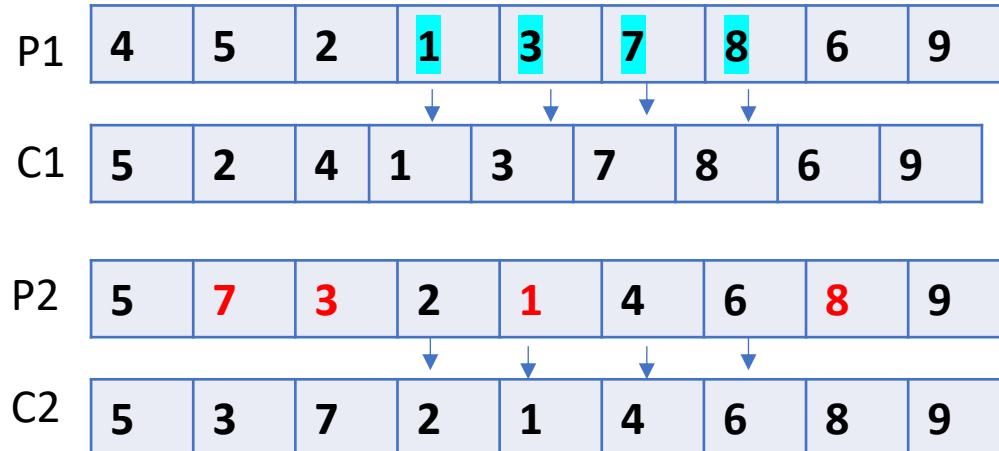
$$\begin{array}{c} P1 \\ \left[\begin{array}{ccc} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right] \end{array} \times \begin{array}{c} P2 \\ \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{array} \right] \end{array}$$



$$\begin{array}{c} C1 \\ \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right] \end{array} \quad \begin{array}{c} C2 \\ \left[\begin{array}{ccc} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{array} \right] \end{array}$$

Cross sites of row and column are chosen randomly.

Select any region between in each layer, either vertically or horizontally and then exchange the information in the region between the mated populations.



- Set of randomly selected string of genes from one parent is considered (**1-3-7-8** from parent 1)
- Transfer the selected genes of parent 1 from child 1 at the same locations.

- Delete **1-3-7-8** from *parent 2* and remaining genes are 9,6,4,2,5 in P2.
- Starting from the bottom to top of C1, fill the empty strings of C1 with the remaining genes of Parent 2 ie 9,6,4,2,5.

P1	4	5	2	1	3	7	8	6	9
----	---	---	---	---	---	---	---	---	---

C1	7	5	3	1	2	4	8	6	9
----	---	---	---	---	---	---	---	---	---

P2	5	7	3	2	1	4	6	8	9
----	---	---	---	---	---	---	---	---	---

C2	4	5	2	1	3	7	8	6	9
----	---	---	---	---	---	---	---	---	---

- Set of randomly selected positions from P1 (**5-1-8-6**) are copied to C1.
- **Delete** the selected nodes of **P1** from P2.

The **remaining nodes 9-4-2-3-7** of **parent 2** is transferred to its **child 1** from bottom to top.

- Mutation is a genetic operator used to maintain *genetic diversity* from one generation of a population of genetic algorithm chromosomes to the next.
- It is analogous to biological mutation. Mutation alters *one or more gene values* in a chromosome from its initial state.
- Mutation helps preventing the *population of chromosomes from becoming too similar to each other*, thus slowing or even stopping evolution. So, it helps the algorithm to *avoid local extreme*
- Mutation allows the *development of un-inherited characteristics* --- it promotes diversity by allowing an offspring to also evolve in ways not solely *determined by inherited traits*.

- After crossover, the strings are subjected to mutation.
- This process enables GAs to jump out of local or suboptimal solutions to avoid premature convergence.
- Mutation plays the *role of recovering the lost genetic material* as well as *for randomly distributing genetic info.*

a) Uniform mutation

$X^t = [X_1, X_2, \dots, X_m]$ - Chromosome

A random number is selected such that $k \in [1, m]$

$X^{t+1} = [X_1, \dots, X'_k, \dots, X_m]$ – Off- Spring

X'_k - A random value generated according to uniform probability distribution from the range $[X_k^L, X_k^U]$

b) Boundary mutation

$X'_k \rightarrow X_k^L \text{ or } X_k^U$

Replacement takes place with equal probability

- Mutation can be implemented as *One's complement Operator.*
- Let C= [01101011] be a chromosome, then one's complement operator gives C=[10010100].
- **Inversion**
 - **Two positions** are randomly selected within a chromosome. Then the substring between these positions is inverted.

$$P_x = [2 \ 5 \ 4 \ \underline{9 \ 6 \ 8} \ 1 \ 3 \ 7]$$

$$C_x = [2 \ 5 \ 8 \ \underline{6 \ 9} \ 4 \ 1 \ 3 \ 7]$$

- **Insertion**
 - A node is randomly selected and inserted in a random position.
 - $P_x = [2 5 4 9 \underline{6} 8 1 3 7]$
 - $C_x = [\underline{2} 6 5 8 9 4 1 3 7]$

- **Heuristic Mutation**
 - It is based on neighborhood technique
- $P_x = [1 2 \underline{3} 4 5 \underline{6} 7 \underline{8} 9]$
- Neighbors are formed with these nodes.

$N_x = [1 2 \underline{3} 4 5 \underline{8} 7 \underline{6} 9]$

$N_x = [1 2 \underline{8} 4 5 \underline{3} 7 \underline{6} 9]$

$N_x = [1 2 \underline{8} 4 5 \underline{6} 7 \underline{3} 9]$

$N_x = [1 2 \underline{6} 4 5 \underline{8} 7 \underline{3} 9]$

$N_x = [1 2 \underline{6} 4 5 \underline{3} 7 \underline{8} 9]$

After evaluating all the neighbors the best neighbor is selected.

Crossover Probability:

Measure of likeliness that the selected chromosomes will be undergo crossover operation.

Crossover rate:

The total no. of crossovers that will be performed on the selected chromosomes.

Mutation Probability:

Measure of likeliness that random elements of the chromosomes will be changed with something else.

Mutation rate:

The no. of chromosomes that will undergo the mutation out of a total population.

Genetic Algorithm

1. Often produces invalid states.
2. GA's are probabilistic search algorithms which mimic the process of natural evolution.
3. In GA's, each individual is a candidate solution.
4. GA has an o/p which is a quantity

Genetic Programming

1. Special case of GAs, where each individual is a computer program
2. GP explores the algorithmic search space and evolve computer program to perform a defined task.
3. GP is an application of GA. O/p of GP is another program.

