IERG4300/ IEMS5709
Web-Scale Information Analytics

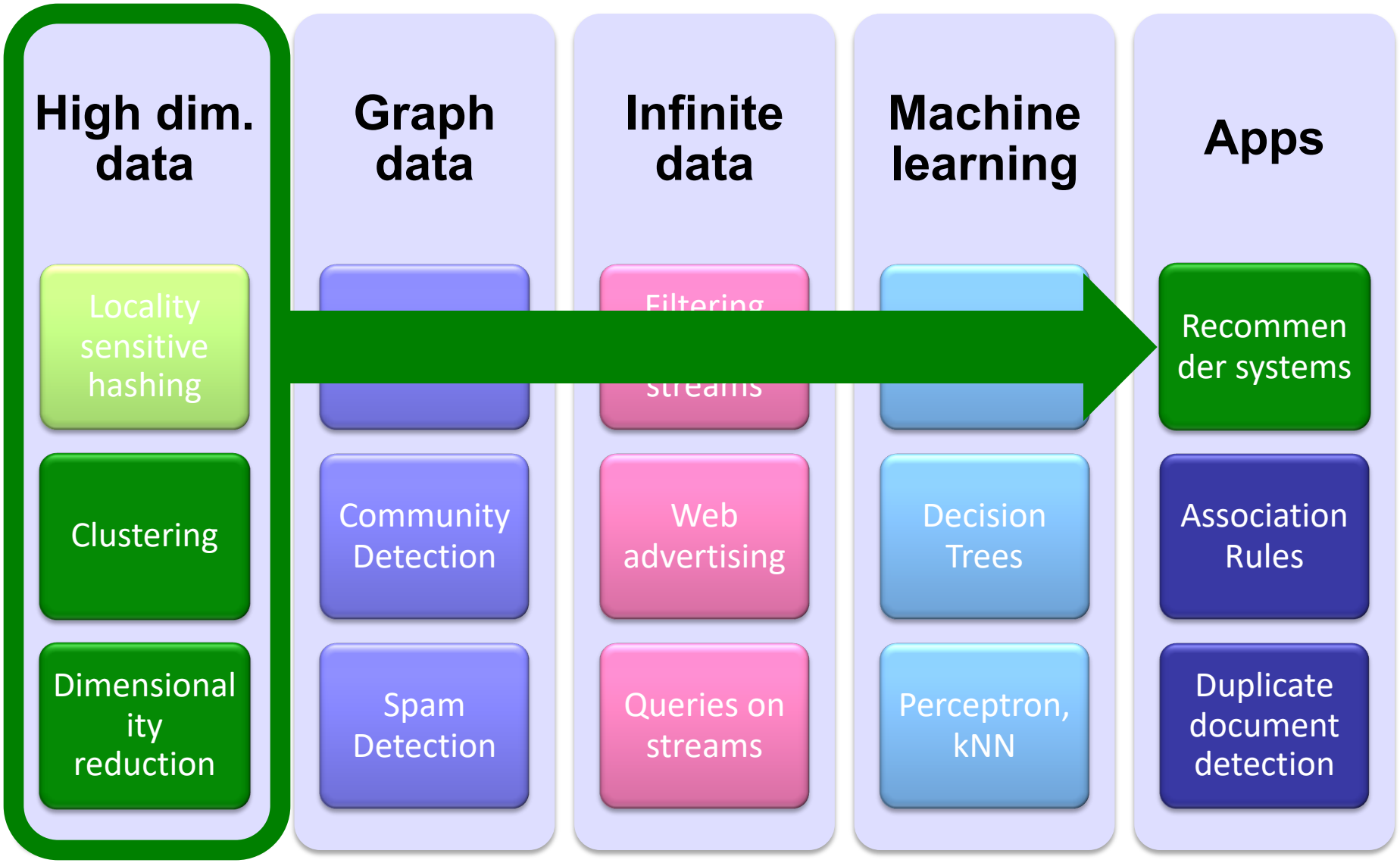
Clustering

Prof. Wing C. Lau
Department of Information Engineering
wclau@ie.cuhk.edu.hk

Acknowledgements

- The slides used in this chapter are adapted from:
  - CS246 Mining Massive Data-sets, by Jure Leskovec, Stanford University.
  - Statistical Data Mining Tutorials – Tutorial Slides by Andrew W. Moore, CMU, http://www.autonlab.org/tutorials/list.html
  - Ch 1 and 9 of Pattern Recognition and Machine Learning (PRML) by Christopher M. Bishop, Publisher: Springer Science and Business.
    - http://www.cse.psu.edu/~rcollins/CSE586Spring2010/papers/prmlMixturesEM.pdf

