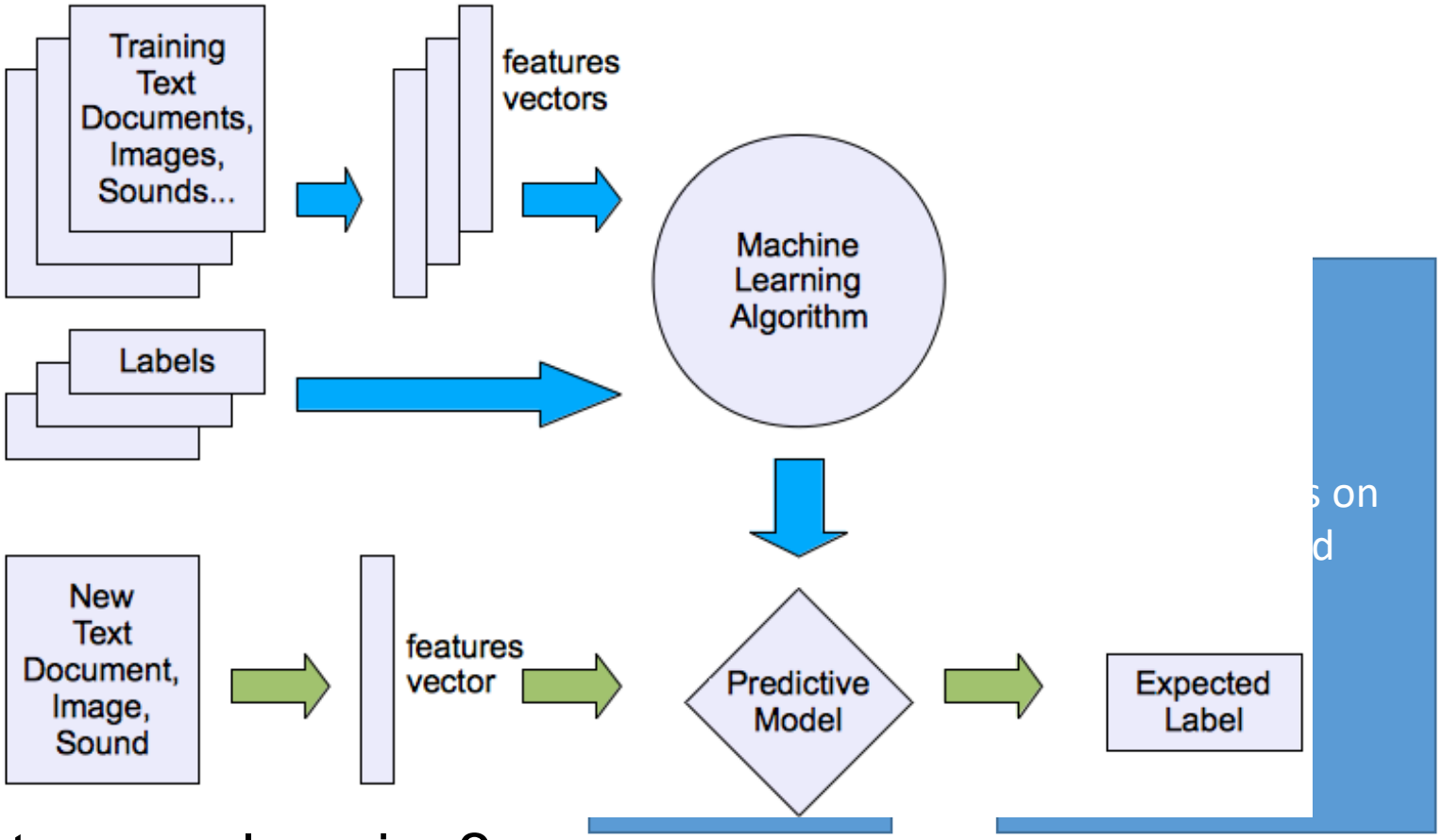
TACC Summer Machine Learning Institute: Unsupervised Learning

Weijia Xu
Research Scientist, Group Manager
Scalable Computational Intelligence
Texas Advanced Computing Center
University of Texas at Austin

TACC

Unsupervised Learning



What are we learning?

Unsupervised Learning

- Learning “structure embedded within the data”
  - What normally happens.
  - Better representations of the data
    - Features
    - Groups.

- Outputs are not a predative models but characteristic of data collection.
  - New may mapped to the structure.

- Common techniques/examples :
  - Clustering: membership based on instance similarity
  - Density estimation: Gaussian mixture model
  - Reduction/Summarization: alternative representations of the data
  - Association: connections/inferences/cooccurrence among instances

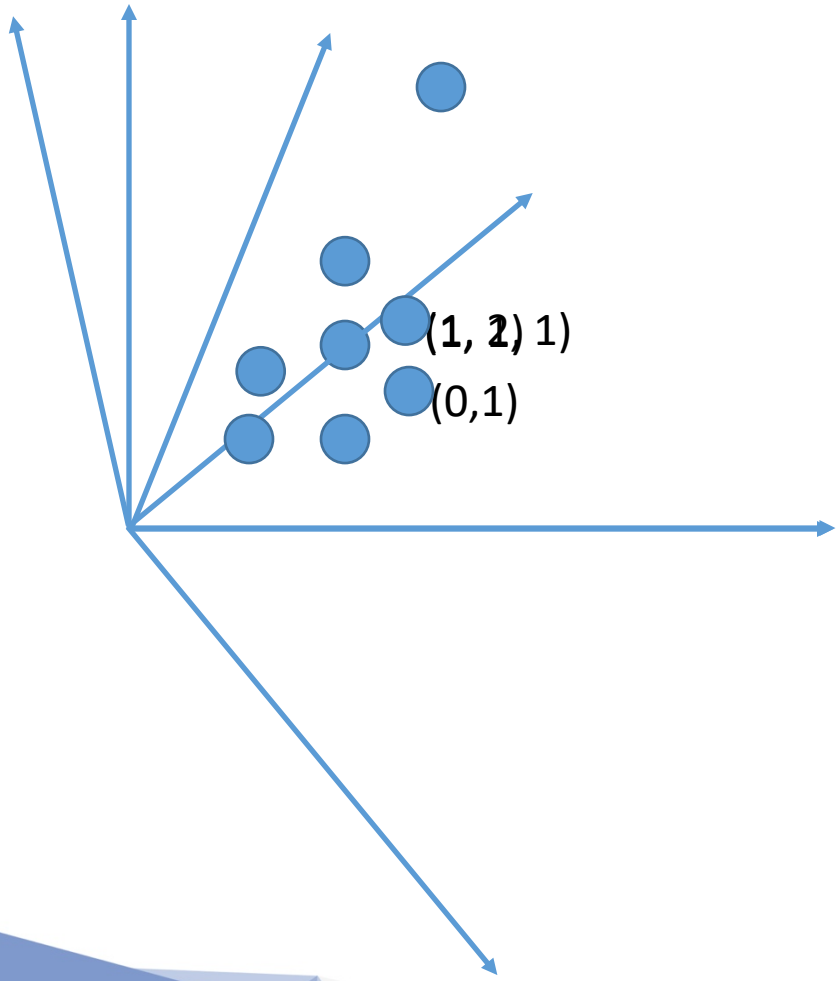
Outline

- Data Representations
  - Principle Component Analysis
  - Manifold learning
- Data Clusters
  - K-Means
  - Hierarchical Clustering
- Density of data
  - Gaussian Mixture Model
  - Density based clustering
- Anomaly Detection

Learning Data Representation

Data Representation

How to describe this?



--- a solid blue circle.

But we want something numeric.

We would like to consider data as points from "some dimensional space". So they can be described as

$$f(v_1, v_2, v_3, \ldots v_n)$$

Principle Component Analysis?

- A data transformation method:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

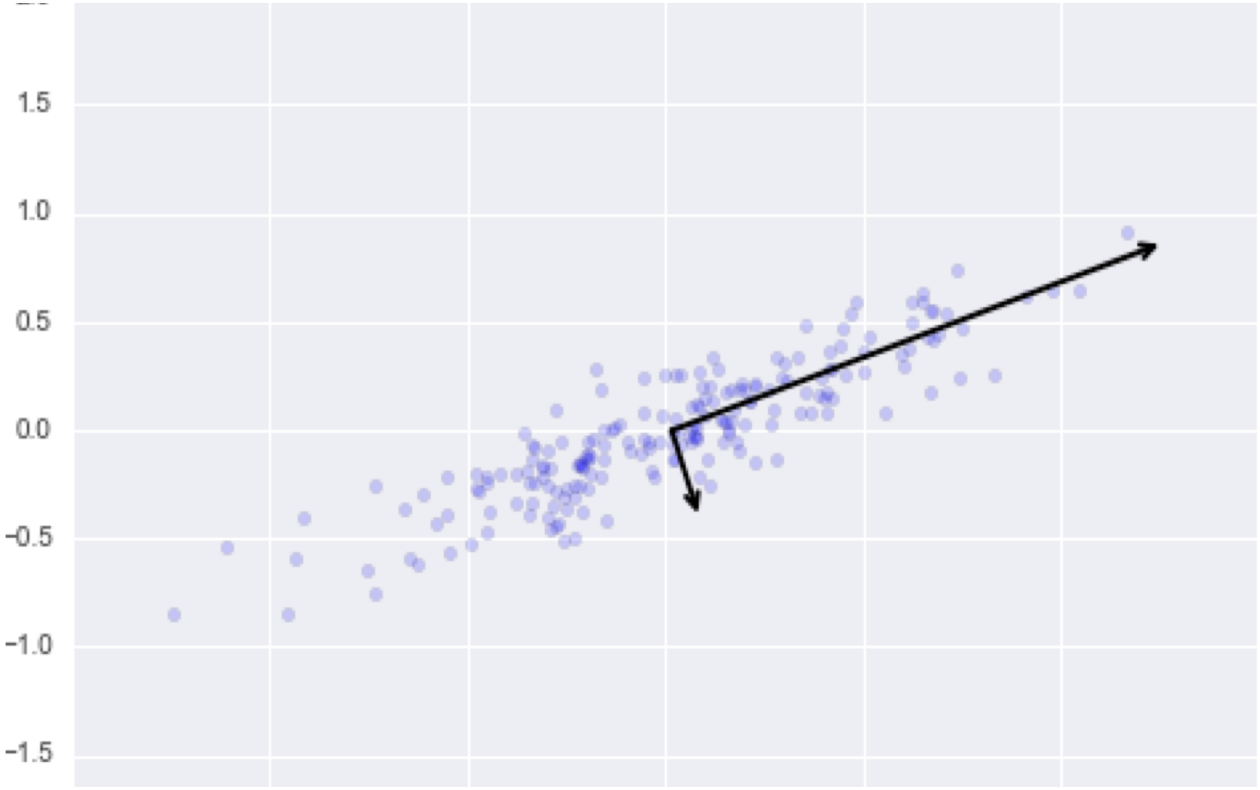
- Potential usages
  - Alternative data representation
  - Data visualization
  - Noise filtering
  - Feature extraction

Why PCA?

- In a nutshell:
  - Compute a covariance matrix over input dimensions.
  - Compute Eigenvalues and Eigenmatrix of the covariance matrix to identify **principal components**
  - Project data with principal components.

- Principal Components
  - A set of new variables constructed as linear combinations initial variables.
  - In a way, the new variables are **uncorrelated** and **unevenly**.