MACHINE INTELLIGENCE

References

- [1] Xin Yao, An empirical study of genetic operators in genetic algorithms, Microprocessing and Microprogramming, Volume 38, Issues 1–5, 1993, Pages 707-714, ISSN 0165-6074.
- [2] Hermawanto D. Genetic algorithm for solving simple mathematical equality problem. arXiv preprint arXiv:1308.4675. 2013 Aug 16.
- [3] An empirical study of genetic operators in genetic algorithms X Yao Microprocessing and Microprogramming, 1993 - Elsevier



THANK YOU

Dr. Arti Arya

Department of Computer Science

artarya@pes.edu

+91 9972032451 Extn 029



Machine Intelligence

Full Example of Genetic Algorithm



Dr. Arti Arya
Department of Computer Science
artiarya@pes.edu
+080-66186629 Extn 6629

MACHINE INTELLIGENCE

Full Example of Genetic algorithm

Dr. Arti Arya

Department of Computer Science and Engineering

A Solved Example of GA

Consider the following optimization problem:

$$\begin{aligned} \text{Min. } f(x) &= x \sin 10\pi x + 1.0 \\ \text{s.t.} \\ -1 &\leq x \leq 2 \end{aligned}$$

Let crossover rate be 0.5 and mutation rate be 0.1. Consider one generation with a population of 10 chromosomes.

A Solved Example of GA

Step 1: Initialization and Evaluation of the population

Let the initial population be :

**-0.4691, 0.1572, -0.5914, 0.6454, 0.7796,
1.0580, 0.0991, 0.2772, 0.5605, 1.6327**

Compute the cost (value of $f(x)$) corresponding to each initial value of x (initial chromosome in the population)

x	y=f(x)
-0.4691	1.3872
0.1572	0.8468
-0.5914	0.8422
0.6454	1.6387
0.7796	0.5339
1.0580	2.0248
0.0991	1.0028
0.2772	1.1820
0.5605	0.4697
1.6327	2.3962

A Solved Example of GA

Sort the values of $f(x)$ in ascending order.

Min $f(x)=0.4697$ for $x=0.5605$

Step 2: Selection (Roulette Wheel)

Normalizing the values of $f(x)$ between 0.1 and 0.9.

Lets normalize using

$$f(x) = \frac{(max. range - min. range)(f(x) - min. f(x))}{max. f(x) - min f(x)} + min. range$$

Here min. range=0.1, max. range=0.9

$f(x_1)=0.4697$ (this is also min cost)

$f(x_2)=2.3962$ (max cost)

x	y=f(x)	Normalized
0.5605	0.4697	0.1000
0.7796	0.5339	0.1269
-0.5914	0.8422	0.2584
0.1572	0.8468	0.2566
0.0991	1.0028	0.3213
0.2772	1.1820	0.3958
-0.4691	1.3872	0.4808
0.6454	1.6387	0.5855
1.0580	2.0248	0.7458
1.6327	2.3962	0.9000

GA Example contd....

For selection,

Probability of i^{th} chromosome to be selected in the pool of population of size n for crossover operation is

$$\text{Prob.} = \sum_{i=1}^n \frac{\text{normalized cost of individual chromosome}}{\text{sum of all the normalized costs}}$$

Finally we'll *flip these probabilities* so that **best chromosome gets the highest probability.**

The best chromosome is the one with the minimum cost and should have highest probability of selection.

x	Normalized f(x)	Prob.	Flipped Prob.
0.5605	0.1000	0.02339	0.2159
0.7796	0.1269	0.03045	0.1789
-0.5914	0.2584	0.06114	0.1405
0.1572	0.2566	0.06157	0.1154
0.0991	0.3213	0.0771	0.0950
0.2772	0.3958	0.0950	0.0771
-0.4691	0.4808	0.1154	0.06157
0.6454	0.5855	0.1405	0.06114
1.0580	0.7458	0.1789	.03045
1.6327	0.9000	0.2159	0.02339

GA Example contd.....

Chromo.	Cost in ascending order	Normalized Prob.	Flipped Prob.	Cumulative Prob.	Random Value generated for selection.
0.5605	0.4697	0.1000	0.2389	0.2160	0.6777
0.7796	0.5344	0.1269	0.13045	0.3949	0.2345
-0.5914	0.8424	0.2584	0.06114	0.5354	0.6190
0.1572	0.8468	0.2566	0.06157	0.6508	0.1389
0.0991	1.0027	0.3213	0.0771	0.7458	0.8430
0.2772	1.1820	0.3958	0.0950	0.8229	0.0879
-0.4691	1.3868	0.4808	0.1154	0.8844	0.9805
0.6454	1.6389	0.5855	0.1405	0.9456	0.7784
1.0580	2.0249	0.7458	0.1789	0.9760	0.4111
1.6327	2.3962	0.9000	0.2159	1.0000	0.2513

For Roulette Wheel Selection Process,

Consider a random no. say 0.6777 and this random no. fits between 0.6508 and 0.7458

Therefore, 5thchromosome ie 0.0991 is selected.

GA Example contd....

So, consider another random number=0.2345

It fits between 0.2160 and 0.3949

Therefore, Chromosome corresponding to 0.3949 ie 2nd chromosome(0.7796) is selected.

For this problem the random nos. considered are 0.6190 (4th Chrom), 0.1389(1st Chro), 0.8430(7th Chrom), 0.0579(1st Chro), 0.9805(10th Chrom), 0.7784(6th), 0.4111(3rd), 0.2513 (2nd) are selected.

Like this, 10 chromosomes that are best fit are selected.

Lets divide the pool of 10 chromosomes into 2 subsets of 5 chromo each.

Population	
Mate 1	Mate 2
#5 (0.0991)	# 1 (0.5605)
#2 (0.7796)	# 10 (1.6327)
#4 (0.1572)	# 6 (0.2772)
#1 (0.5605)	# 3 (-0.5914)
#7 (-0.4691)	# 2 (0.7796)

GA Example contd....

Step 3: Crossover

As crossover rate =0.5

Therefore, # of crossovers=5

And # of mutations =1

Crossover operator used is Directional crossover given as follows:

$$C_1 = P_1 + R(P_2 - P_1)$$

$$C_2 = P_2 + R(P_1 - P_2)$$

Where R is a random number

(The random nos. considered for crossover step are

0.3451,0.4539,0.8674,0.1004,0.7801)

GA Example contd....

Selection of Chromosome for mating		Random number for Directional X	Crossover Operator
5 th (0.0991)	1 st (0.5605)	0.3451	$C_5: 0.0991 + 0.3451(0.5605 - 0.0991) = 0.2583$ $C_1: 0.5605 + 0.3451(0.0991 - 0.5605) = 0.4013$
2 nd (0.7796)	10 th (1.6327)	0.1004	$C_2: 0.7796 + 0.4539(1.6327 - 0.7796) = 1.1668$ $C_{10}: 1.6327 + 0.4539(0.7796 - 1.6327) = 1.2454$
4 th (0.1572)	6 th (0.2772)	0.4539	$C_4: 0.1572 + 0.8674(0.2772 - 0.1572) = 0.2613$ $C_6: 0.2772 + 0.8674(0.1572 - 0.2772) = 0.1731$
1 st (0.5605) (0.4013)	3 rd (-0.5914)	0.7801	$C_1: 0.4013 + 0.1004(-0.5914 - 0.4013) = 0.3016$ $C_3: -0.5914 + 0.1004(0.4013 + 0.5914) = -0.4917$
7 th (-0.4691)	2 nd (0.7796) (1.1668)	0.8674	$C_7: 0.8071$ $C_2: -0.1094$

The new chromosomes so obtained will be looked into for whether they lie within the boundary limits of -1 and 2 , if not then parent chromosome will not be replaced by the new offspring.