High Dimensional Data



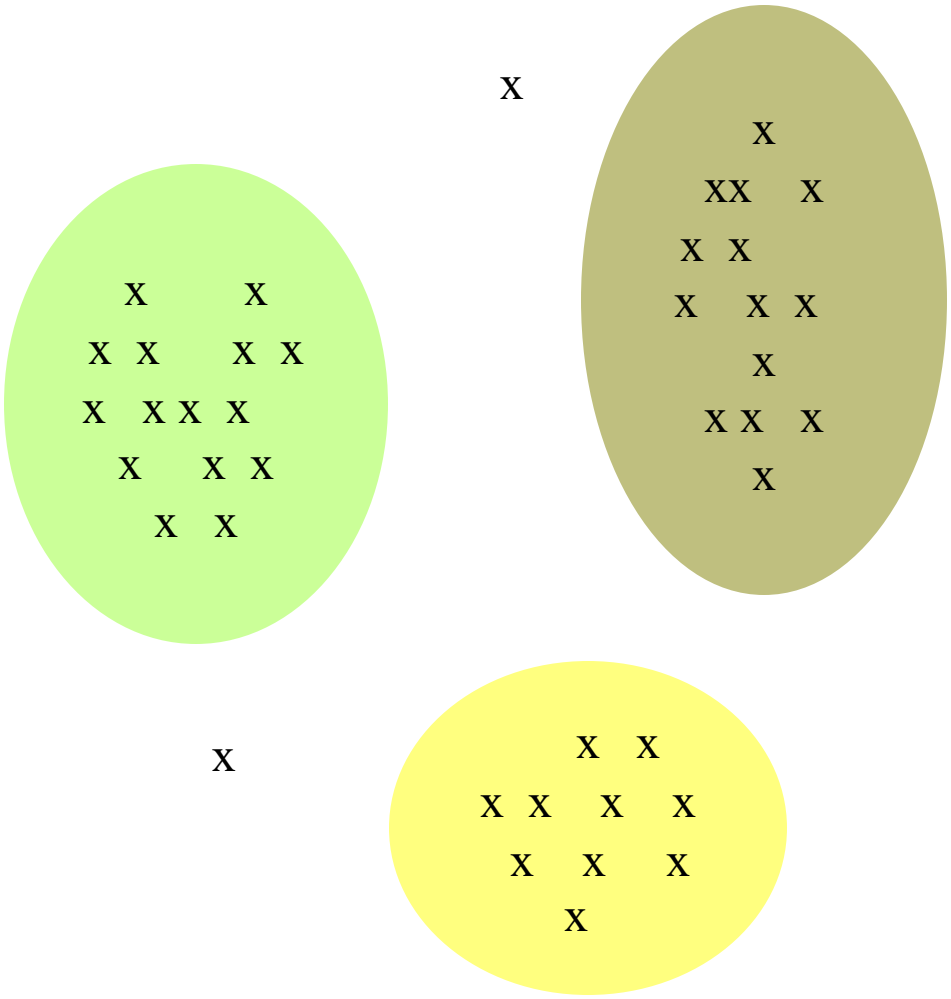
High Dimensional Data

- Given a cloud of data points we want to understand their structure

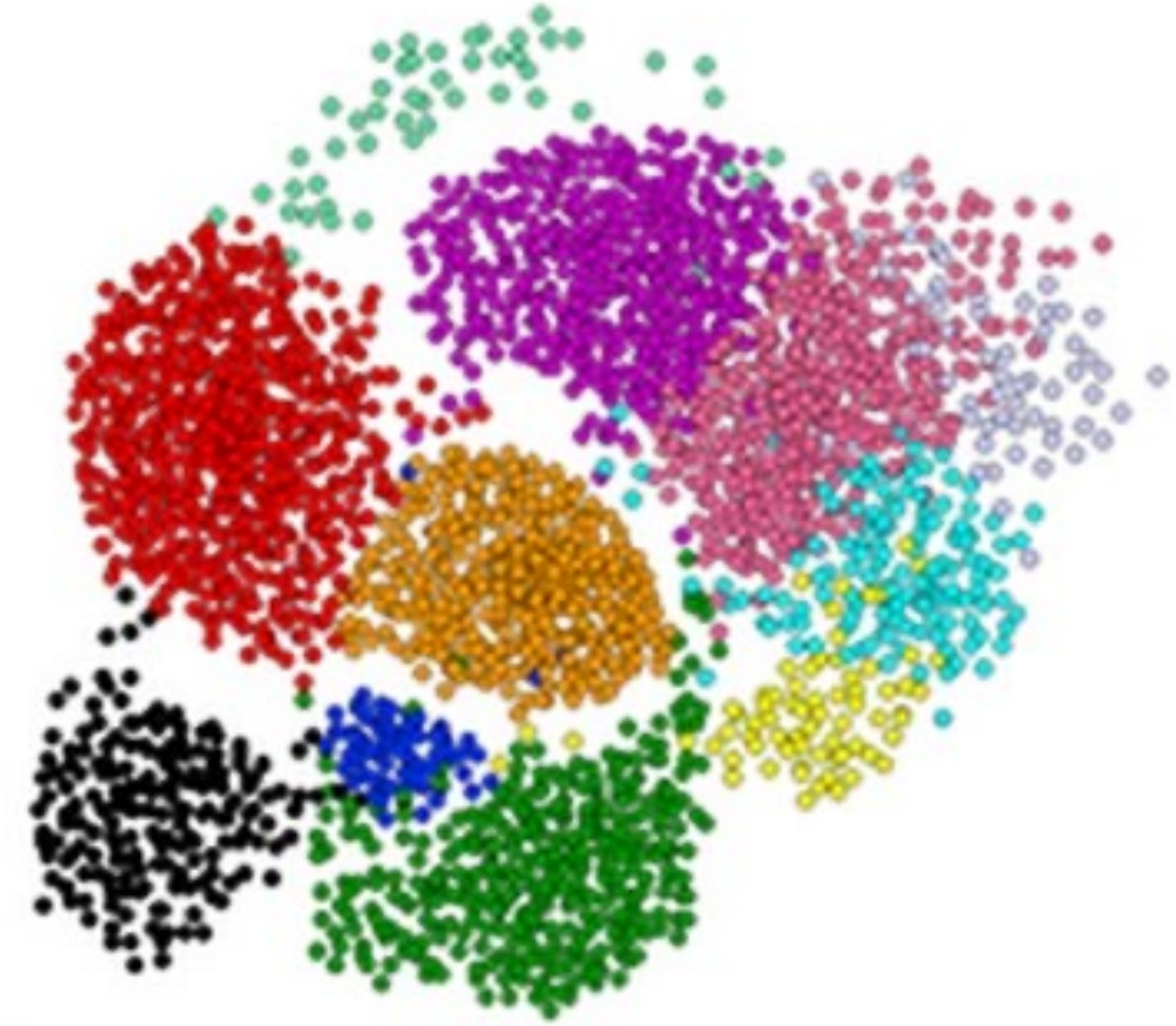
The Problem of Clustering

- Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***, so that
  - Members of a cluster are close/similar to each other
  - Members of different clusters are dissimilar
- **Usually:**
  - Points are in a high-dimensional space
  - Similarity is defined using a distance measure
    - Euclidean, Cosine, Jaccard, edit distance, …

Example: Clusters



Clustering is a hard problem!



Why is it hard?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- And in most cases, looks are *not* deceiving
- Many applications involve not 2, but 10 or 10,000 dimensions
- **High-dimensional spaces look different**

Curse of Dimensionality

- **High-dimensional spaces look different:** The so-called "Curse of Dimensionality"
  - Need many more data points scattered in the High Dimensional (HD) space in order to form clusters (instead of being isolated dots in the mostly empty space) !
  - Almost all pairs of points are at about the same distance in HD space

=> The notion of neighborhood becomes not very useful as the distance between a data point and is its nearest neighbor approaches the distance to its farthest neighbor.

=> "neighborhood is not that local !