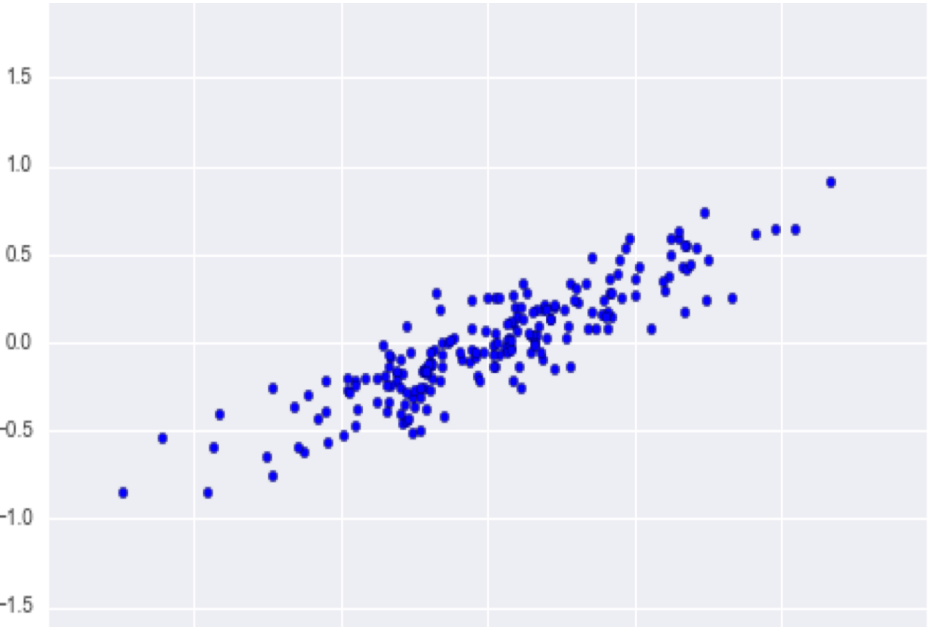
Change Data Representation



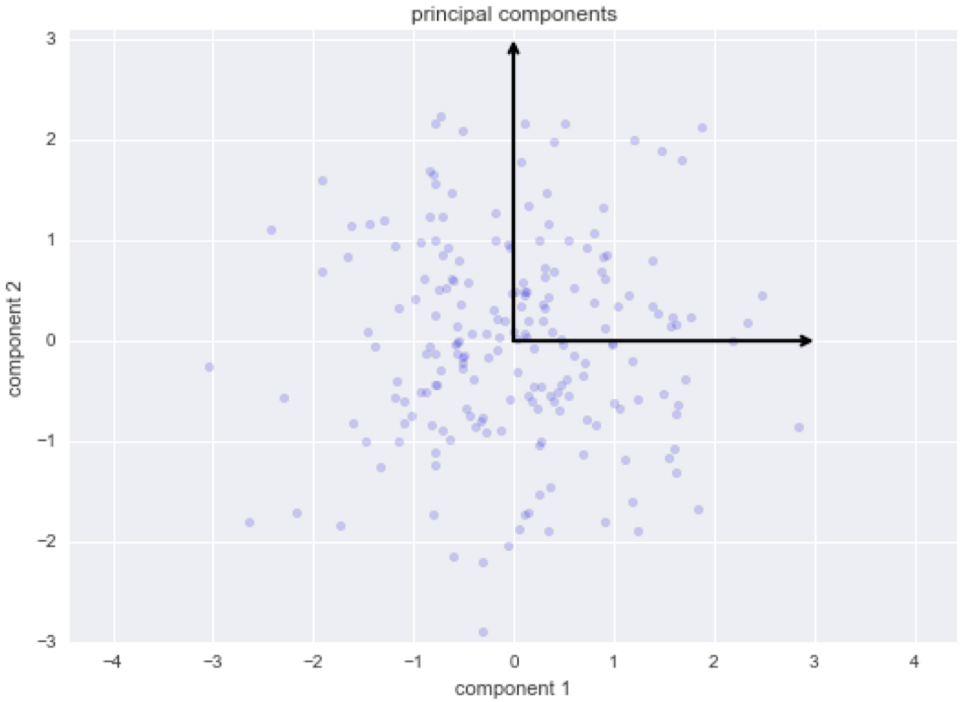
First two principle components

Change Data Representation

Original Coordinate Spaces



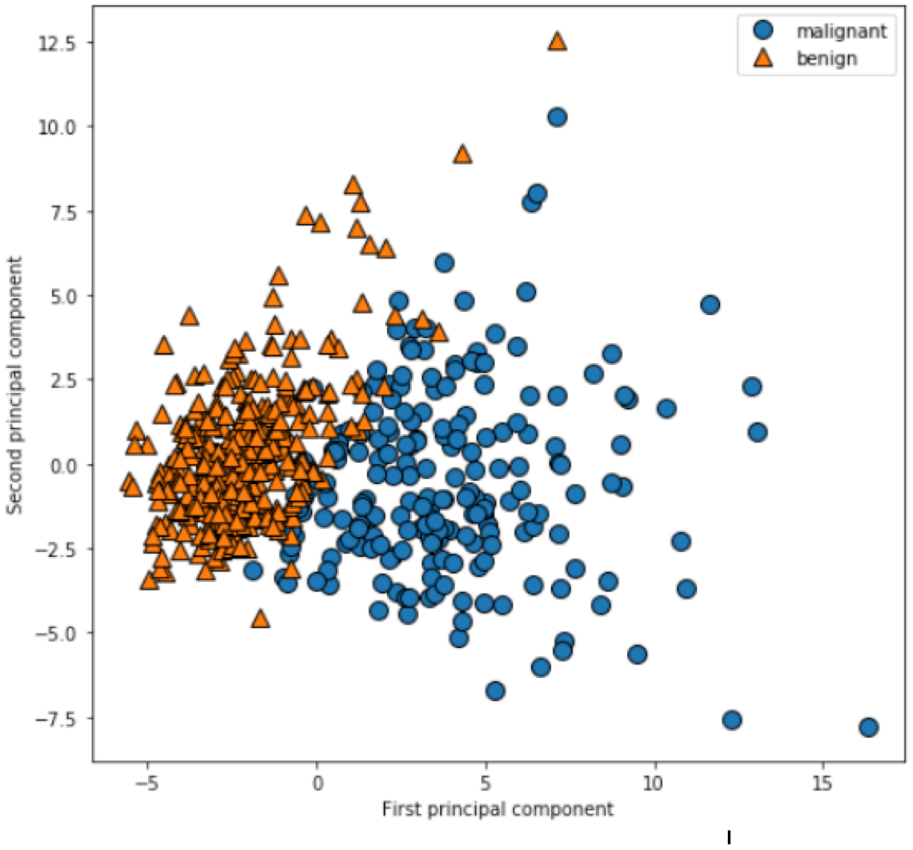
First two component space



Visualizing High Dimensional Data

- A common usage of PCA is to visualize high dimensional data using the first two components.

- Breast Cancer Data:
  - 2 classes
  - 569 observation
  - 30 features

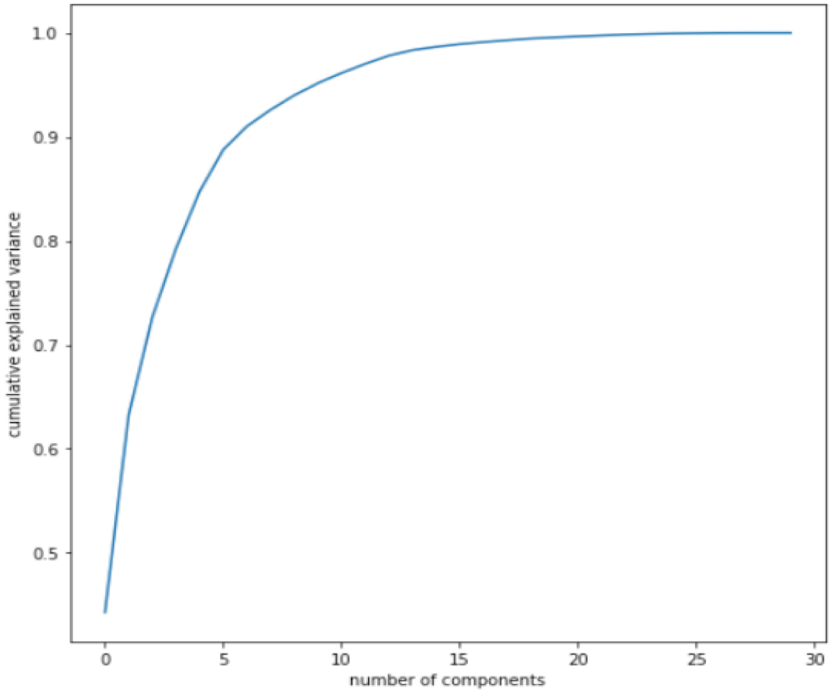


PCA as Noise Filters

- PCA provide a way to create alternative view of the data.

- When choose only *m* components from decomposition results, we lose some information.

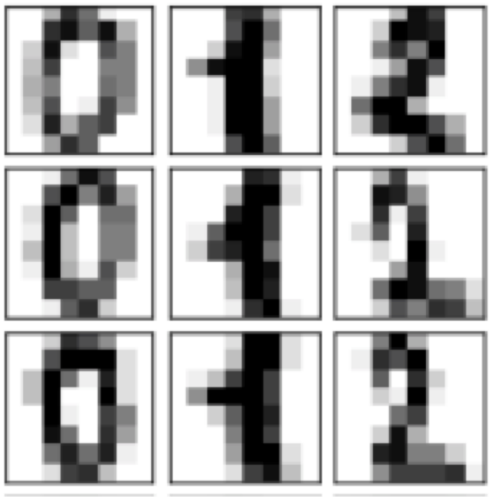
$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$$



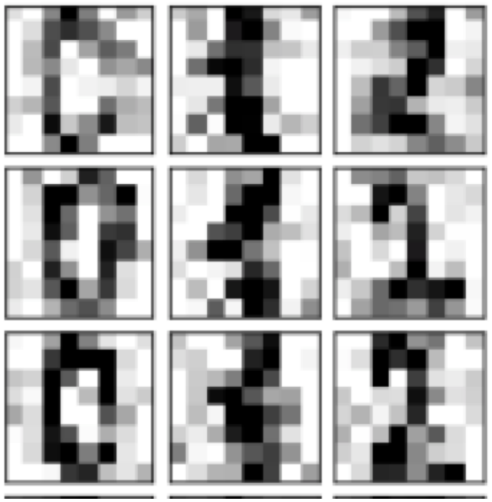
PCA as Noise Filtering

- Components with smaller eigenvalue means less variance
- Components with variance much larger value are less sensitive to the noise.
- Hence, by selecting fewer components to reconstruct the data, we can remove noise points.

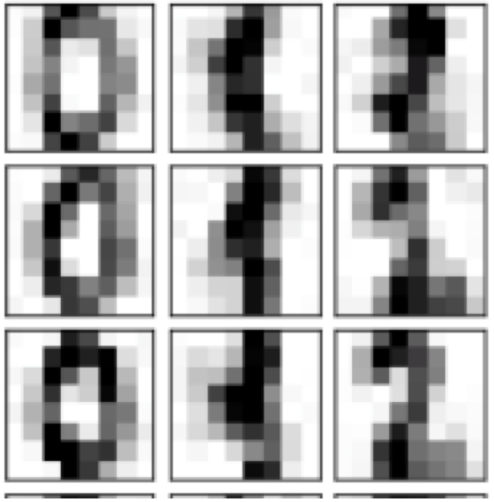
Original Data



Noise Data



Filtered Data



PCA Summary

- PCA is more than just for dimensionality reduction.
- Easy to compute and interpretable
- Sensitive to outliers
  - Randomized PCA
  - SparsePCA
- Works best with linear relationships within the data?

Manifold Learning

- Describe data as low-dimensional manifolds embedded in high-dimensional spaces

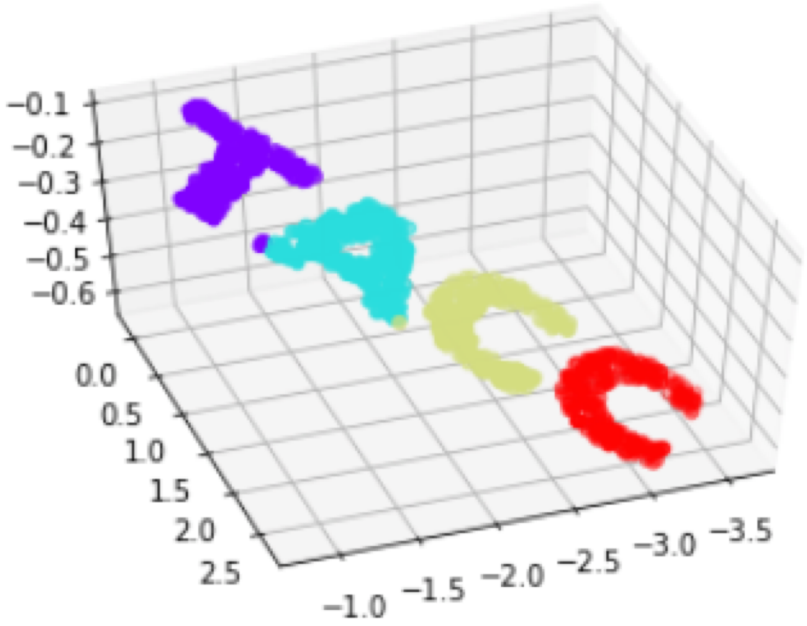
- Manifold,
  - Viewing n-dimensional space through k dimensional space. ($k < n$)
  - e.g. a three dimensional space as a surface.

- Common methods:
  - MDS, Multidimensional embedding
  - LLE, locally linear embedding

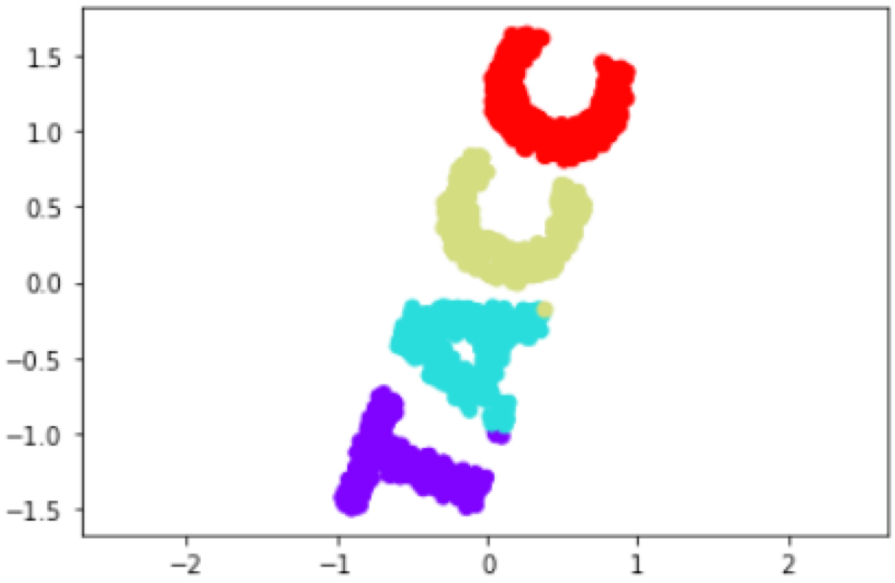
Multidimensional Scaling (MDS)

- Consider input data are represented in n-dimensions.
- If we care about the similarity among the data the most, we can preserve it in a lower dimension.
- Given a distance matrix between points, MDS recovers a representation of the data in *d* dimensions such that pairwise distance in d-dimensions approximates the distance matrix.
- works well with linear transformation

Data in 3D

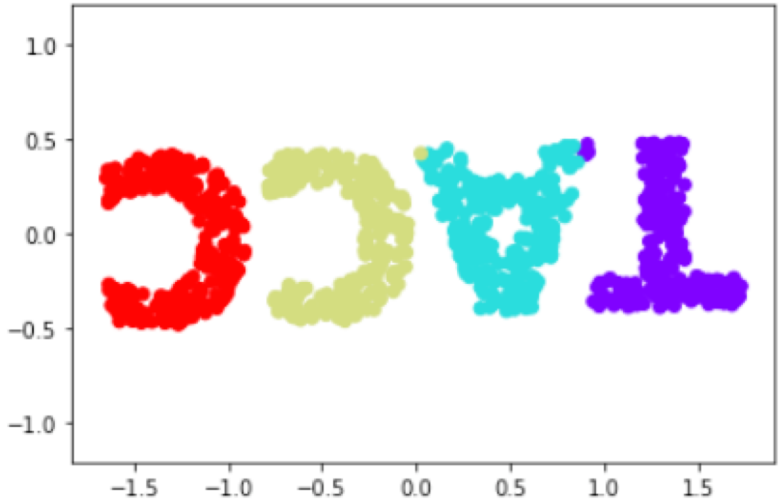


MDS projection to 2D



PCA vs MDS

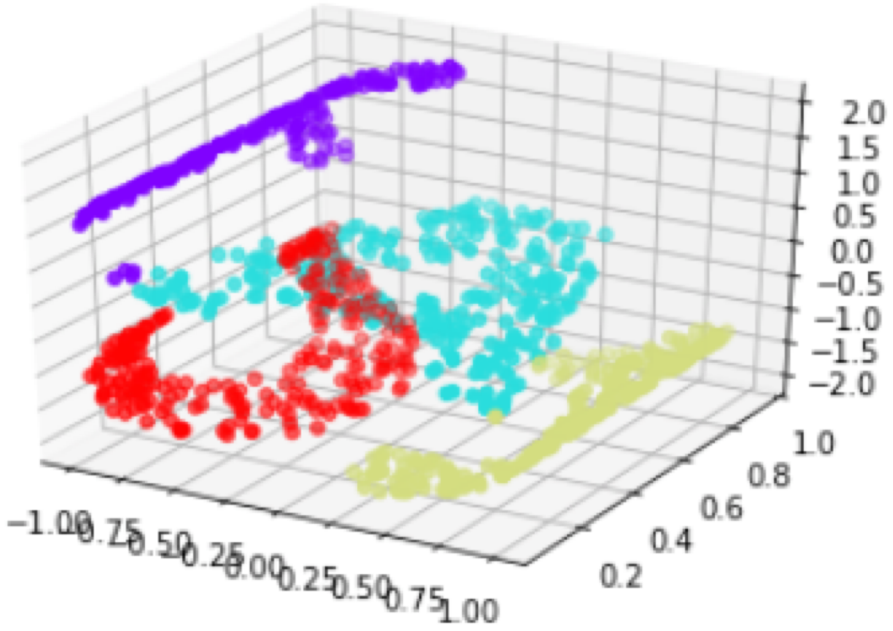
- PCA projection are also good.