GA Example contd....

Mutation

Lets select 3rd chromosome(offspring) randomly.

Replace it by a new random no. between -1 and 2.

Population after crossover	Randomly selected chromo .	Replace the selected chromo . with new mutation limits .
0.3016		0.3016
-0.1094		-0.1094
0.4917	Let's select 3 rd chromo . at random .	-0.6283
0.2613		0.2613
0.2583		0.2583
0.1731		0.1731
0.8071		0.8071
0.6454		0.6454
1.0580		1.0580
1.2454		1.2454

New population for the next generation.

Step 5: Termination

Depending on how many generations or any other criteria for stopping the algorithm.



THANK YOU

Dr. Arti Arya

Department of Computer Science

artiarya@pes.edu

080-66186629 Extn 6629



MACHINE INTELLIGENCE

An Application of GA on Decision Trees for Optimal Results

Dr. Arti Arya

Department of Computer Science

artiarya@pes.edu

+080-66186629 Extn 6629

MACHINE INTELLIGENCE

An Application of GA on Decision Trees
for Optimal Results

Dr. Arti Arya

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Decision Trees Using Genetic Algorithm

Consider a problem statement as follows:

In order to serve mobile users efficiently, a mobile service provider wants to know the type of his customers in order to reach out to them better.

Can you help the mobile service provider??

Hint: Consider the title of the slide as a constraint!!!

Basically, as **Computer Science Engineers**, we could provide this Service provider with a system that classifies the Mobile User Dataset.



For Mobile Users classification, authors have found it different from General User Classification because:

1. Other than general attributes, **context information** providing attributes are considered.
2. Not all attributes **contribute equally** towards the classification.

So, a dataset is provided by the mobile service provider to you.

You will undergo the following steps:

1. Preprocessing
2. Rule generation from Decision Trees
3. Optimization of the Rules by Genetic Algorithm
4. Test the Optimized Rules

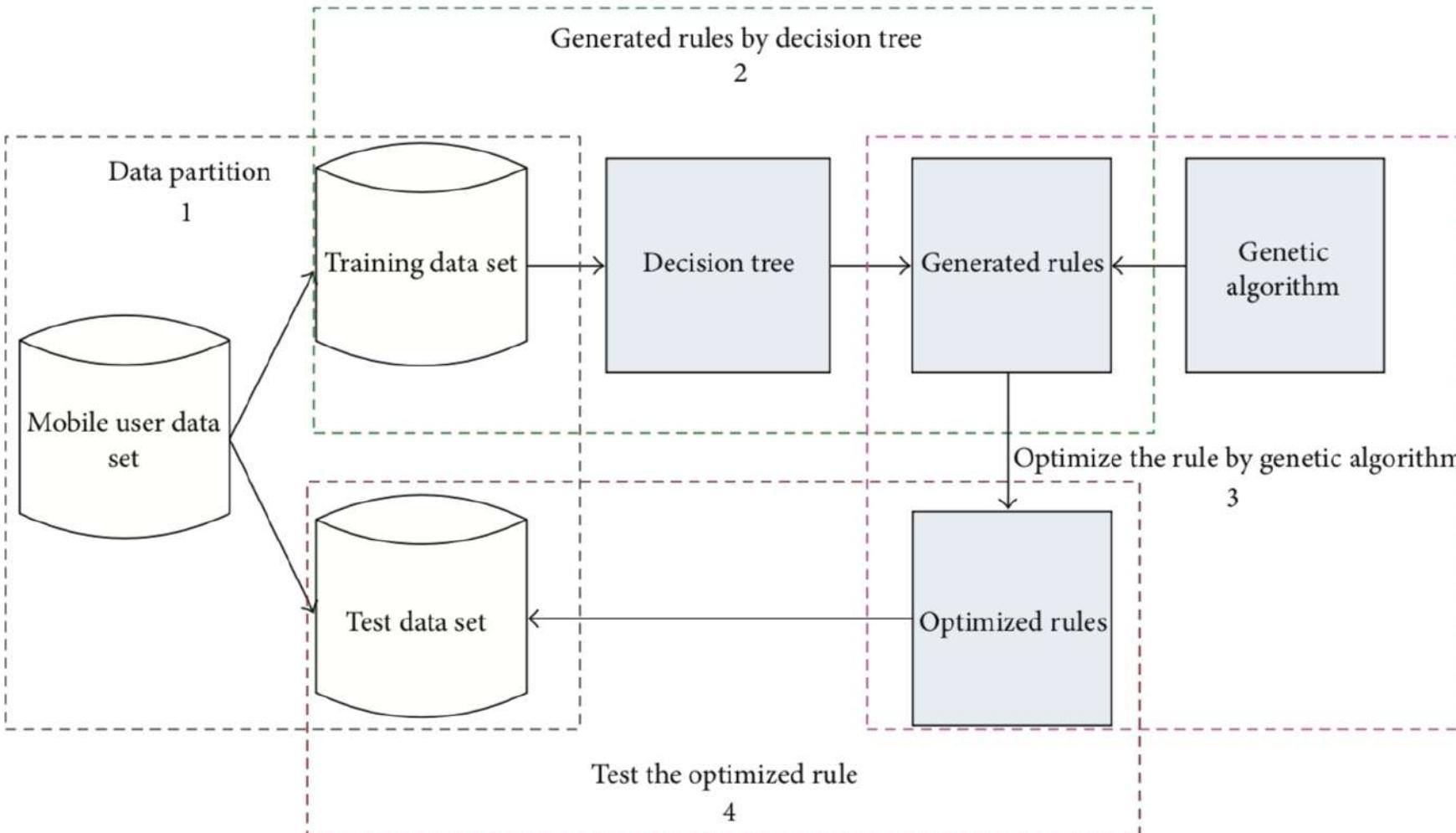


Fig Courtesy: [1] Liu DS, Fan SJ. A modified decision tree algorithm based on genetic algorithm for mobile user classification problem. The Scientific World Journal. 2014;2014

Entropy, Gain Ratio and Split Information is used to find the most appropriate tree.