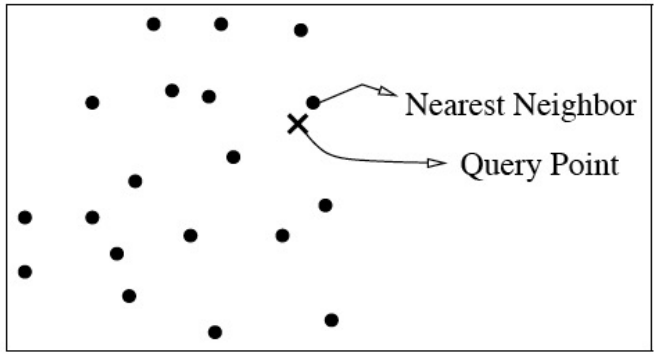


Fig. 1. Query point and its nearest neighbor.

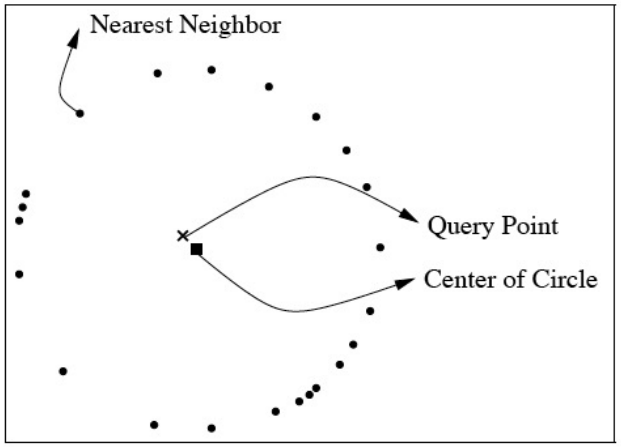


Fig. 2. Another query point and its nearest neighbor.

Source: K.Beyer et al, "When is "Nearest Neighbor" Meaningful ?", ICDT 99.

Curse of Dimensionality (cont'd)

Consider a sphere of radius r=1 in a D-dimensional space

Fraction of the volume of the sphere that lies between radius r=1-ε and r=1 is given by:

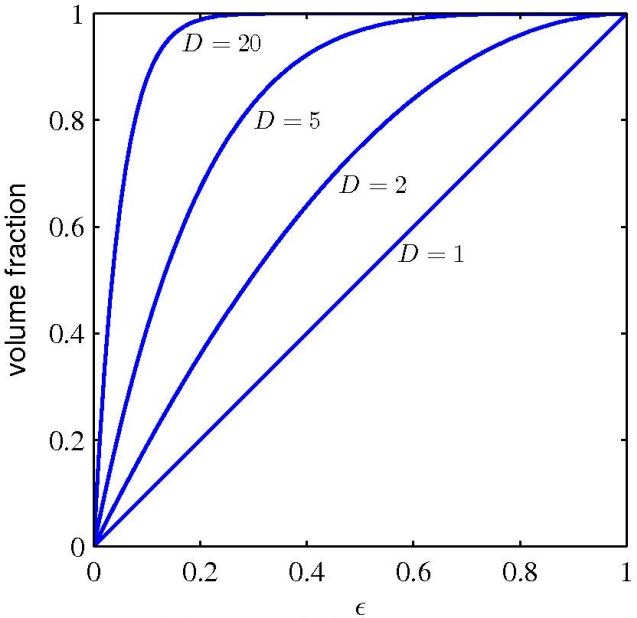
$$\frac{V_D(1) - V_D(1-\varepsilon)}{V_D(1)} = 1 - (1-\varepsilon)^D$$

where

$V_D(r)$ = Vol. of a D-dim sphere of radius r

$= K_D r^D$

for some constant K_D.