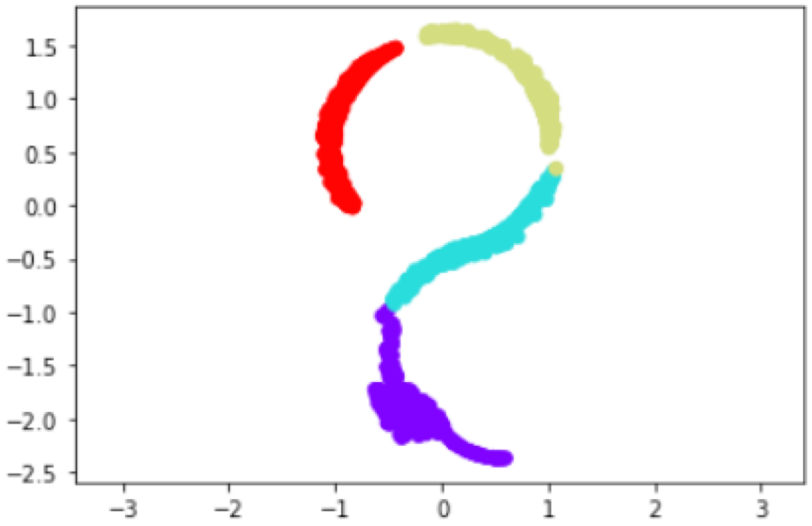
- Both methods can project data to lower dimensions with different idea
  - PCA is to identify and preserve variance
  - MDS is to preserve the distance between points.
  - Both works well with linear transformation.

Non-linear transformation

Data in 3D



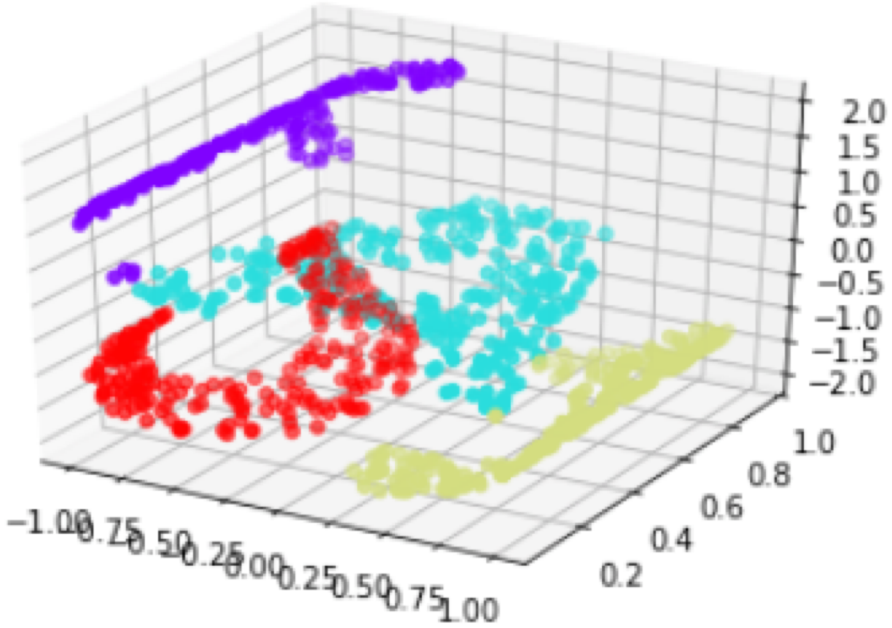
MDS Projection



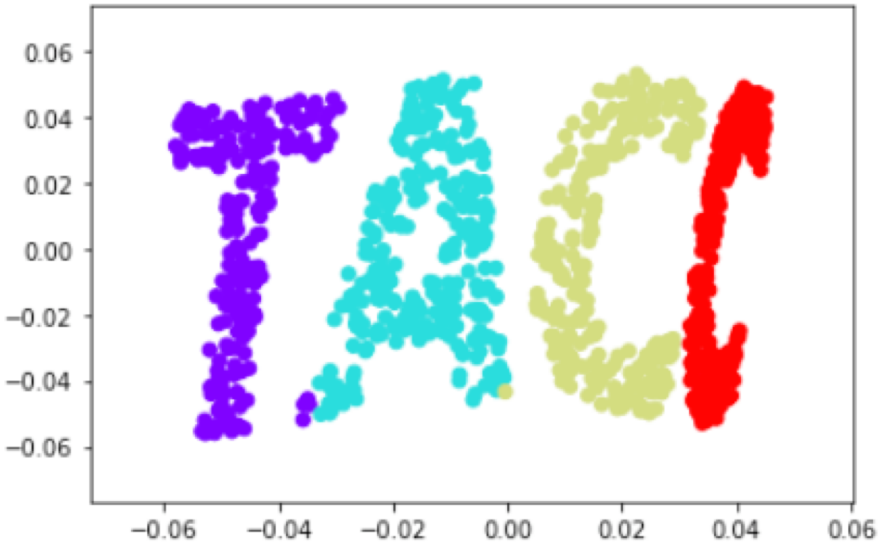
Locally Linear Embedding

- Both PCA and MDS will fail with non-linear transformation.
  - The problem is caused by trying to preserve the global distance matrix.
- LLE Not trying to preserve all the distance, but local distances
- LLE tries to preserve “neighborhood” by only considering n nearest neighbors.

Data in 3D



LLE Projection



Manifold Learning Summary

- Sensitive to data quality
  - Noise data
  - Missing data
- Require more tuning
  - LLE sensitive to the choice of neighbors.
  - Optimal output dimensions
- High computational costs, requires pair-wise distance computation.
- Other methods:
  - Isomap, Isometric mapping
  - t-SNE, T-distributed stochastic neighbor embedding

http://scikit-learn.org/stable/modules/manifold.html

Learning Data Representation Summary

- General goal: identify alternative data representations
  - i.e. transforming data from N dimensional space to M dimensional space.

- How?
  - Based on variance among data points.
    - PCA.
  - Based on distances among data points
    - MDS, preserving global distances between point
    - LLE, preserving distances within local neighborhood.

- Commonly used for:
  - Reduce dimensionality.
  - Visualizing data for exploratory data analysis.
  - Improving data quality, denoise

Hands on session

1. Log on to the Visualization Portal with your TACC account.
2. Launch a Jupyter Notebook job from the Visualization Portal.

   Use reservation "**ML_Institute_day3**" on **Frontera**.

   Use "**dev**" queue

Wait for Jupyter notebook to start

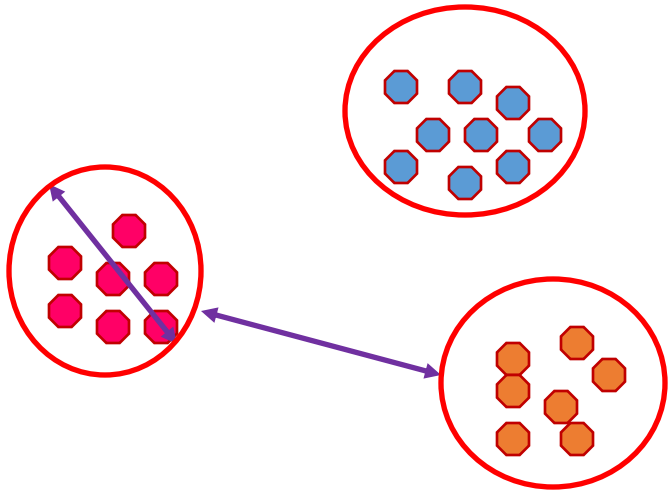
3. Open a new terminal from your Jupyter session.

4. Copy example notebook over:
   cp /work/00791/xwj/DMS/TACC_SSI_2021/unsupervised.ipynb .
5. Open 'unsupervised.ipyb'

Learning Structure

Clustering Analysis

- Given a set of data, can we automatically put them in to groups?
- Based on what?
  - Distance among data
    - Inter-cluster distance
    - Intra-cluster distance
    - K-means, bi-secting kmeans
  - Statistics distribution
    - Fit data based on statistics distribution
    - LDA, GMM



K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

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

(EM) Expectation-Maximization

- A generic algorithm with variety context

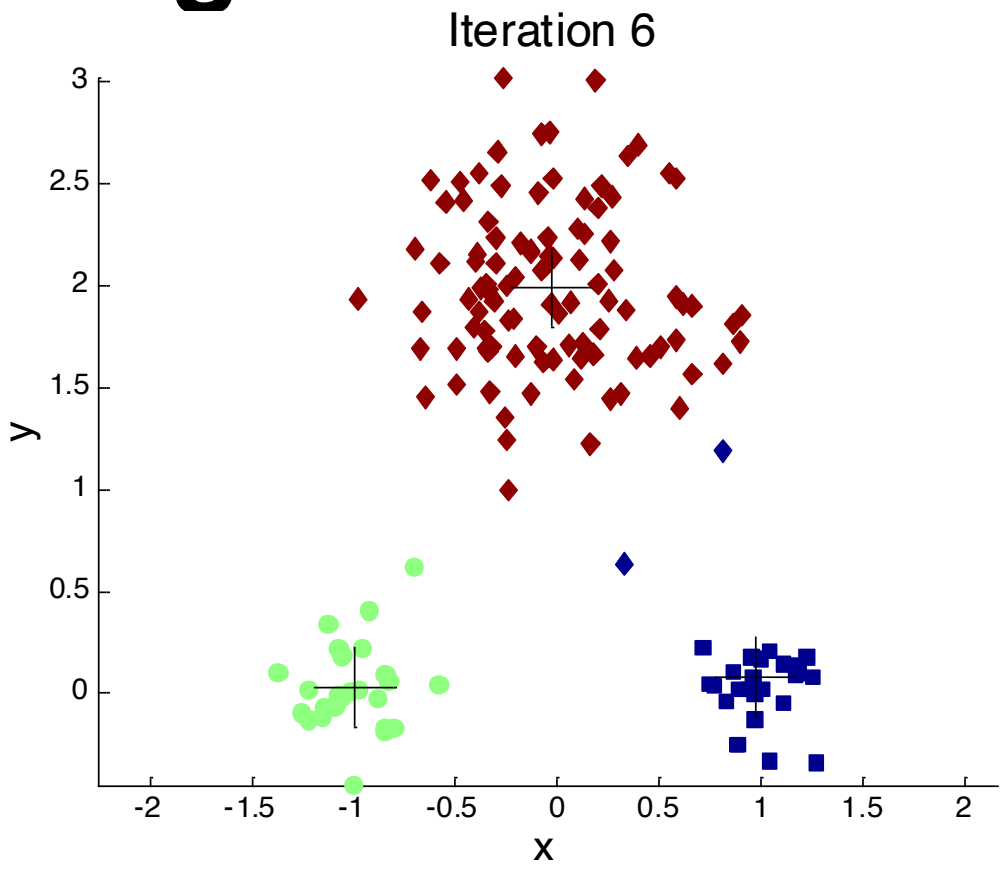
- Iterative through two steps:
  - Expectation Step (E) : Updating the expectations
  - Maximization Step (M): maximizing fitness function.

- K-Means:
  - Expectation: which cluster each points belong to.

  - E: Assign points to nearest cluster center,
  - M: Update the cluster centers.

An Example of k-means Clustering

K=3

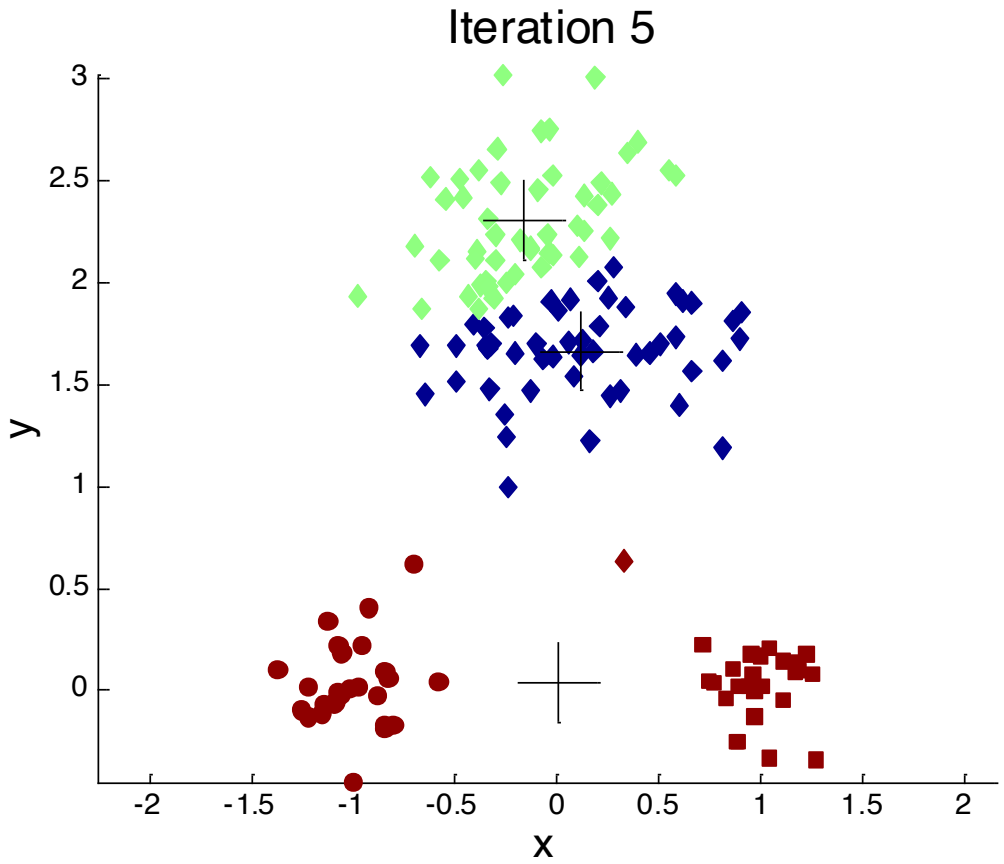


Examples are from Tan, Steinbach, Kumar *Introduction to Data Mining*

More about on K-means

- Will the algorithm always converge?
  - Convergence, algorithm reaches its goal, e.g. error within a threshold
  - K-means will usually converge fast for common similarity measures.
    - Euclidean distance
    - Mahalanobis distance
    - Bregman divergence
- Will the algorithm guarantee to find the most optimal solutions?
- Will the algorithm always find the same solution.

Initial Centroids Selection



Select Best Initial Centroids?

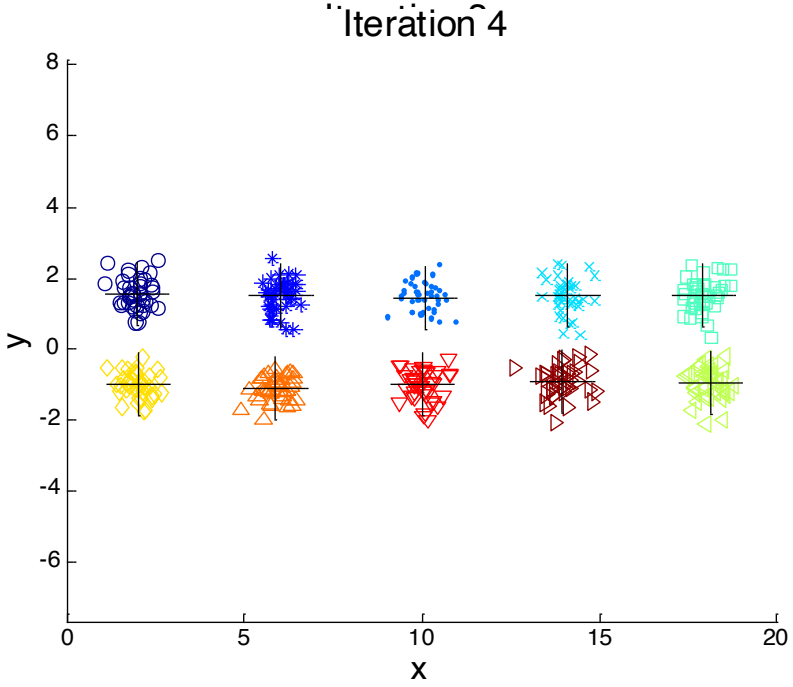
If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.