$$I(s_1, s_2, \dots, s_m) = - \sum_{i=1}^m p_i \log_2(p_i),$$

$$E(A) = \sum_{i=1}^v \frac{(s_{1j} + s_{2j} + \dots + s_{mj})}{s} I(s_{1j}, s_{2j}, \dots, s_{mj}).$$

$$I(s_{1j}, s_{2j}, \dots, s_{mj}) = - \sum_{i=1}^m p_{ij} \log_2(p_{ij}),$$

$$\text{Split_info}(A) = \sum_{j=1}^v \left| \frac{s_j}{s} \right| \log_2 \left(\left| \frac{s_j}{s} \right| \right),$$

$$\text{gain_ration}(A) = \frac{\text{Gain}(A)}{\text{Split_info}(A)}.$$

Brush up off Binary Encoding

- Each **chromosome** represents a **classification rule**.
- Some chromosomes will become the **solution of problem**.
- The final rule set will be **sorted by the quality** of the rule.
- When the rule set recognizes a new sample, **the best rule will be considered firstly**; if the best rule cannot recognize the sample, then we can choose the next rule.
(Rule based ordering.....Remember!!!!)
- If the rule in the rule set cannot recognize the sample either, the sample will be classified as default class.

- [1] Liu DS, Fan SJ. A modified decision tree algorithm based on genetic algorithm for mobile user classification problem. *The Scientific World Journal*. 2014;2014

- [2] Khatwani S, Arya A. A novel framework for envisaging a learner's performance using decision trees and genetic algorithm. In *2013 International Conference on Computer Communication and Informatics 2013 Jan 4* (pp. 1-8). IEEE.



THANK YOU

Dr. Arti Arya
artiarya@pes.edu
+91 66186629 Extn 6629



Machine Intelligence Module-5 Neuro Genetic Systems



Dr. Arti Arya
Department of Computer Science
artarya@pes.edu
+080-66186629 Extn 6629

Neuro Genetic Systems: Introduction

- Neuro-Genetic Systems are a combination of artificial neural networks and genetic algorithms.
- Two such hybrid systems are
 - 1. Neuro-Genetic System for **weight determination of multi layer feed forward networks**.
 - 2. A technique that artificially evolves neural network topologies using Genetic Algorithms.

Neuro Genetic Systems

- Weights are usually determined through backpropagation learning method.
- In backpropagation of errors the interconnection weights are randomly initialized during network design.
- Recall how with backpropagation, network tries to optimize the weights.
- During training, the actual output is compared with the target output and the error, if any, is backpropagated for adjustments of interconnection weights.
- The error is calculated as

$$E = \frac{1}{2} \sum_i (T O_i - O_i)^2,$$

where $T O_i$ is the target output and O_i is the actual output at the i^{th} output unit.

Neuro Genetic Systems

- During training, the actual output is compared with the target output and the error, if any, is backpropagated for adjustments of interconnection weights.
- The error is calculated as

$$E = \frac{1}{2} \sum_i (T O_i - O_i)^2,$$

where $T O_i$ is the target output and O_i is the actual output at the i^{th} output unit.

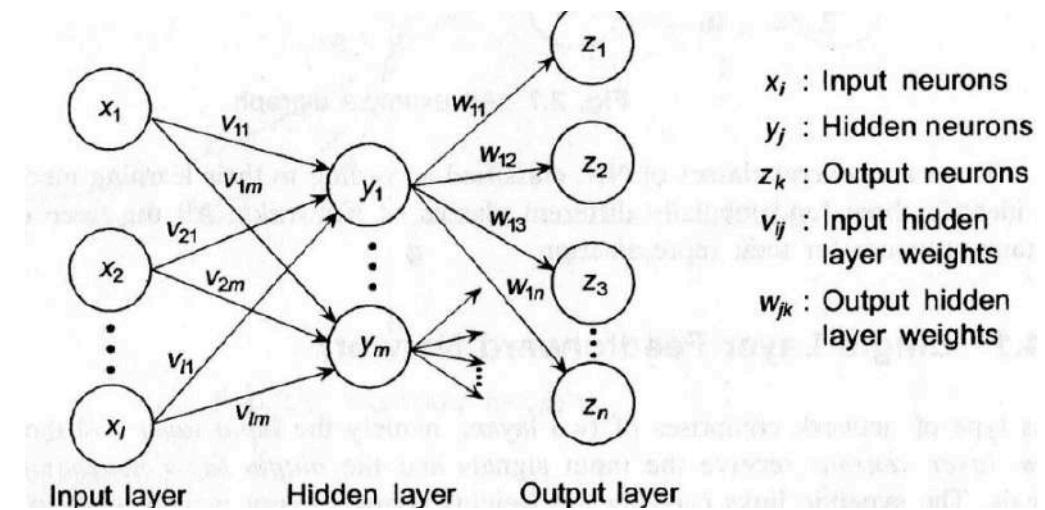


Image source:<https://www.semanticscholar.org/paper/A-Survey-on-Backpropagation-Algorithms-for-Neural-Vora-Yagnik>

Neuro Genetic Systems



- During backpropagation of the error, the network adjusts its weights to return better results in the next iteration.
- This error back propagation follows a gradient descent rule and therefore is vulnerable to the problem of settling down at local minima.
- Another limitation of the gradient descent technique is that it is slow since the number of iterations needed to properly train the network is usually considerably high.

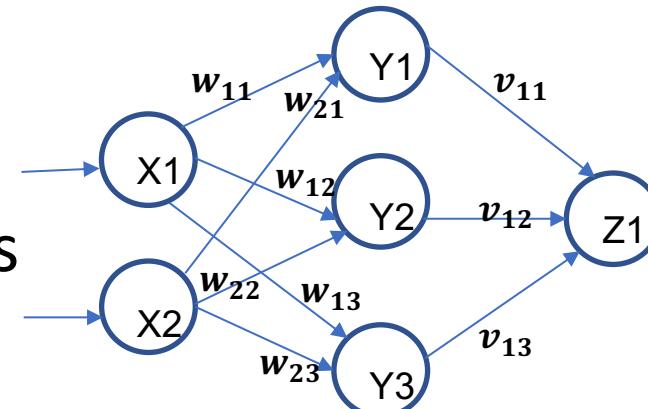
Wt. determination of Multilayered N/w using GA

- Consider a single hidden layer network having
 - $m+n+r$ number of nodes.
 - Total no. of interconnecting weights in the network would be $(mn+rn)$
- Represent each weight by a gene, so a chromosome will be having
$$(mn+rn) \text{ number of genes.}$$
- Structure of gene: $d_1 d_2 d_3 d_4 d_5$ (representing an interconnection weight)
 - Where d's are digits
 - d_1 is used to determine the sign
 - $d_2 d_3 d_4 d_5$, the interconnection weights '+' or '-' depending on whether d_1 is even or odd.
- The magnitude is obtained by dividing $d_2 d_3 d_4 d_5$ by 100
- A chromosome is then a linear array of $(m+r) \times 5$ digits.