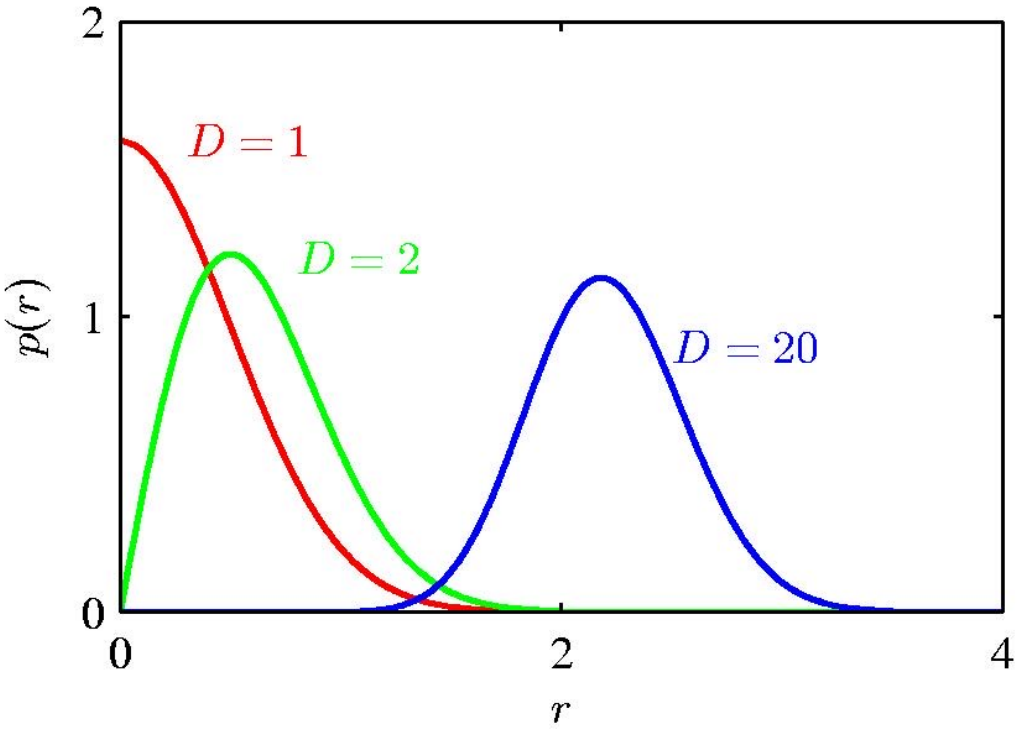
=> As D >> 1, most of the vol. of the sphere is concentrated in a thin-shell near the surface of the sphere……(**)

Assume data points are randomly scattered in the D-dim space under uniform density (i.e. const. # of data points per unit volume)

For an arbitrary data point P (whose position is taken as the origin), (**) implies that most of the neighboring points of P are of more or less the same distance from it (and NOT quite local to P) when D >> 1

Curse of Dimensionality (cont'd)

Gaussian Densities in higher dimensions



Clustering Problem: SkyCat

- **A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands)**
- **Problem: Cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.**
- Sloan Digital Sky Survey is a newer, better version of this

Example: Clustering CD's

- **Intuitively: Music divides into categories, and customers prefer a few categories**
  - But what are categories really?
- Represent a CD by a set of customers who bought it
- Similar CDs have similar sets of customers, and vice-versa

Example: Clustering CDs

Space of all CDs:

- Think of a space with one dim. for each customer
  - Values in a dimension may be 0 or 1 only
  - A CD is a point in this space is $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} customer bought the CD
    - Compare with boolean matrix: rows = customers; cols. = CDs
- For Amazon, the dimension is tens of millions
- **Task:** Find clusters of similar CDs

- **An alternative:** Use Minhash/LSH to get Jaccard distance between "close" CDs
- Use that as input to clustering

Example: Clustering Documents

Finding topics:

- Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} word (in some order) appears in the document
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words

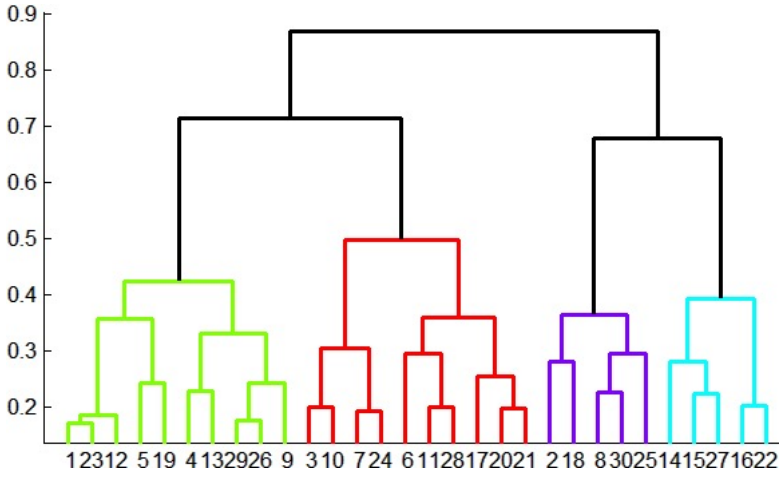
- **Documents with similar sets of words may be about the same topic**