- Chance is relatively small when K is large
- If clusters are the same size, n, then

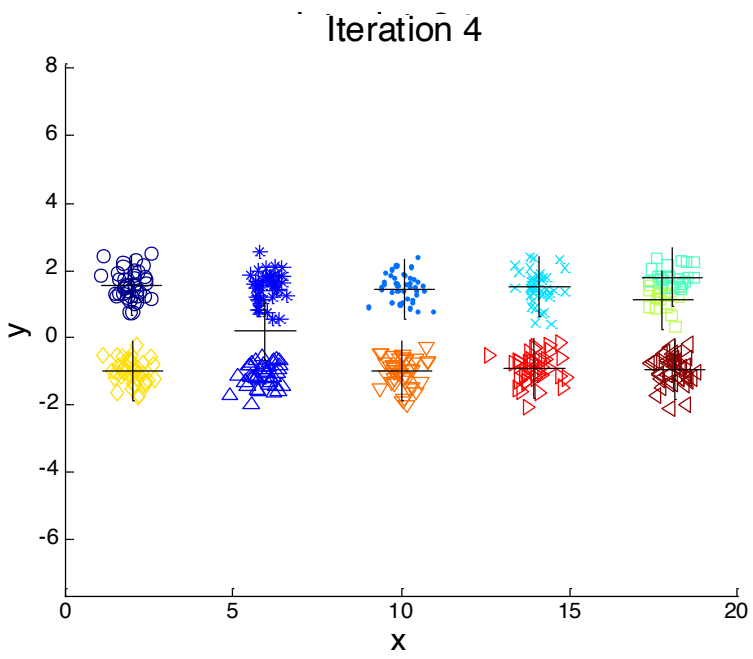
$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- 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
- Consider an example of five pairs of clusters

10 Clusters Example

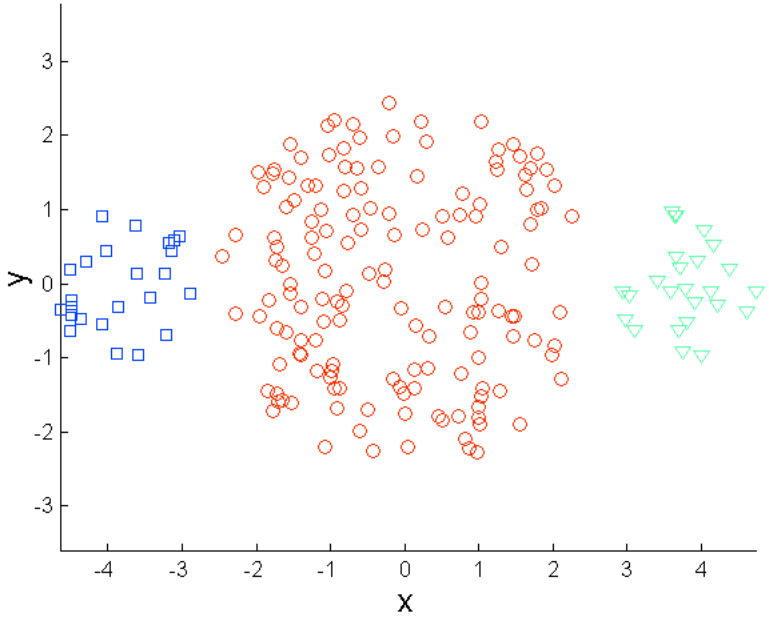


Starting with two initial centroids in one cluster of each pair of clusters

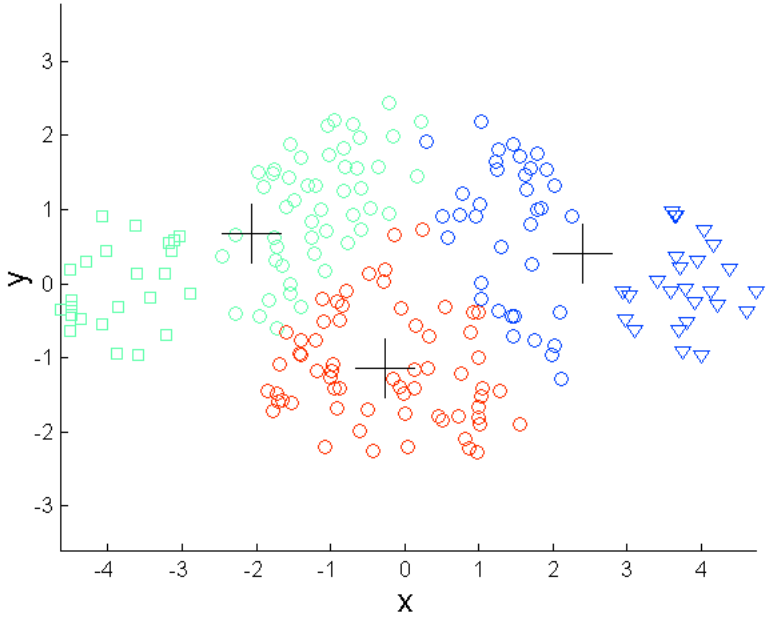


Starting with some pairs of clusters having three initial centroids, while other have only one.

Impact of Cluster Size

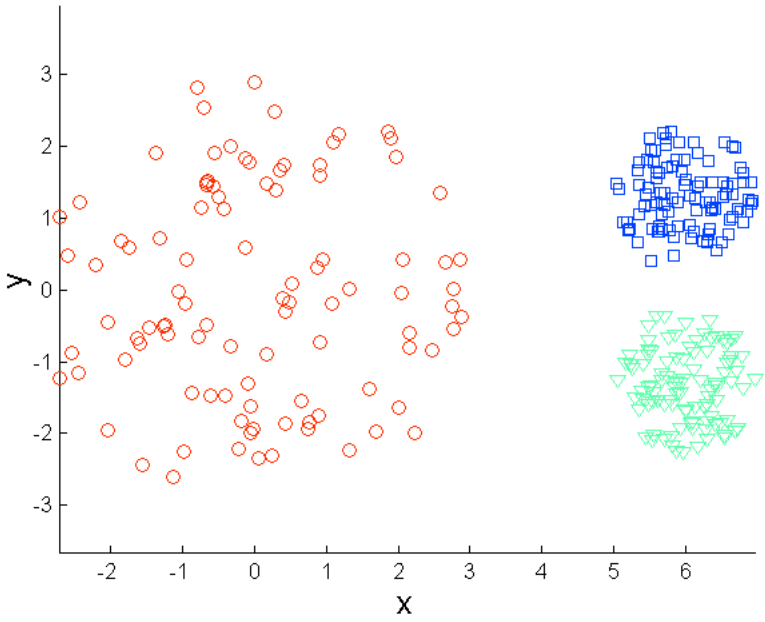


Original Points

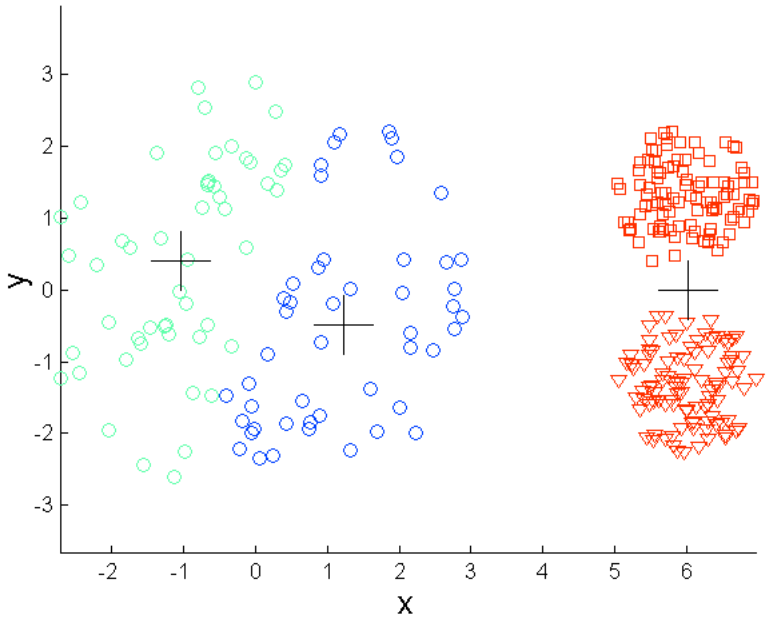


K-means (3 Clusters)

Impact of Cluster Density

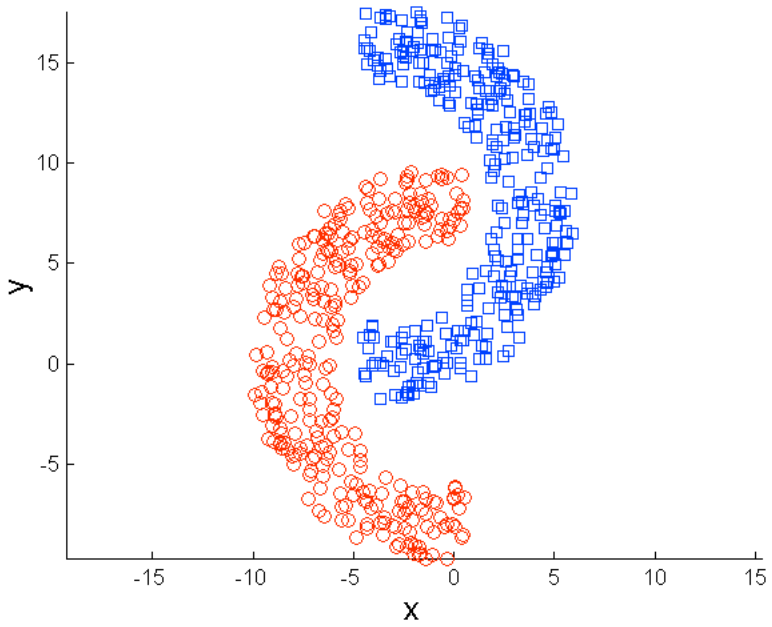


Original Points

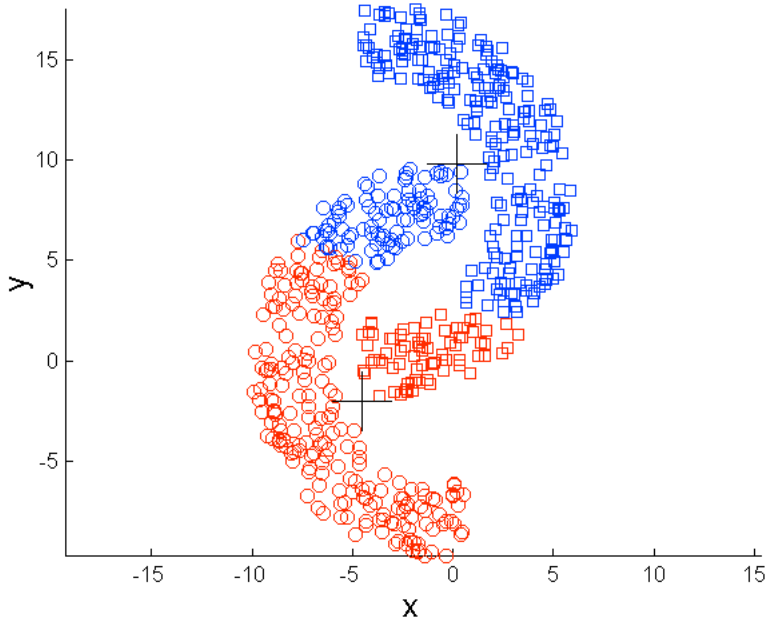


K-means (3 Clusters)

Impact of Cluster Shapes

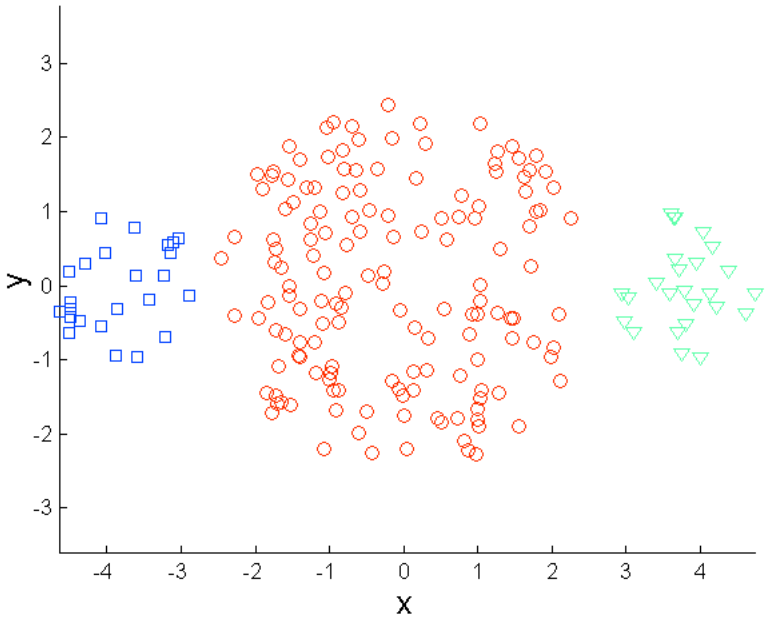


Original Points

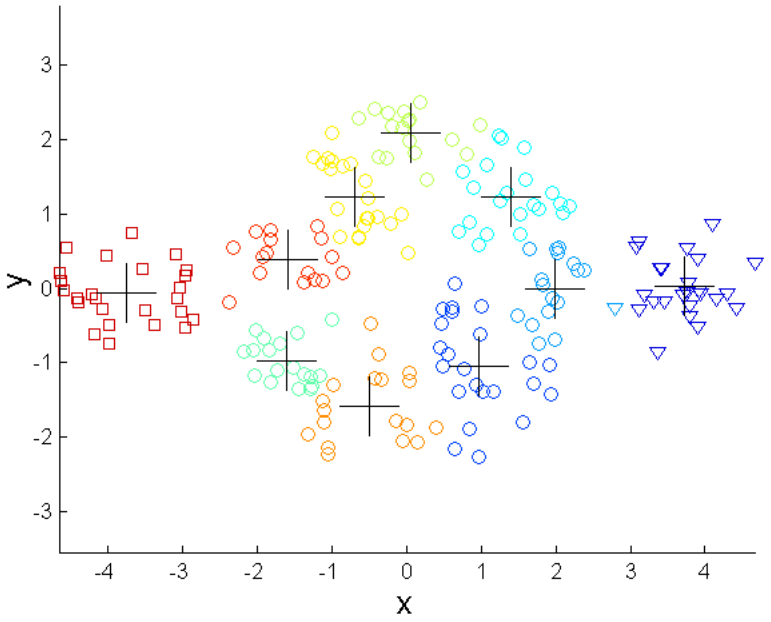


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points



K-means Clusters

- Multiple runs
- More clusters

BiSecting K-means

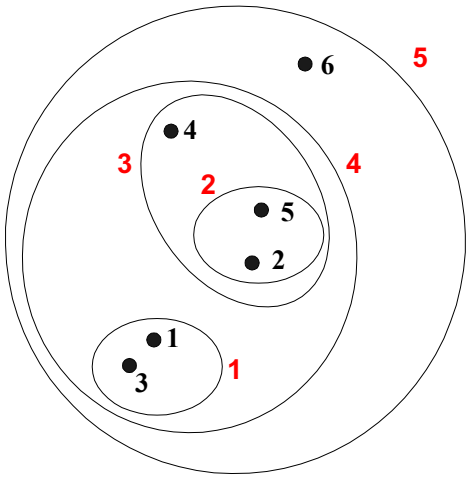
- A top down clustering
  - Starts with one cluster of all data points
  - Split clusters at each level.
- Generates an hierarchical structure of data
- Inherent the same randomness of k-means.