Wt. determination of Multilayered N/w using GA

- Consider a 2-3-1 multi-layer network.
- The weights between the input layer and the hidden layer are $w_{11} w_{12} w_{13} w_{21} w_{22}$ and w_{23} .
- And between the hidden layer and the output layer are $v_{11} v_{21} v_{31}$.
- Therefore a chromosome for this network corresponds to an arrangement of weights as given by:

w_{11}	w_{12}	w_{13}	w_{21}	w_{22}	w_{23}	v_{11}	v_{12}	v_{13}
----------	----------	----------	----------	----------	----------	----------	----------	----------



Wt. determination of Multilayered N/w using GA

- In the present case, the chromosome is an array of $(2+1) \times 3 \times 5 = 45$ digits.
- For instance, let
143459076543210765430456713509246809478562589
be a chromosome.
- The mapping between the chromosome , weights and interconnections as shown below

14345	90765	43210	76543	04567	13509	24680	94785	62589
-43.45	-07.65	+32.10	-65.43	+45.67	-35.09	+46.80	-47.85	+25.89

w_{111} w_{112} w_{113} w_{121} w_{122} w_{123} w_{131} w_{132} w_{133}

Wt. determination of Multilayered N/w using GA



- The initial population consists of a set of randomly generated chromosomes .
- Fitness is measured in terms of the error term $E = \frac{1}{2} \sum_i (TO_i - AO_i)^2$.
- In order to compute the error, a chromosome is mapped to its corresponding BPN net .
- The network is then tested by applying the input of a test pair and computing the actual output for the said input .
- This actual output when compared with the target output in $E = \frac{1}{2} \sum_i (TO_i - AO_i)^2$ gives the error for that training pair .

Wt. determination of Multilayered N/w using GA

- The same is computed for every training pair and the average is considered for fitness calculation .
- Since the aim is to minimize the error whereas A is a maximization process, we cannot directly use the error E as fitness measure .
- An obvious way is to take the reciprocal of E as the fitness

$$F = \frac{1}{E}$$

- the rest of the process is usual GA .
- It may be noted that the GA – based learning of multilayer nets does not involve any backpropagation of error.
- The journey towards the *minimum error multilayer network is now controlled by the GA instead of the backpropagation learning method process.*

Neuro Genetic Systems



Neuro Genetic Systems





THANK YOU

Dr. Arti Arya

Department of Computer Science

artarya@pes.edu

+91 9972032451 Extn 029



MACHINE INTELLIGENCE

Swarm Intelligence

Dr. Arti Arya

Department of Computer Science

artiarya@pes.edu

+080-661896629 Extn 6629

MACHINE INTELLIGENCE

Swarm Intelligence

Dr. Arti Arya

Department of Computer Science and Engineering

- A loosely structured collection of interacting agents
 - **Agents:**
 - Individuals that belong to a group (but are not necessarily identical)
 - They contribute to and benefit from the group
 - They can recognize, communicate, and/or interact with each other
 - A swarm is better understood if thought of *as agents exhibiting a collective behavior.*

- An AI technique based on the **collective behavior** which is decentralized, self-organized systems.
- Self organized systems are robust, reliable and simple.
- Example of Self Organized systems:
- People commonly self-organize without a leader. For example, in many cultures people will naturally form a line to implement a system of fairness when waiting for something.
- Self-organization is the *basis for swarm robotics, a technique that involves small robots that cooperate to complete work as opposed to being centrally controlled.*

- Generally made up of **agents** who interact with each other and **the environment**.
- No centralized control structures.
- Based on **group behavior** found in nature.

Examples of Swarms in Nature

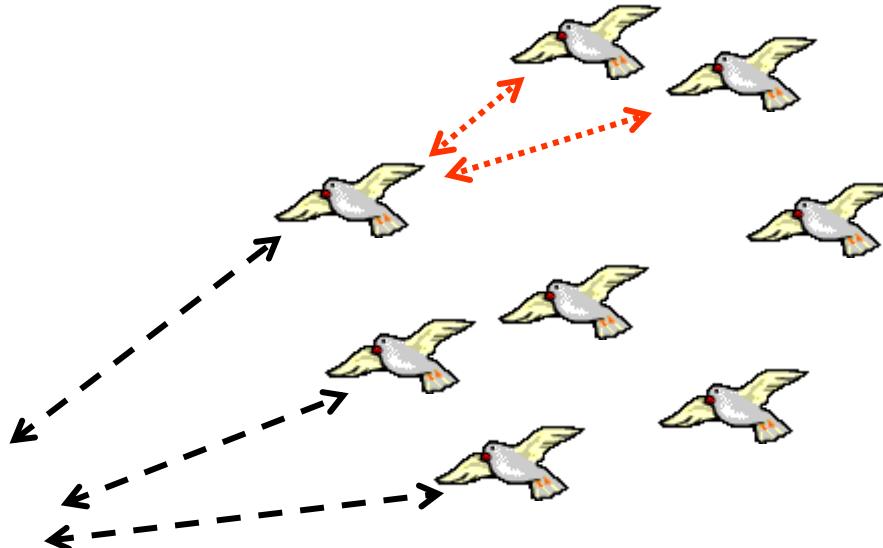
- Classic Example: Swarm of Bees
- Can be extended to other similar systems:
 - Ant colony
 - ✓ Agents: ants
 - Flock of birds
 - ✓ Agents: birds
 - Traffic
 - ✓ Agents: cars
 - Crowd
 - ✓ Agents: humans
 - Immune system
 - ✓ Agents: cells and molecules

Why Insects?

- Insects have a few hundred brain cells.
- However, organized insects have been known for:
 - ✓ Architectural marvels. (*Just check out 9 Wonderful Animals and Insects who are better at Architecture and buildings than humans*)
 - ✓ Complex communication systems.
 - ✓ Resistance to hazards in nature.
- In the 1950's E.O. Wilson observed:
 - ✓ A **single ant acts** (almost) randomly – often leading to its own demise
 - ✓ A **colony of ants** provides food and protection for the entire population