Cosine, Jaccard, and Euclidean

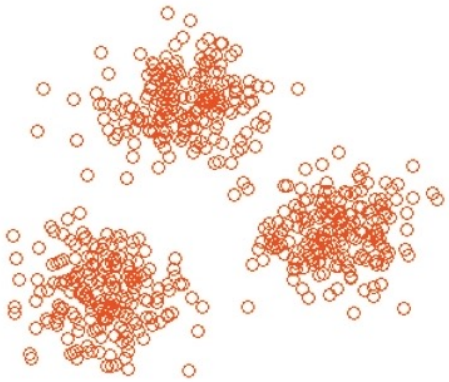
- **As with CDs we have a choice when we think of documents as sets of words or shingles:**
  - **Sets as vectors:** measure similarity by the cosine distance
  - **Sets as sets:** measure similarity by the Jaccard distance
  - **Sets as points:** measure similarity by Euclidean distance

Overview: Methods of Clustering

- **Hierarchical:**
  - **Agglomerative** (bottom up):
    - Initially, each point is a cluster
    - Repeatedly combine the two "nearest" clusters into one
  - **Divisive** (top down):
    - Start with one cluster and recursively split it

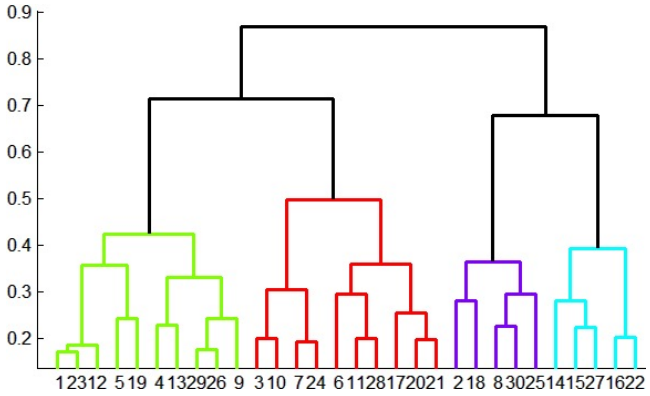


- **Point assignment:**
  - Maintain a set of clusters
  - Points belong to "nearest" cluster