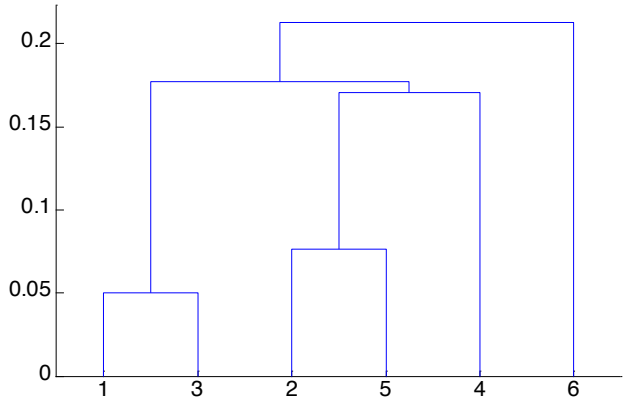
Generating Hierarchical Clusters

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

Two main types of hierarchical clustering

- Agglomerative (bottom-up):
  - Start with the points as individual clusters
  - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
- Divisive (top-down):
  - Start with one, all-inclusive cluster
  - At each step, split a cluster until each cluster contains a point (or there are k clusters)

Traditional hierarchical algorithms use a similarity or distance matrix

- Merge or split one cluster at a time

Agglomerative Clustering Algorithm

More popular hierarchical clustering technique

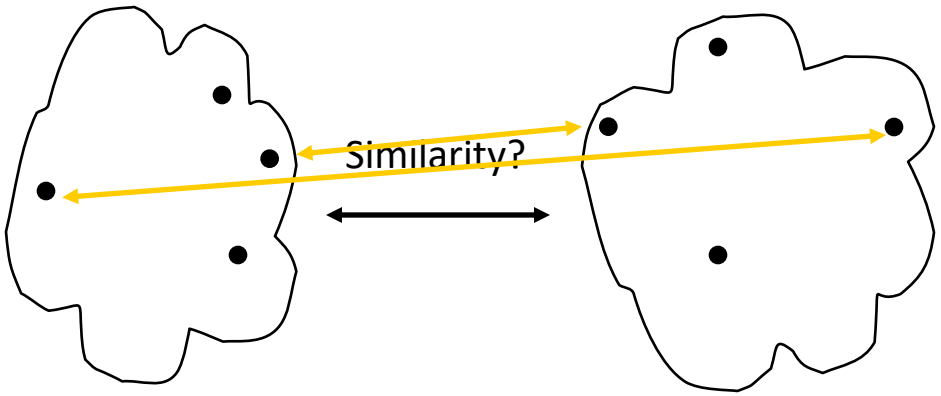
Basic algorithm is straightforward

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

Inter-Cluster Similarity



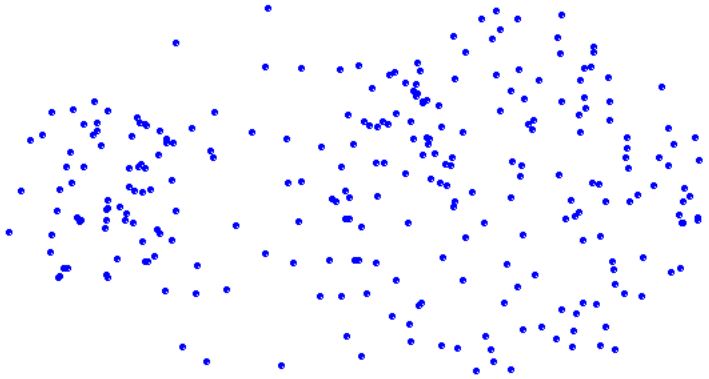
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

. .

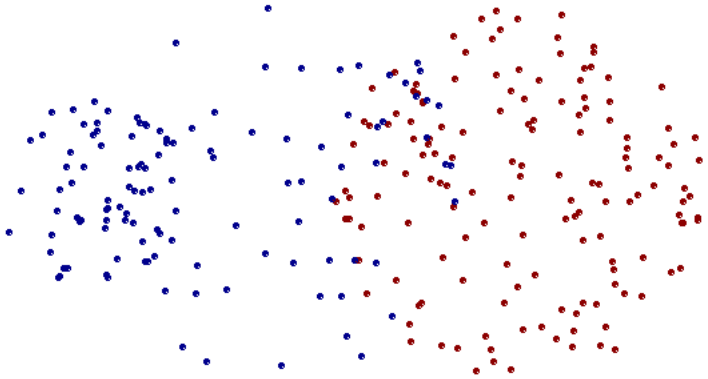
. Proximity Matrix

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

Limitations of MIN



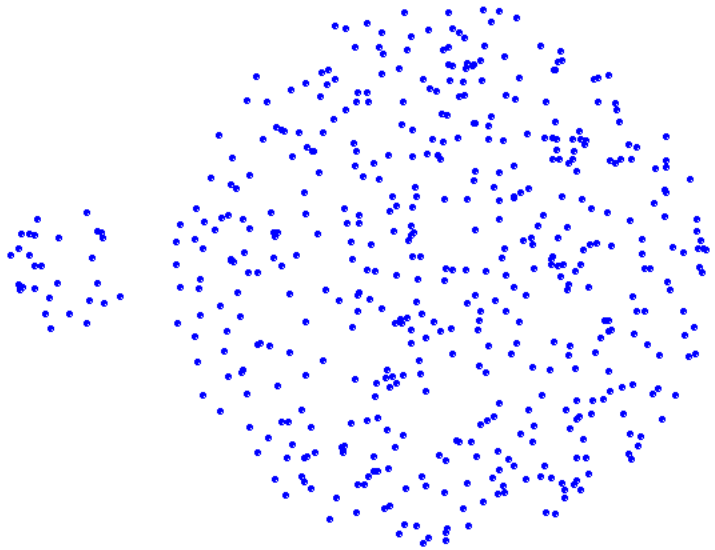
Original Points



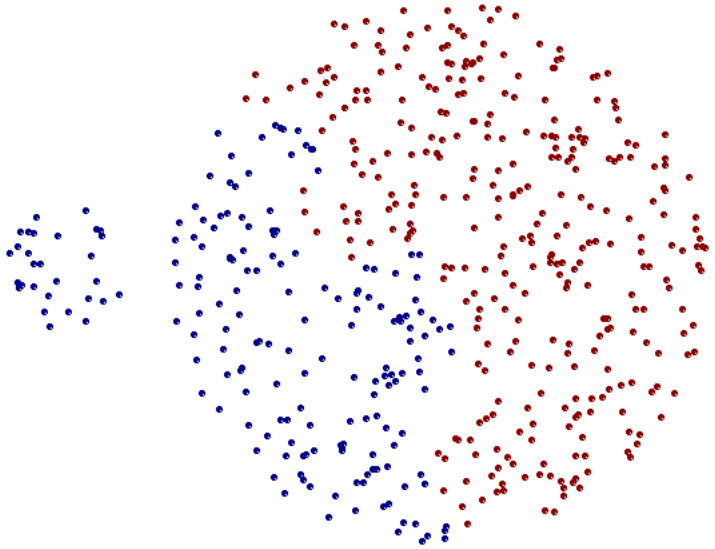
Two Clusters

- Sensitive to noise and outliers

Limitations of MAX



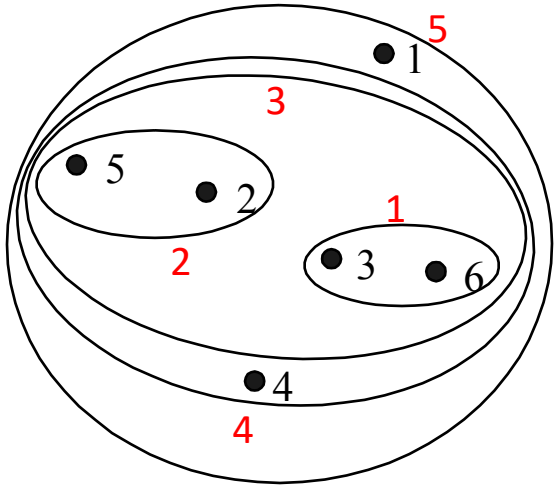
Original Points



Two Clusters

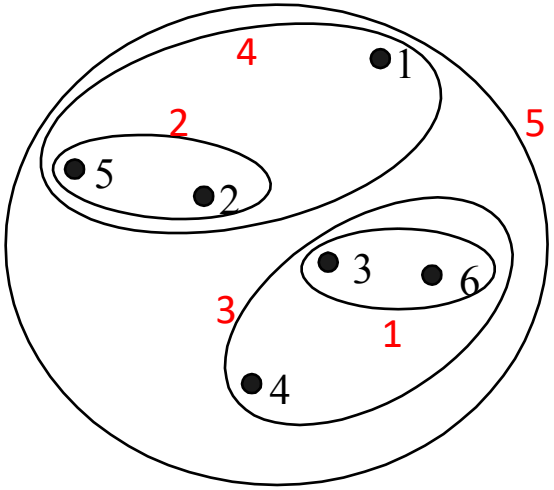
- Tends to break large clusters
- Biased towards globular clusters

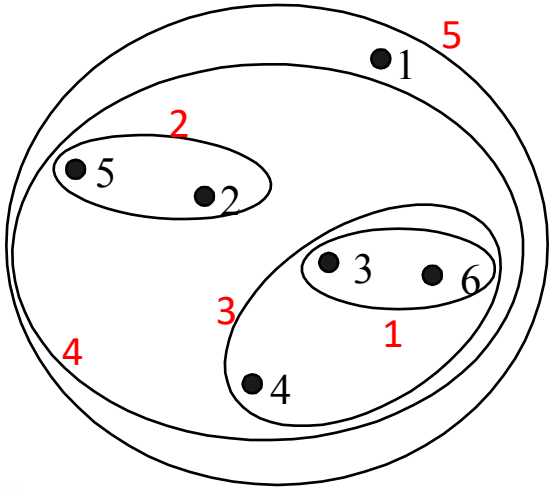
Comparison of Different Measures for Similarity among Clusters



MIN

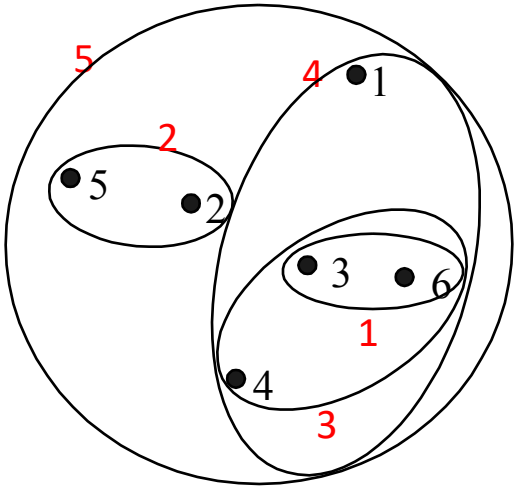
MAX





Group Average

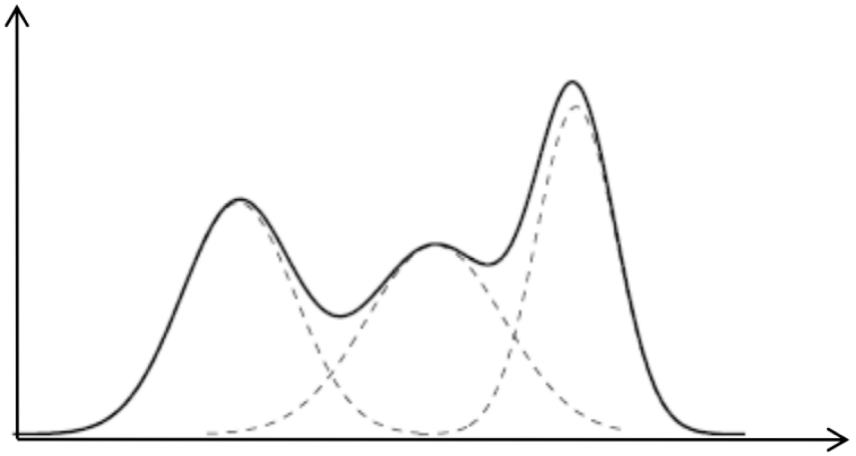
Ward's Method



Learning Data Distribution (Density)

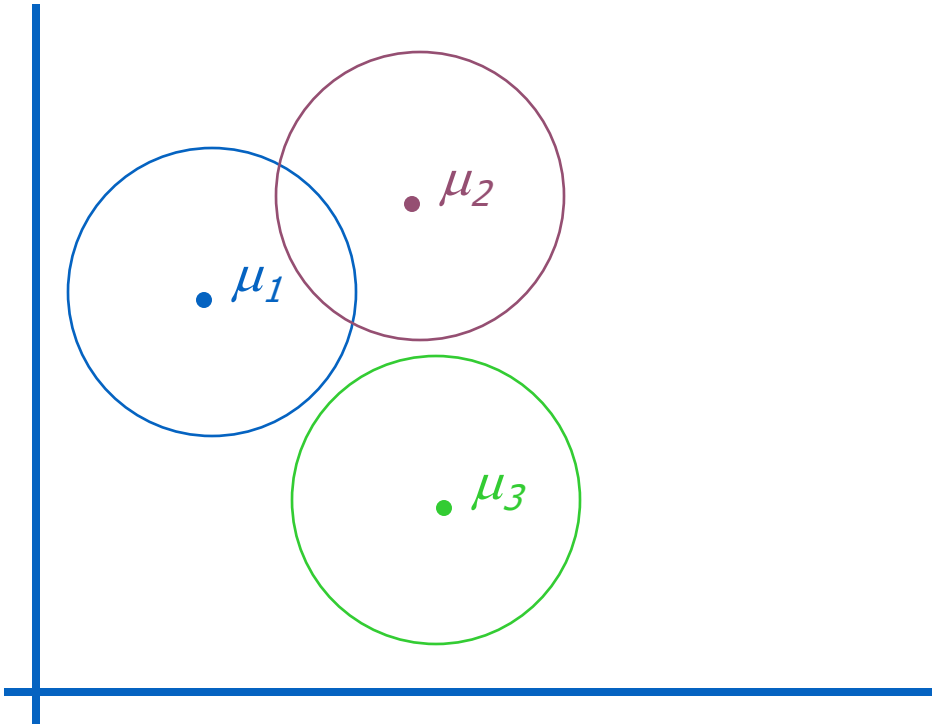
Gaussian Mixture Model

- Consider the observation is a mixture of K Gaussian distributions.
- Given a set of input data, estimate distribution parameters that can best describe date.
- Each cluster consists of observation follows the same Gaussian distribution.