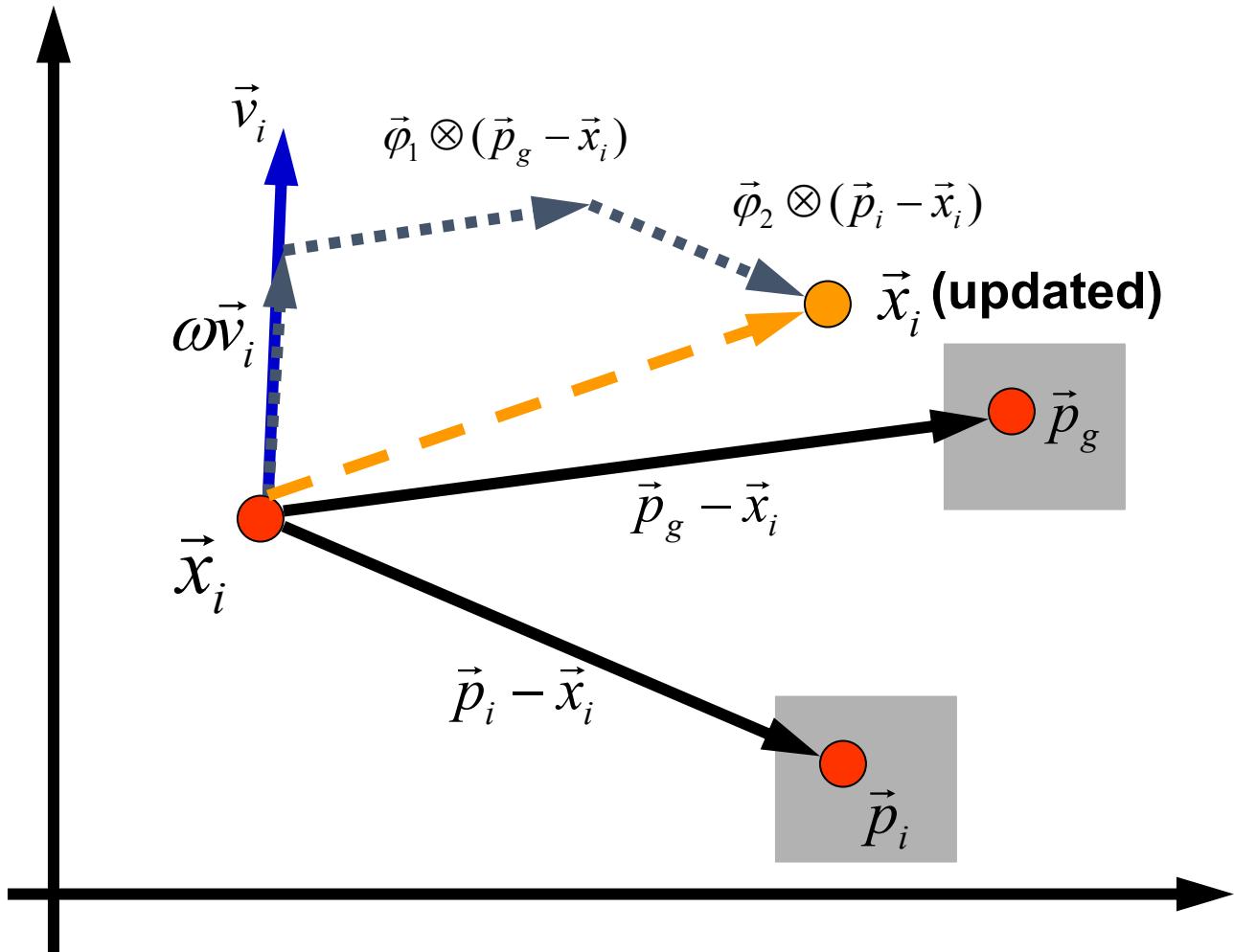
Particle Swarm Optimization- PSO

- As described by the inventors James Kennedy and Russell Eberhart, “**particle swarm algorithm imitates human (or insects) social behaviour.**”
- Individuals interact with one another while learning from their own experience, and gradually the population members move into better regions of the problem space”.



Why named as “**particle**”, not “points”? Both Kennedy and Eberhart felt that velocities and accelerations are more appropriately applied to particles.

- ❖ A population based **stochastic optimization technique**.
- ❖ Searches for an **optimal solution** in the computable search space.
- ❖ Developed in 1995 by Dr. Eberhart and Dr. Kennedy.
- ❖ Inspiration: Swarms of Bees, Flocks of Birds, Schools of Fish



- ✓ Each particle is searching for the optimum.
- ✓ Each particle is *moving* and hence has a *velocity*.
- ✓ Each particle *remembers* the position it was in where it had *its best result* so far (*its personal best*)
- ✓ *But this would not be much good on its own; particles need help in figuring out where to search.*

The Basic Idea

- The particles in the swarm *co-operate*.
- They exchange information about *what they've discovered in the places they have visited*
- The co-operation is very simple. In basic PSO it is like this:
 - ✓ A particle has a *nbd* associated with it.
 - ✓ A particle knows the *fitness* of those in its *nbd*, and uses the *position* of the one with best fitness.
 - ✓ This position is simply used to adjust the particle's velocity.

MACHINE INTELLIGENCE

PSO

- ✓ PSO shares many similarities with **evolutionary computation techniques** such as Genetic Algorithms (GA).
- ✓ The system is **initialized with a population of random solutions** and searches for optima by updating generations.
- ✓ However, unlike GA, PSO has **no evolution operators** such as **crossover and mutation**.
- ✓ In PSO, the potential solutions, called particles, ***fly through the problem space by following the current optimum particles.***

MACHINE INTELLIGENCE

on PSO



- In PSO individuals strive to improve themselves and often achieve this by observing and imitating their neighbors.
- Each PSO individual has the ability to remember.
- PSO has simple algorithms and low overhead
 - ✓ Making it more popular in some circumstances than Genetic/Evolutionary Algorithms.
 - ✓ Has only one operation calculation:
 - *Velocity: a vector of numbers that are added to the position coordinates to move an individual*

Procedure of the Global Version

1. An array of population of particles with random positions and velocities on d dimensions in the problem space are initialized
2. Evaluate the fitness function in d variables for each particle.
3. Compare particle's fitness evaluation with particle's "pbest". If the current value is better than "pbest", then the current value is saved as the "pbest" and the "pbest" location corresponds to the current location in D-dimensional space.
4. Compare fitness evaluation with the population's overall previous best. If the current value is better than the "gbest", then current value is saved as "gbest" to the current particle's array index and value.

Particle Swarm Intelligent Systems

Suppose that the search space is D-dimensional, then the i^{th} **particle of the swarm** can be represented by a D-dimensional vector

$$X_{id} = (x_{i1}, x_{i2}, \dots, x_{iD})^T$$

The **velocity (position change) of this particle**, can be represented by another D-dimensional vector

$$V_{id} = (v_{i1}, v_{i2}, \dots, v_{iD})^T.$$

The **best previously visited position** of the i^{th} particle is denoted as

$$P_{id} = (p_{i1}, p_{i2}, \dots, p_{iD})^T$$

Procedure of the Global Version

Personal influence

5. Modify the **velocity and position** of the particle according to the following equations

$$v_{id}^{t+1} = v_{id}^t + c_1 rand_1 * (p_{id}^t - x_{id}^t) + c_2 rand_2 * (p_{gd}^t - x_{id}^t),$$
$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1}$$

momentum

Social influence

6. If the desired criterion is not met, go to step 2 otherwise stop the process. Usually the desired criterion may be a good fitness function or a maximum number of iterations.