Basic Assumption

- There are k components. The i' th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian with mean μ_i and covariance matrix $\sigma^2\boldsymbol{I}$



General Steps

- Similar EM types algorithm like K-means
- Steps in GMM,
  - choose initial guess of the parameters
  - E-step: Find excepted classes for each data point
  - M-Step: update parameters of each component, e.g. location, normalization and shape.
- Goal: maximize the log likelihood of all data.
- The output cluster is described by a smooth Gaussian model instead of sphere.

E.M. for GMMs

Iterate. On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1(t), \mu_2(t) \ \ldots \ \mu_c(t) \}$$

E-step

Compute "expected" classes of all data points for each class

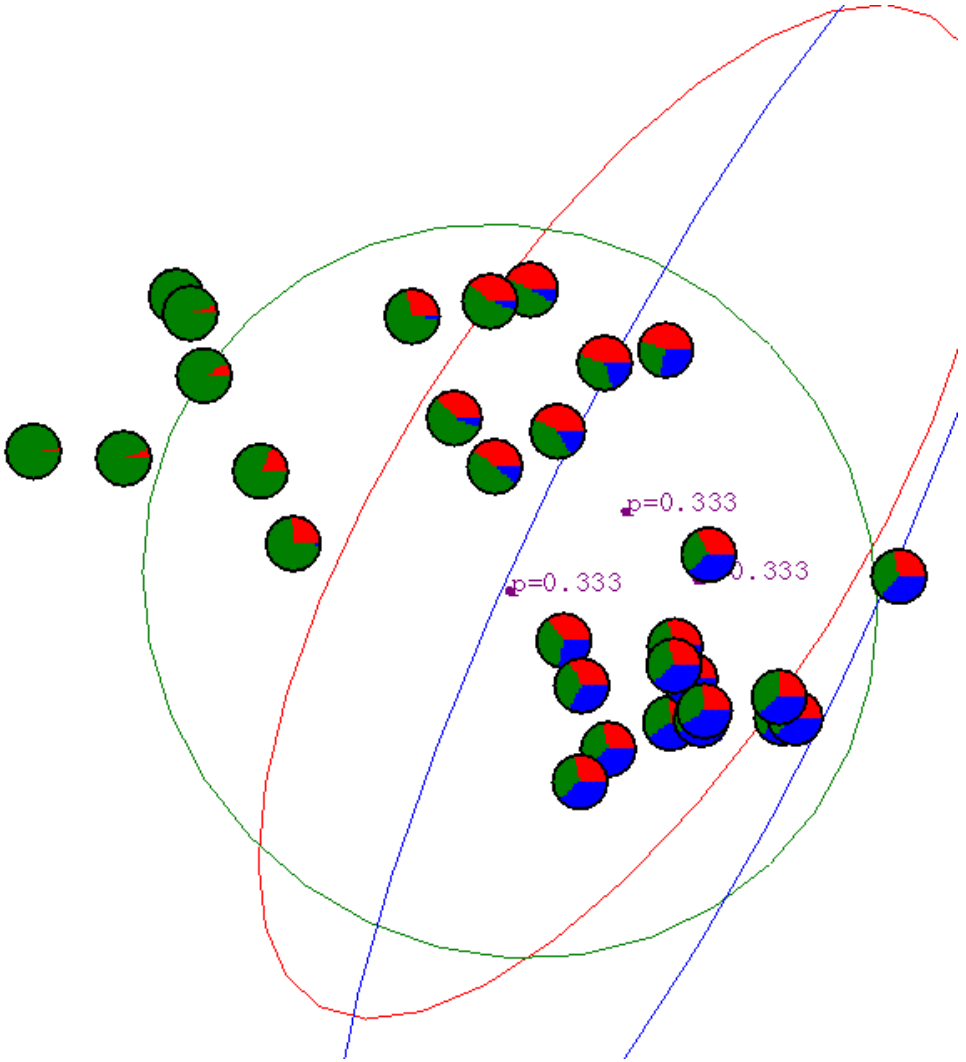
$$P(w_i | x_k, \lambda_t) = \frac{p(x_k | w_i, \lambda_t) P(w_i | \lambda_t)}{p(x_k | \lambda_t)} = \frac{p(x_k | w_i, \mu_i(t), \sigma^2 \mathbf{I}) p_i(t)}{\sum_{j=1}^{c} p(x_k | w_j, \mu_j(t), \sigma^2 \mathbf{I}) p_j(t)}$$

M-step.

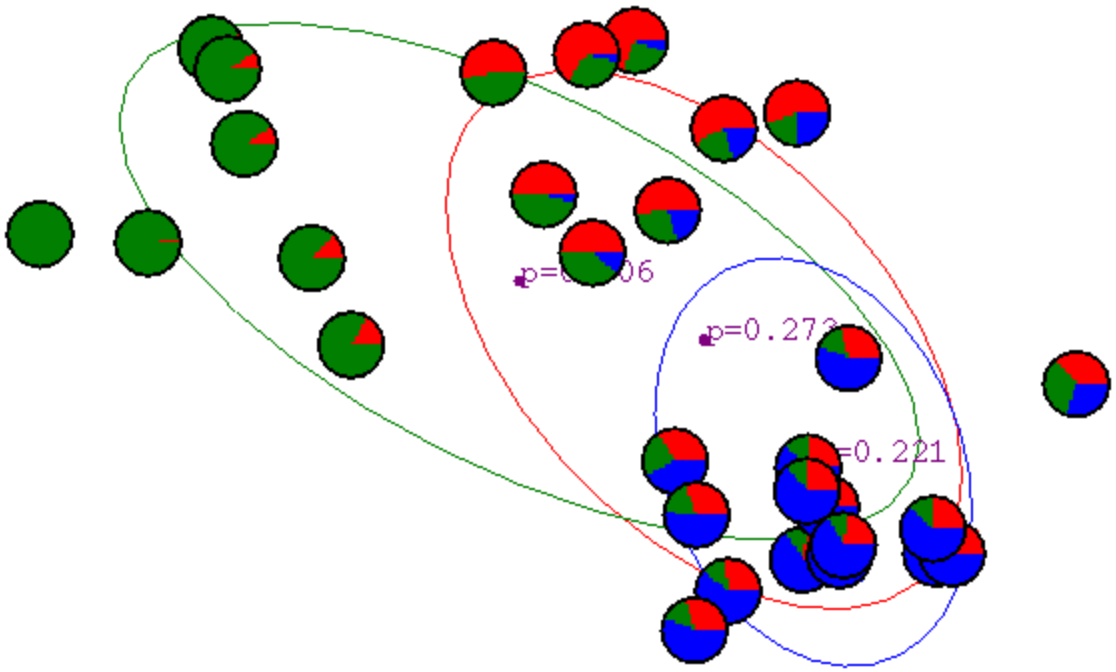
Compute Max. like **μ** given our data's class membership distributions

$$\mu_i(t+1) = \frac{\sum_k P(w_i | x_k, \lambda_t) x_k}{\sum_k P(w_i | x_k, \lambda_t)}$$