Particle Swarm Intelligent Systems

Defining g as the index of the best particle in the swarm (i.e., the g^{th} particle is the best), and let the superscripts denote the iteration number, then the swarm is manipulated according to the above stated two equations,

- where $d = 1, 2, \dots, D$; $i = 1, 2, \dots, N$, and N is the size of the swarm
- c_1 & c_2 are positive constants, called cognitive and social parameters.
- rand_1 and rand_2 are random numbers, uniformly distributed in $[0, 1]$
- t determines the iteration number

Mechanism to control the velocity

$$\text{if } v_{id} > V_{\max}, \text{ then } v_{id} = V_{\max} \quad \text{if } v_{id} < -V_{\max}, \text{ then } v_{id} = -V_{\max}$$

PSO is able to locate the optimum area faster than the EC techniques, but fails in adjusting its velocity step size to continue the search for a finer grain. The problem is addressed by incorporating a weight parameter for the previous velocity of the particle

$$v_{id}^{t+1} = \varphi(wv_{id}^t + c_1 \text{rand}()^t * (p_{id}^t - x_{id}^t) + c_2 \text{Rand}()^t * (p_{gd}^t - x_{id}^t)),$$

where w is called inertia weight and φ is a constriction factor, which is used, alternatively to w to limit velocity, c_1 and c_2 are two positive constants, called cognitive and social parameter respectively.

The inertia weighted PSO can converge under certain conditions without using Vmax.

Parameters of PSO

- **pbest (p_{id})**: ‘pbest’ is the best position of the particle attained so far and can be considered as the particles memory and one memory slot is allotted to each particle.

- **‘nbest’ (p_{nd}) and ‘gbest’ (p_{gd})**: The best position that neighbours of a particle achieved so far is the ‘nbest’, where ‘gbest’ is the extreme of ‘nbest’, where it takes whole population as the neighbours of each particle.

- The selection of the ‘nbest’ consists of two phases. In the first phase the determination of the neighbourhood come into picture and in the second phase the selection of the ‘nbest’ is done.

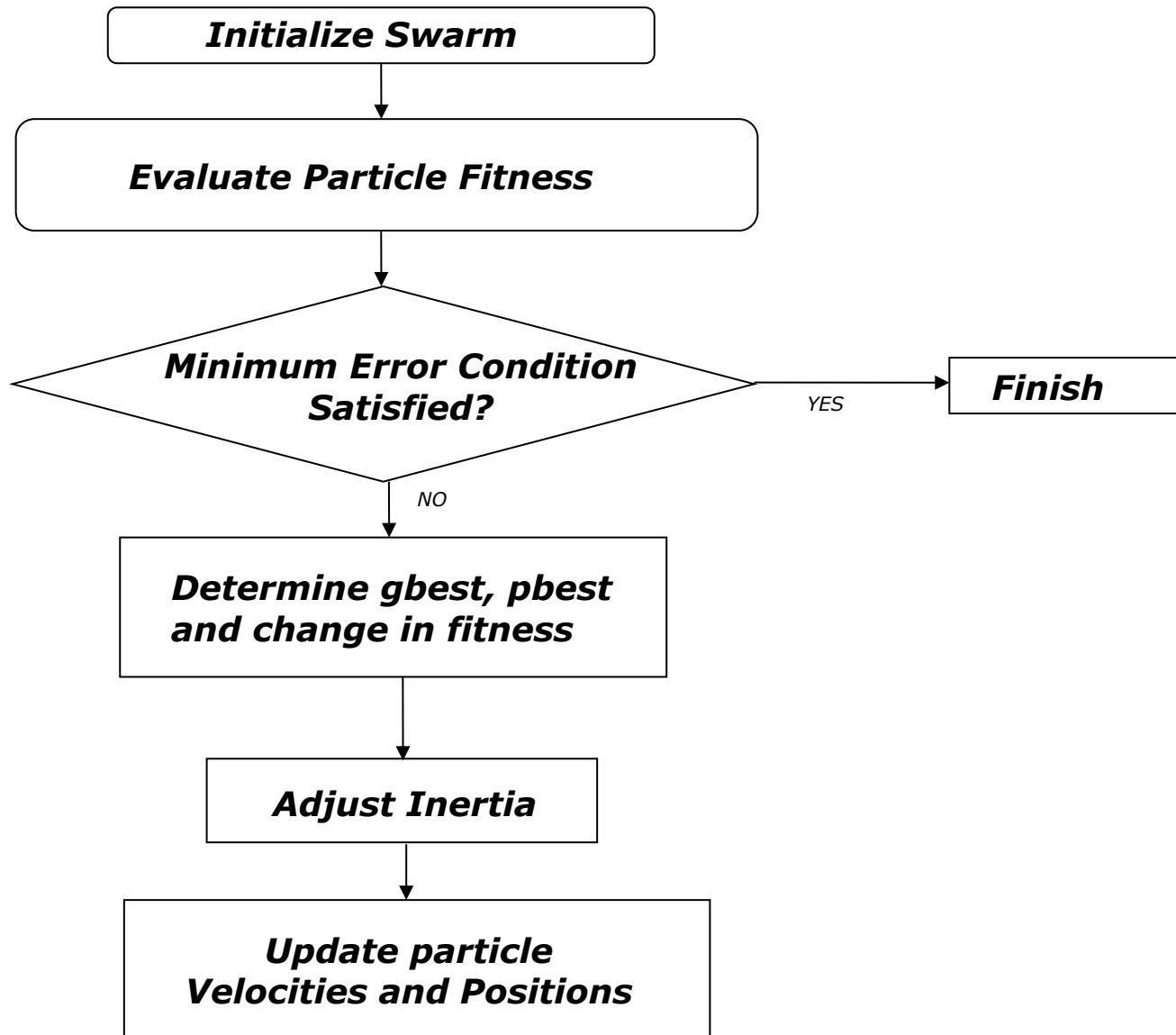
- **Learning factors**: The constants c_1 and c_2 are the learning factors, which represent the weighting of the stochastic acceleration terms that pull each particle towards ‘pbest’ and ‘nbest’ positions.

Parameters of PSO

- **Inertia weight:** The inertia weight is initially set to a constant, but later experimental results suggested having a larger value initially, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions.
- **Constriction factor (ϕ):** Use of the constriction factor ϕ may be necessary to *ensure convergence of PSO*.

$$\varphi = \frac{2}{\left| 2 - \psi - \sqrt{\psi^2 - 4\psi} \right|} \quad \text{where } \psi = c_1 + c_2, \psi > 4$$

Working of PSO



References

1. Some material of these slides are from Prof. Stephany Coffman-Wolph slides on PSO.





THANK YOU

Dr. Arti Arya

Department of Computer Science

artarya@pes.edu

+080-66186629 Extn 6629



Machine Intelligence Example of Particle Swarm Optimization



Dr. Arti Arya
Department of Computer Science
artiarya@pes.edu
+080-66186629 Extn 6629

MACHINE INTELLIGENCE

Example of Particle Swarm Optimization

Dr. Arti Arya

Department of Computer Science and Engineering

A Solved Example of PSO

Q:- Solve the constrained optimization problem:

$$\text{Min } f(x) = x \sin 10\pi x + 1.0$$

where $-1 \leq x \leq 2$.

Solⁿ: Let the position and velocity of i^{th} particle in the search space are represented as vectors:

$$X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$$

$$V_i = (v_{i1}, v_{i2}, \dots, v_{iD}) \text{ resp.}$$

(D is the no. of variables in the function to be minimized.)

PSO Example contd....

Velocity & position of a particle is updated by

$$v_{ij}^{t+1} = w v_{ij}^t + c_1 \text{rand}_1(p_{ij}^t - x_{ij}^t) + c_2 \text{rand}_2(p_{gj}^t - x_{ij}^t)$$

$$w = \frac{(w_{\min} - w_{\max})(i_{\text{tu}} - 1)}{(i_{\text{tumax}} - 1)} + w_{\max}$$

PSO Example contd....

$$\& \quad x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1}$$

where $i = \{1, 2, \dots, N\}$
 $j = \{1, 2, \dots, D\}$

Step 1: Initialize PSO control parameters

Assumption:

particles = 5

Let $C_1 = 2, C_2 = 2$

$w_{max} = 0.9, w_{min} = 0.2$

PSO Example contd....

Given # variables = 1 , upper limit = 2,
lower limit = -1 , iter = 1, max iter = 100.

Step 2: Initial Generation of Population & Velocity

Generate random nos. for one variable $x_{ij}(x_{i1})$ with upper & lower limit, where i is the no. of particles, j is the no. of variables.
 $i=1,2,3,4,5$, $j=1$.

Velocity $v_{ij}(v_{i1})$ is also generated randomly within limits.

PSO Example contd....



$$X_{i1} = \begin{cases} -1.00000 & \dots x_{11} \\ 2.00000 & \dots x_{21} \\ -0.2671582 & \dots x_{31} \\ 1.28415387 & \dots x_{41} \\ 0.188660 & \dots x_{51} \end{cases}$$

$$V_{i1} = \begin{cases} 3.94634 & \dots v_{11} \\ 4.00000 & \dots v_{21} \\ 1.67851605 & \dots v_{31} \\ -4.0000 & \dots v_{41} \\ 2.7102747 & \dots v_{51} \end{cases}$$

PSO Example contd....

Step 3: Evaluating fitness of the population.

Fitness value is computed by

$$\text{Fitness} = \begin{cases} \frac{1}{\text{obj} + 1} & \text{when } \text{obj} \geq 0 \\ 1 + |\text{obj}| & \text{when } \text{obj} < 0. \end{cases}$$

$$\text{Obj} = x \sin 10\pi x + 1.0$$

$$= \begin{cases} 1.00000 \\ 1.00000 \\ 1.2294 \\ 1.615081 \\ 0.934201 \end{cases}$$

PSO Example contd....



$$\text{Fitness} = \left\{ \begin{array}{l} 0.500025001 \\ 0.500025 \\ 0.44855 \\ 0.382397 \\ 0.51700 \end{array} \right.$$

Step 4:- Memorizes g_{best} value

g_{best} = Corresponding to $x_{51} = 0.1886$

g_{fit} = 0.51700

PSO Example contd....

Step 5 :- Set iteration = 1.

Step 6 :- Updating particle's velocity.

Weighting factor: for 1st iteration $w = [0.9]$

The corresponding velocity for the weighting factor

The inertial weighting factor w

$$w = \frac{(w_{\min} - w_{\max})(i_{\text{te}} - 1)}{(i_{\text{te}}_{\max} - 1)} + w_{\max}$$

$$= \frac{(0.2 - 0.9)(1 - 1)}{100 - 1} + 0.9 = 0.9$$

PSO Example contd....

Let $r_{\text{rand}_1} = 0.5498602$, $r_{\text{rand}_2} = 0.144954$.

$$\begin{aligned}\therefore V_{11} &= (0.9 \times 3.9463) + 2(0.5498602) \times (1 - (-1)) \\ &\quad + 2(0.144954) + (0.1887 - (-1)) \\ &= 3.8962\end{aligned}$$

11y₁

$$\begin{aligned}V_{21} &= (0.9 \times 4) + 2(0.5498602)(2 - 2) + \\ &\quad 2(0.144954)(0.1887 - 2) \\ &= 3.074\end{aligned}$$

PSO Example contd....

$$V_{31} = 1.6427$$

$$V_{41} = -3.9174$$

$$V_{51} = 2.43924$$

$$\therefore V = \begin{cases} 3.8962 \\ 3.074 \\ 1.6427 \\ -3.9174 \\ 2.43924 \end{cases}$$

In case, the velocity violates the limits,
set the velocity value = max. value.

PSO Example contd....

Step 7: Modification of Particle Position.

$$\therefore X_{11} = -1 + 3.8963 \\ = 2.8963$$

If violates the upper limit $\therefore \hat{x}_{11} = 2$

My, $X_{21} = 2 + 3.074 = 3.34$
 \therefore Set $X_{21} = 2$

My, $X_{31} = 1.711403, X_{41} = -1.000, X_{51} = 2.000$.

PSO Example contd....

Step 8:- Evaluating of fitness of the modified position of particle.

$$Obj = \begin{cases} 1 \\ 0.39989 \\ 0.999 \\ 0.9999 \end{cases}$$

$$\text{fitness} = \begin{cases} 0.5 \\ 0.5 \\ 0.714 \\ 0.5002 \\ 0.5002 \end{cases}$$

PSO Example contd....

Step 9:- Memorize g_{best} & p_{best} .

$$g_{best} = 1.7114$$

$$g_{fit} = 0.714$$

$$p_{best} = \begin{cases} 2.000 \\ 2.000 \\ 1.7114 \\ -1.000 \\ 2.000 \end{cases}$$

Step 10:- Memorize the best result so far
& increment $\underline{\underline{iter = iter + 1}}$.

PSO Example contd....

Step 8:- Evaluating of fitness of the modified position of particle.

$$Obj = \begin{cases} 1 \\ 0.39989 \\ 0.999 \\ 0.9999 \end{cases}$$

$$\text{fitness} = \begin{cases} 0.5 \\ 0.5 \\ 0.714 \\ 0.5002 \\ 0.5002 \end{cases}$$

PSO Example contd....

Step 8:- Evaluating of fitness of the modified position of particle.

$$Obj = \begin{cases} 1 \\ 0.39989 \\ 0.999 \\ 0.9999 \end{cases}$$

$$\text{fitness} = \begin{cases} 0.5 \\ 0.5 \\ 0.714 \\ 0.5002 \\ 0.5002 \end{cases}$$



THANK YOU

Dr. Arti Arya

Department of Computer Science

artarya@pes.edu

080-66186629 Extn 6629