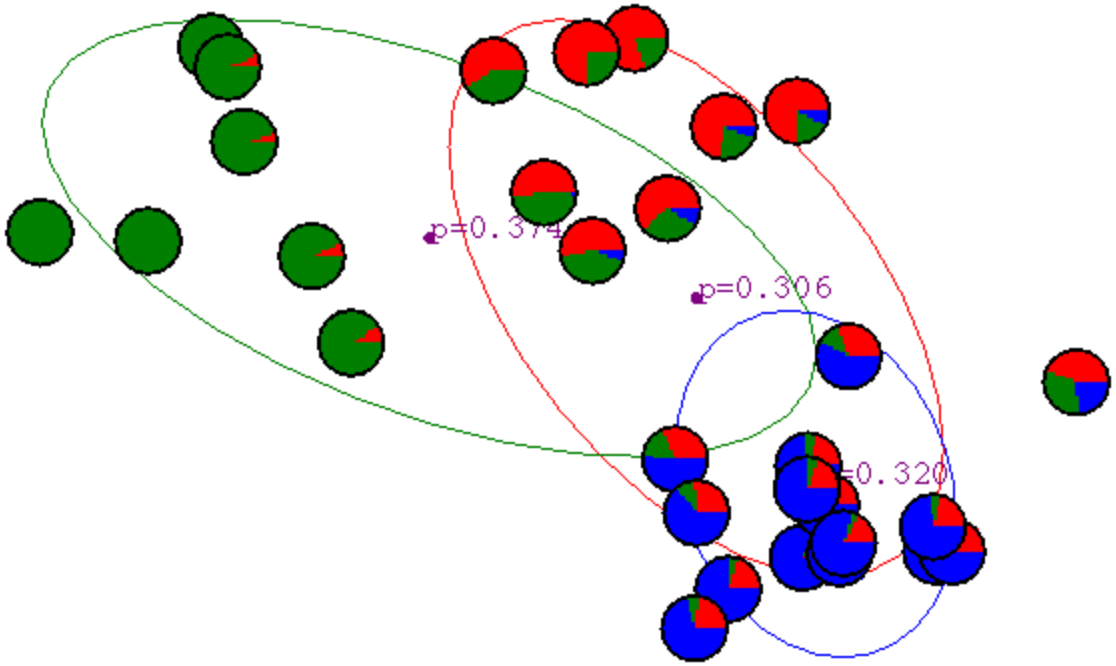
Gaussian Mixture Example: Start



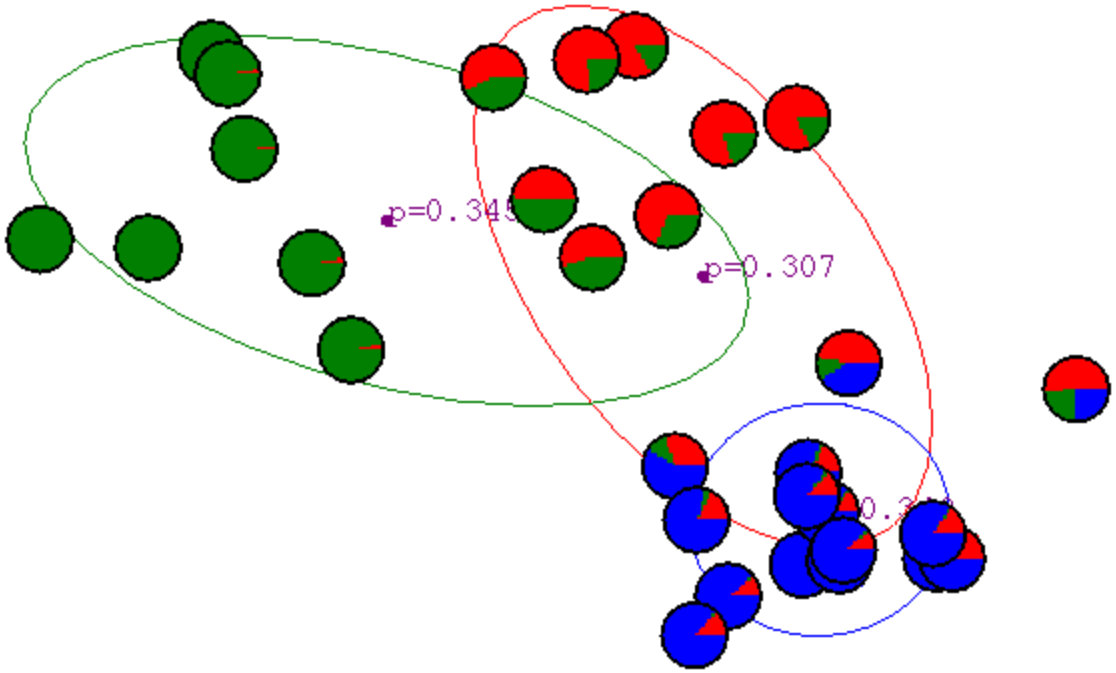
After first iteration



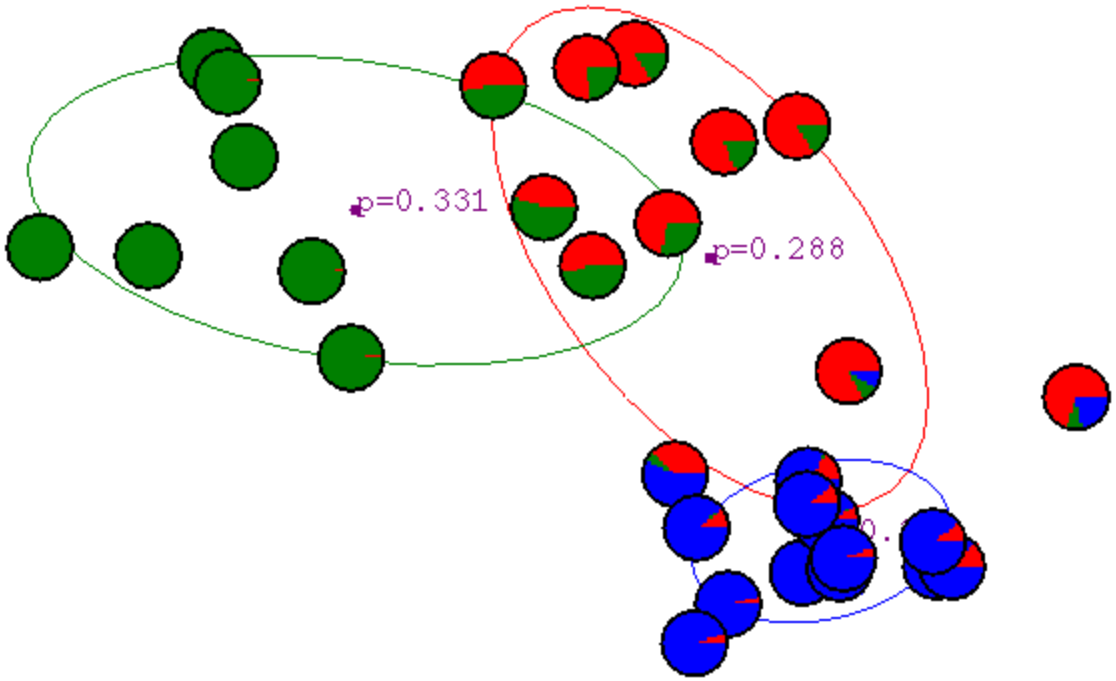
After 2nd iteration



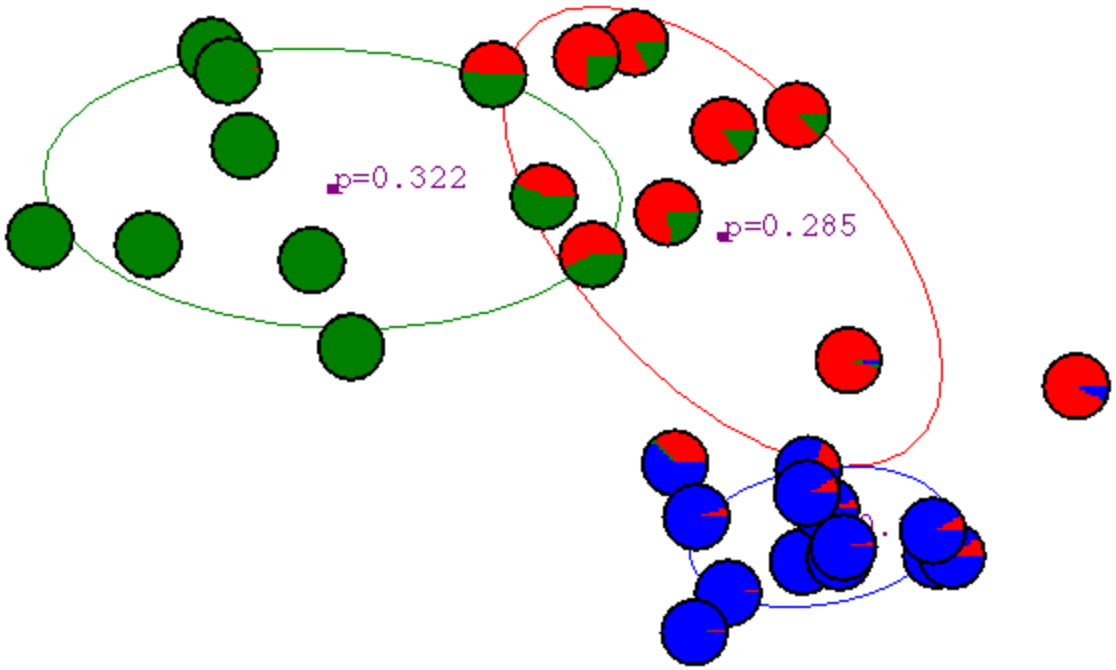
After 3rd iteration



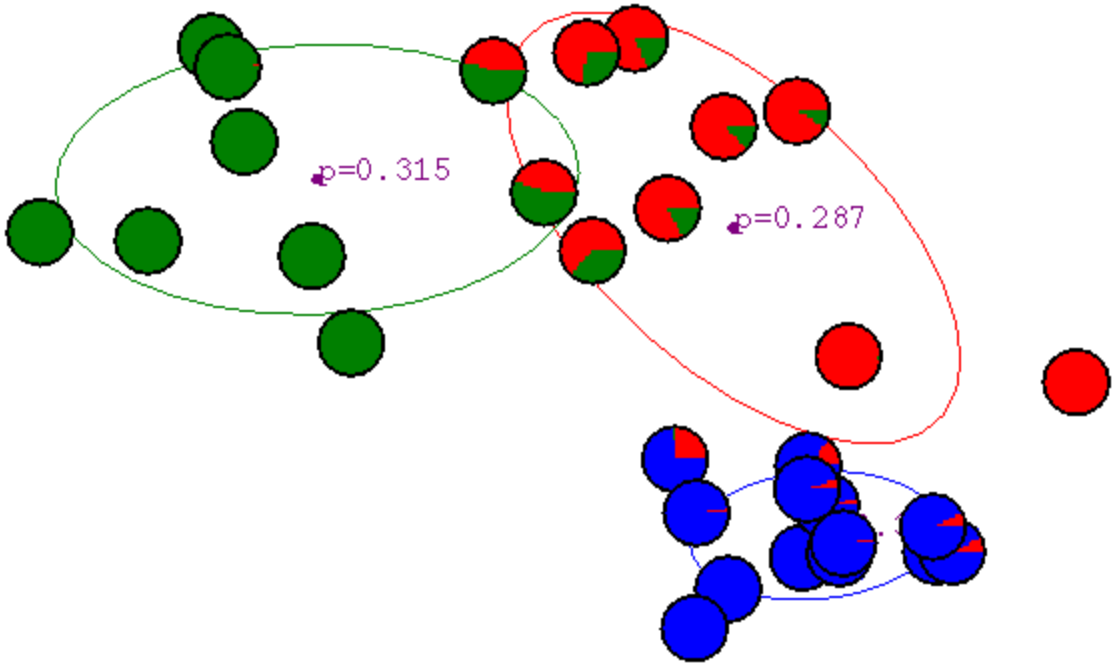
After 4th iteration



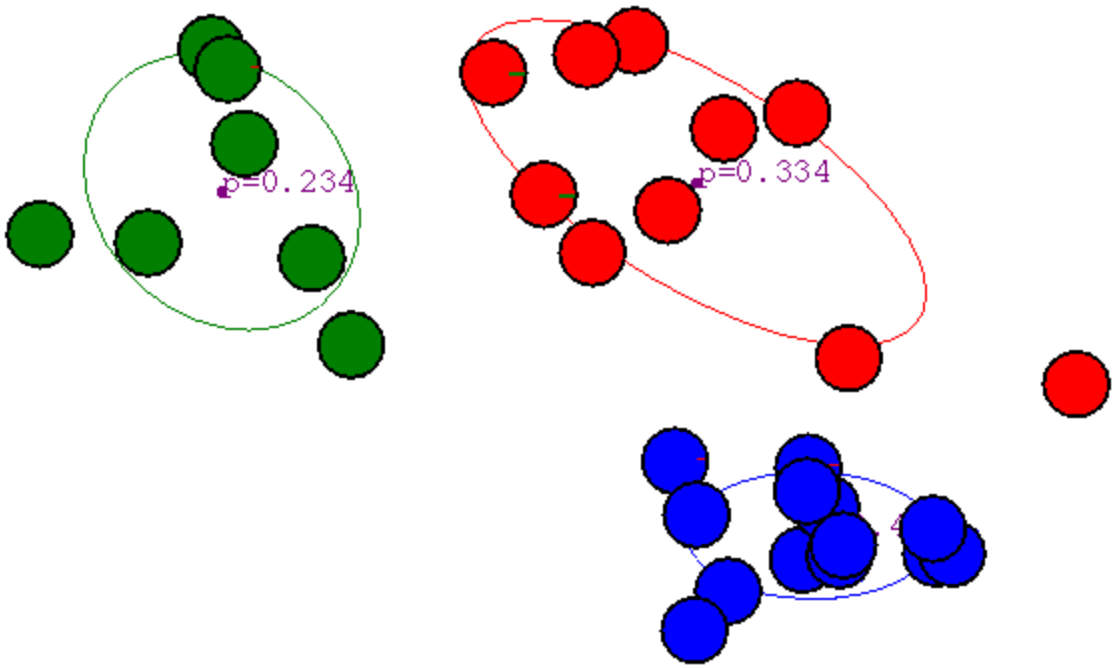
After 5th iteration



After 6th iteration

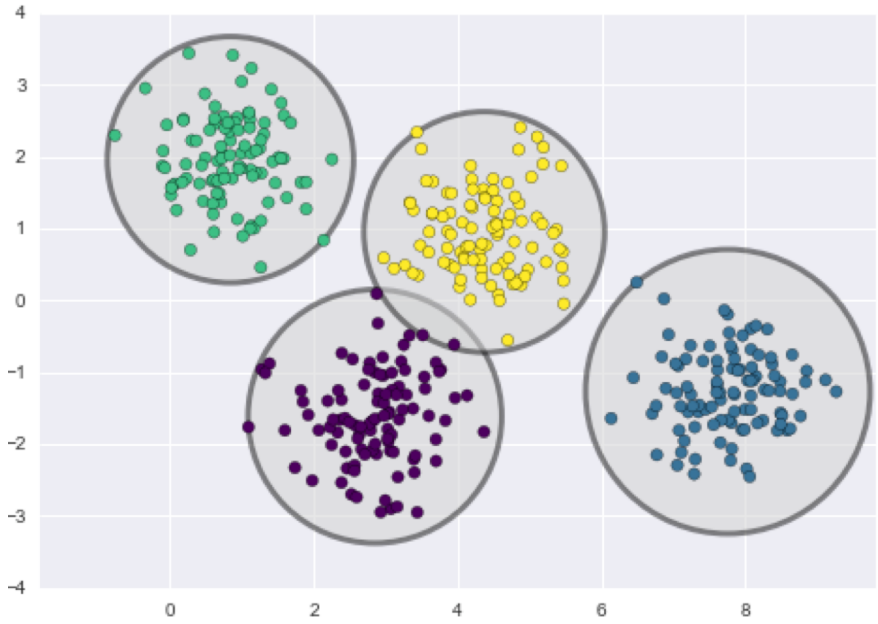


After 20th iteration

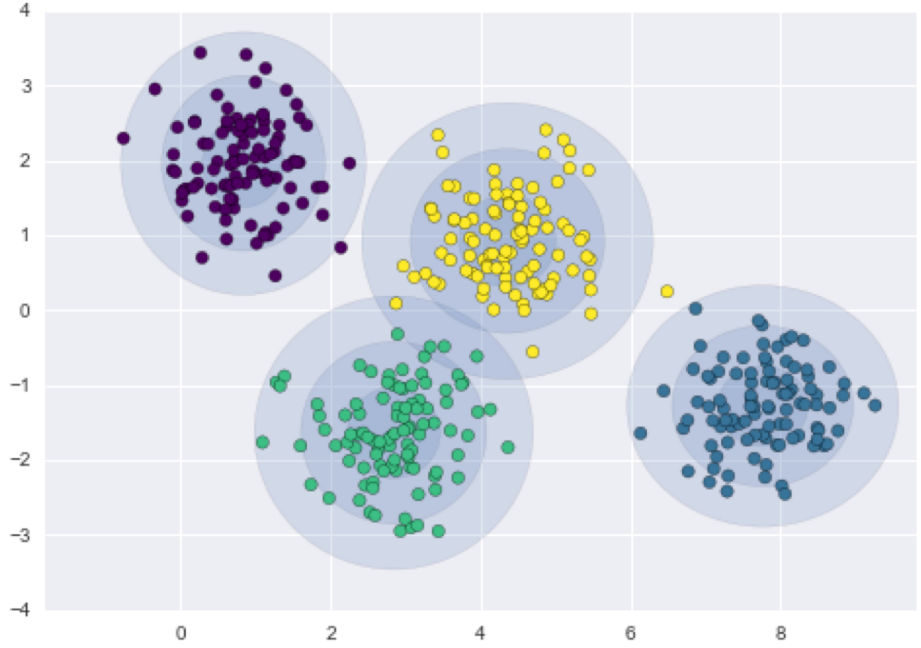


GMM vs. K-Means

Clustering Boundaries in K-Means



Clustering Boundaries with GMM



DBSCAN -- Density Based Clustering

Density = number of points within a specified radius (Eps)

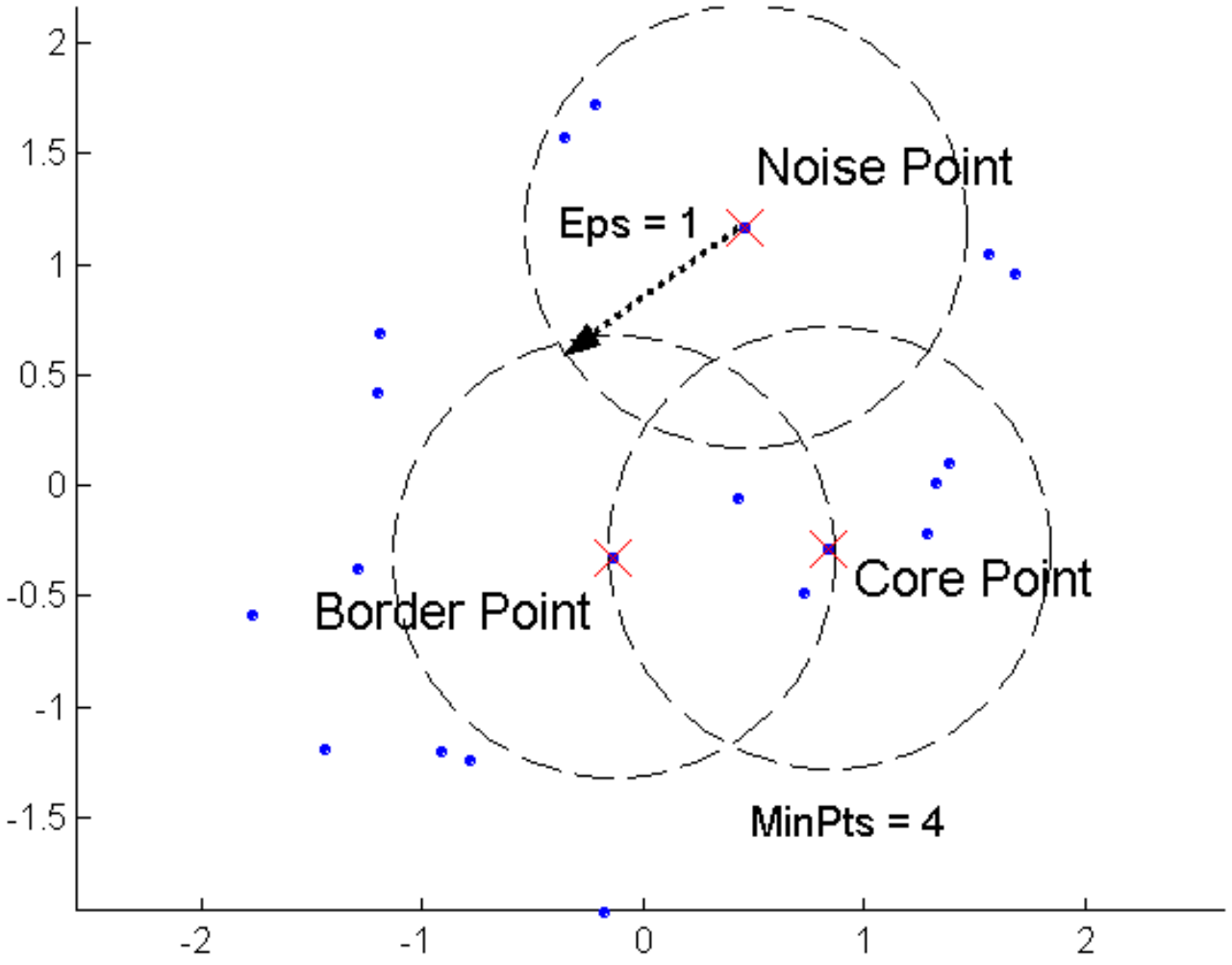
A point is a core point if it has more than a specified number of points (MinPts) within Eps

These are points that are at the interior of a cluster

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point

A noise point is any point that is not a core point or a border point.

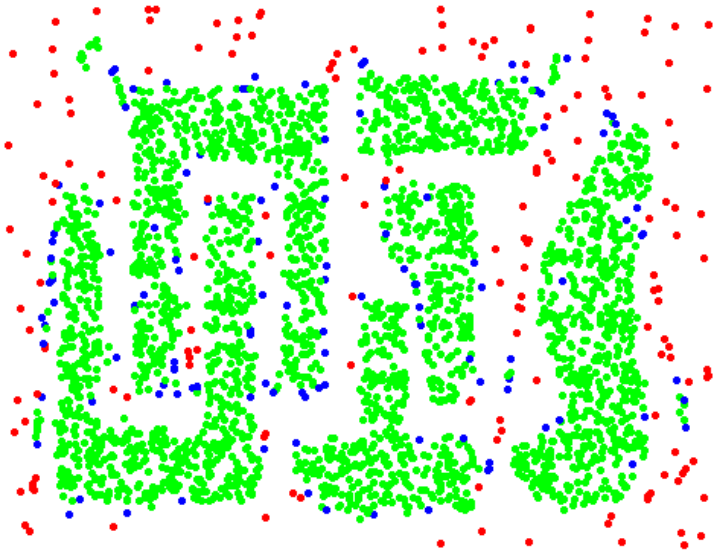
Core, Border, and Noise Points



Density based clustering: Example

Original Points

Eps = 10, MinPts = 4



Point types: core, border and noise

DBSCAN Algorithm

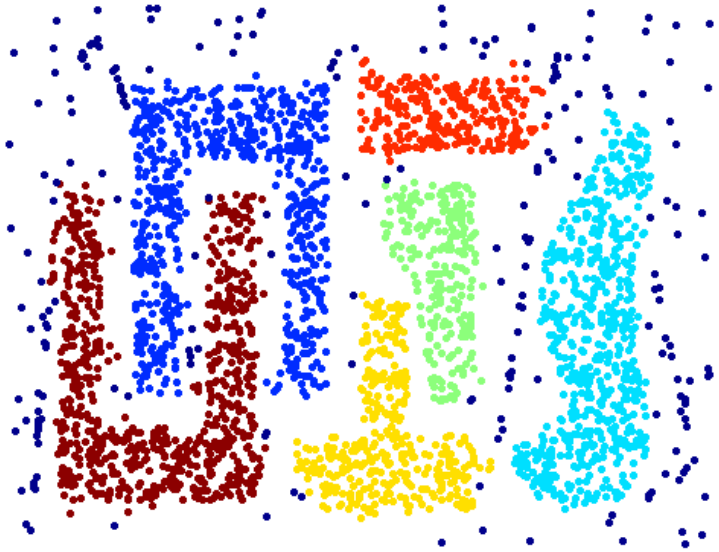
Eliminate noise points

Perform clustering on the remaining points

```
current_cluster_label ← 1
for all core points do
  if the core point has no cluster label then
    current_cluster_label ← current_cluster_label + 1
    Label the current core point with cluster label current_cluster_label
  end if
  for all points in the Eps-neighborhood, except i^th the point itself do
    if the point does not have a cluster label then
      Label the point with cluster label current_cluster_label
    end if
  end for
end for
```

Density based clustering: Example

Original Points

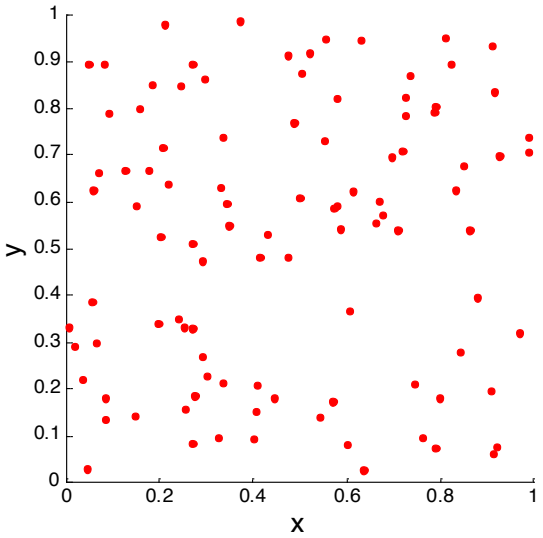


Clusters

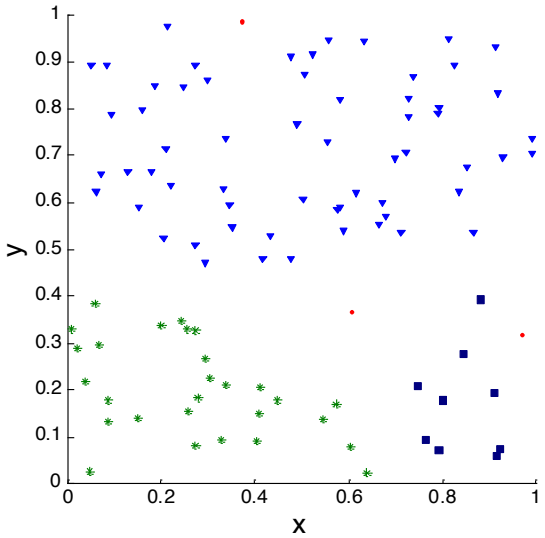
- Resistant to Noise
- Can handle clusters of different shapes and sizes

Clusters found in Random Data

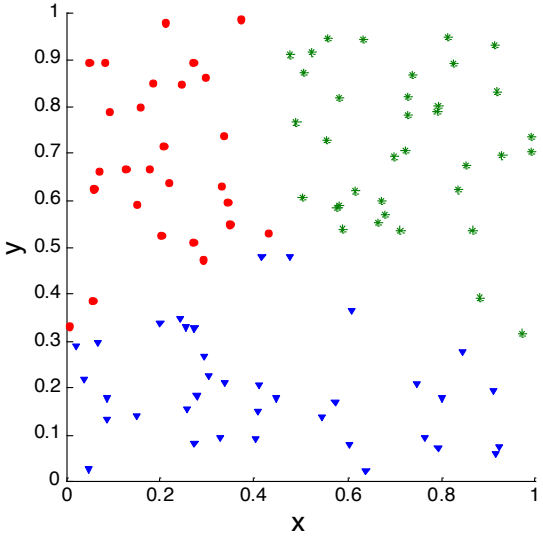
Random Points



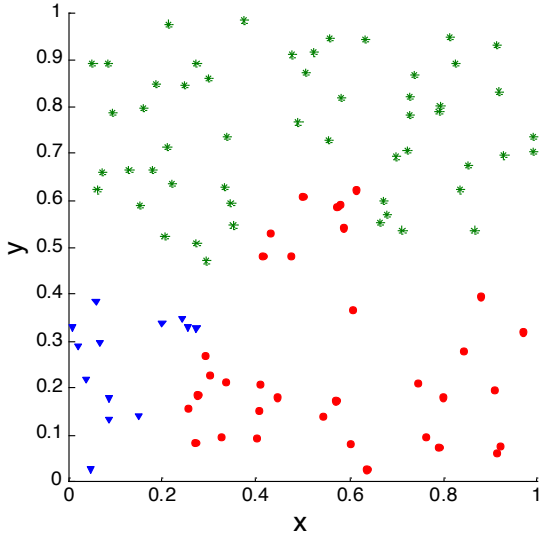
DBSCAN



K-means



Complete Link



Hands on session

1. Log on to the Visualization Portal with your TACC account.
2. Launch a Jupyter Notebook job from the Visualization Portal.

   Use reservation "**ML_Institute_day3**" on **Frontera**.

   Use "**dev**" queue

Wait for Jupyter notebook to start

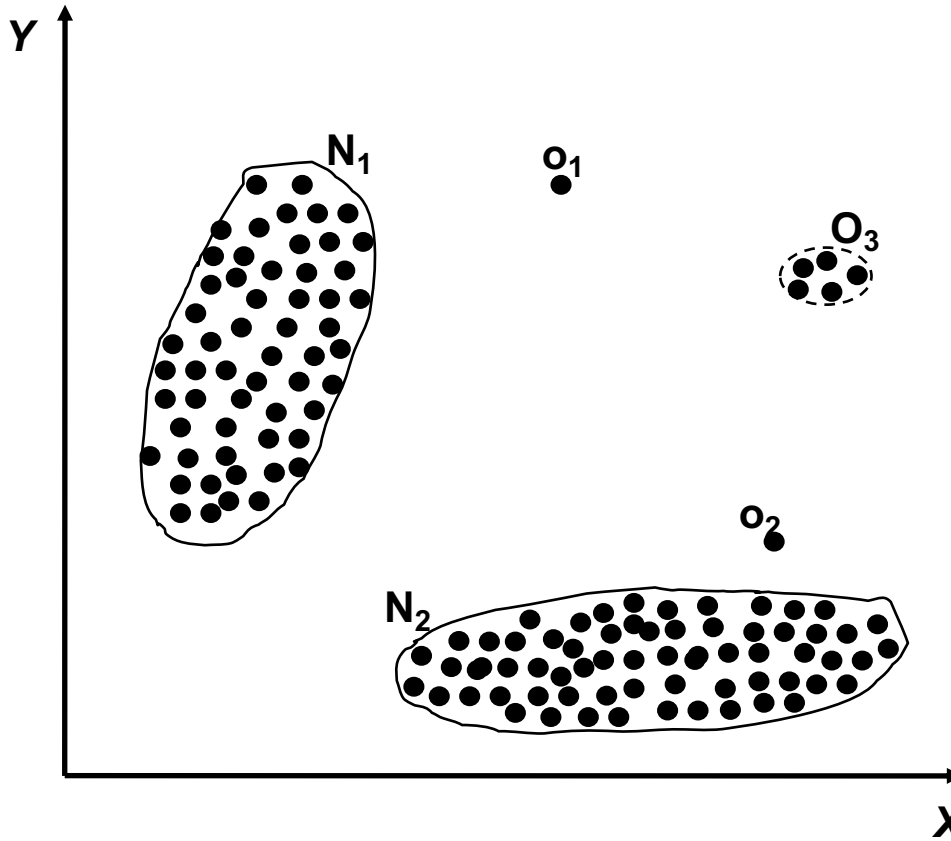
3. Open a new terminal from your Jupyter session.

4. Copy example notebook over:
   cp /work/00791/xwj/DMS/TACC_SSI_2021/unsupervised.ipynb .
5. Open 'unsupervised.ipyb'

Anomaly Detection

Anomaly Detection

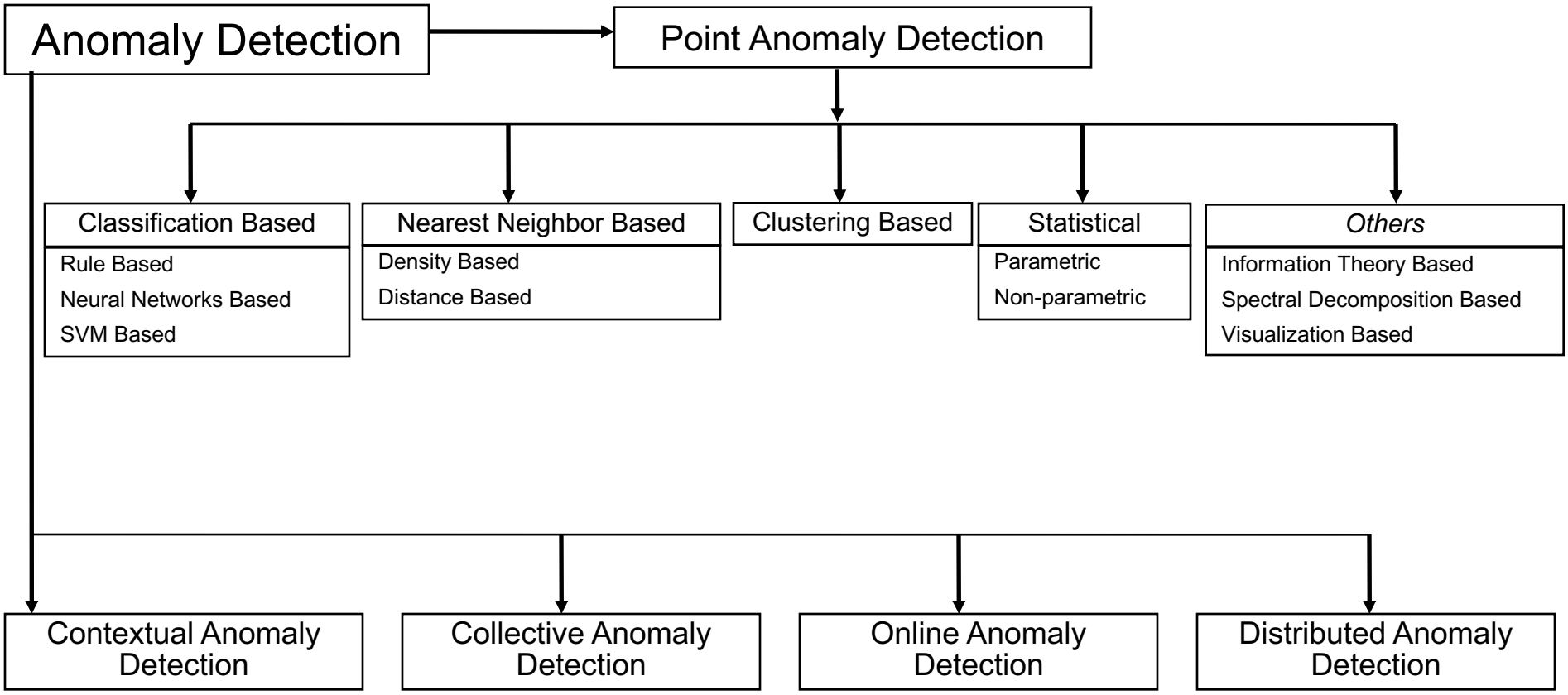
- Anomaly is a pattern in the data that does not conform to the expected behaviour
  - Usually in small numbers.
  - Occurred infrequently
  - Out of normal feature space
- Important use cases
  - Credit Card Fraud detection
  - Early warning system
  - Cyber attack intrusion



Anomaly Detection Problem

- It represents a class of problem that can be solved with various methods depends on
  - Nature of data
  - Availability of training data
  - Type of anomaly.
  - …
- Many of the supervised and unsupervised learning methods may potentially adapted for this problem.
- Key issues:
  - Define the representation of “normal”?
  - Outliers vs Noise.

Taxonomy*



* Outlier Detection – A Survey, Varun Chandola, Arindam Banerjee, and Vipin Kumar, Technical Report TR07-17, University of Minnesota (Under Review)

Supervised Anomaly Detection

- Data comes with label of normal or outliers
- Build a classification model to distinguish normal and outliers classes for new data.
- Any supervised learning methods can be used for this purpose
  - However, classes maybe unbalanced, e.g. fewer outliers but a lot of “normal”
    - Oversample rare class / downsample majority class.
    - Generate synthetic anomaly.

Unsupervised Anomaly detection

- As a “one-class” classification problem
  - One-class SVM.
    - Expected number of outliers.
    - Using modified RBF kernel.
- Distance based methods
  - Distance to Nearest neighbor / cluster
- Density based methods
  - How many points within ”neighborhood”
  - Local Outlier Factor
- Distribution based methods
  - Identify normal data distribution
  - Minimum Covariance Determinant
- Feature space based methods
  - Isolation forests.

Hands on session

1. Log on to the Visualization Portal with your TACC account.
2. Launch a Jupyter Notebook job from the Visualization Portal.

   Use reservation "**ML_Institute_day3**" on **Frontera**.

   Use "**dev**" queue

Wait for Jupyter notebook to start

3. Open a new terminal from your Jupyter session.

4. Copy example notebook over:
   cp /work/00791/xwj/DMS/TACC_SSI_2021/unsupervised.ipynb .
5. Open 'unsupervised.ipyb'