Hierarchical Clustering

- **Key operation:**
  Repeatedly combine two nearest clusters

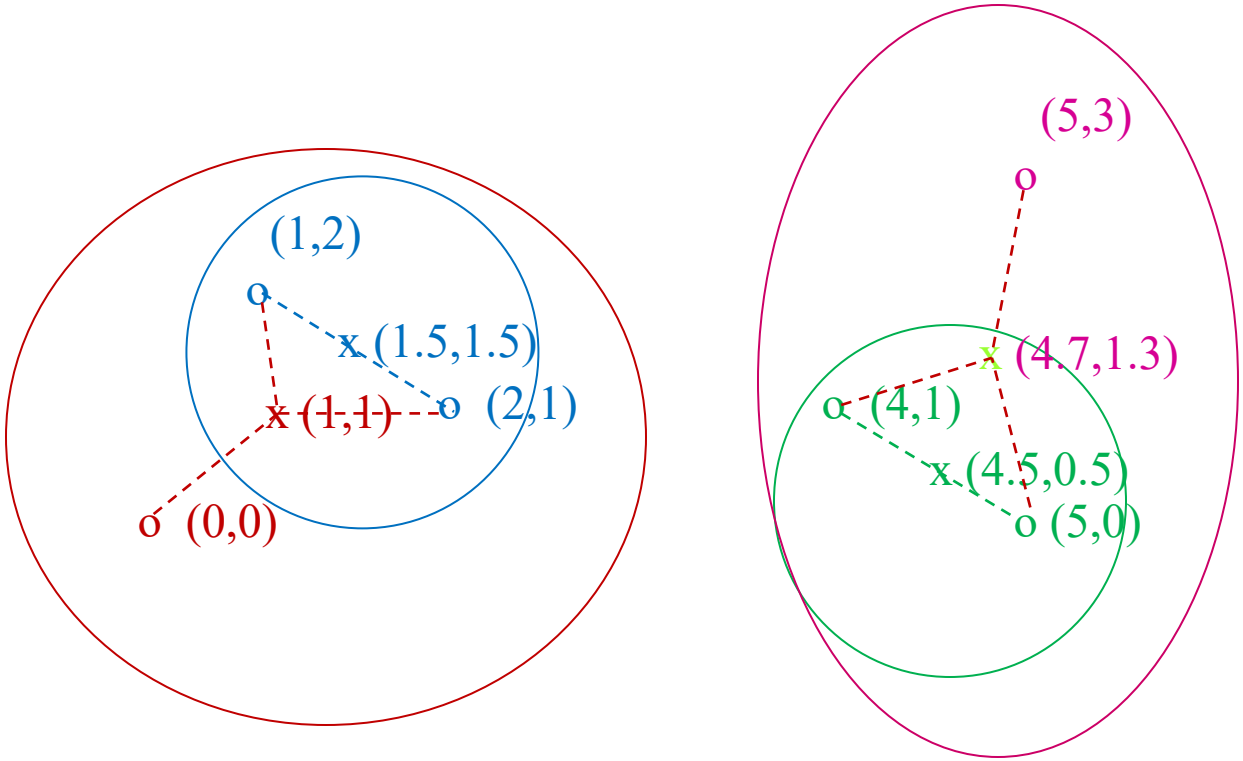


- **Three important questions:**
  - **1)** How do you represent a cluster of more than one point?
  - **2)** How do you determine the “nearness” of clusters?
  - **3)** When to stop combining clusters?

Hierarchical Clustering

- **Key operation: Repeatedly combine two nearest clusters**
- **(1) How to represent a cluster of many points?**
  - **Key problem:** As you build clusters, how do you represent the location of each cluster, to tell which pair of clusters is closest?
- **Euclidean case:** each cluster has a *centroid* = average of its (data)points
- **(2) How to determine "nearness" of clusters?**
  - Measure cluster distances by distances of centroids

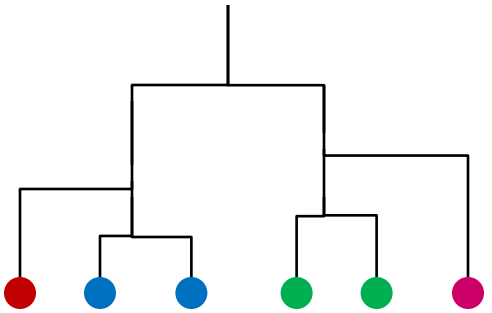
Example: Hierarchical clustering



Data:

o … data point

x … centroid



Dendrogram

And in the Non-Euclidean Case?

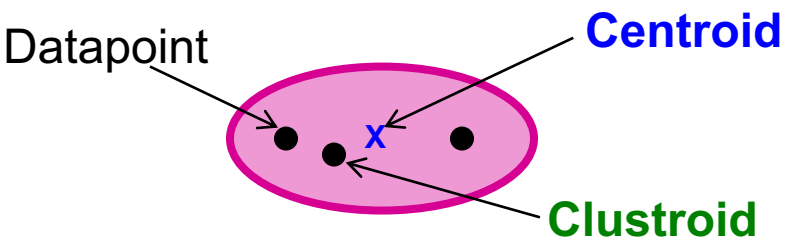
What about the Non-Euclidean case?

- The only "locations" we can talk about are the points themselves
  - i.e., there is no "average" of two points

- **Approach 1:**
  - **(1) How to represent a cluster of many points?**
    clustroid = (data)point "***closest***" to other points
  - **(2) How do you determine the "nearness" of clusters?** Treat clustroid as if it were centroid, when computing intercluster distances

"Closest" Point?

- **(1) How to represent a cluster of many points?**
  clustroid = point "***closest***" to other points
- **Possible meanings of "closest":**
  - Smallest maximum distance to other points
  - Smallest average distance to other points
  - Smallest sum of squares of distances to other points
    - For distance metric ***d*** clustroid ***c*** of cluster ***C*** is: $\min_c \sum_{x \in C} d(x,c)^2$



Cluster on
3 datapoints

Centroid is the avg. of all (data)points in the cluster. This means centroid is an "artificial" point.
Clustroid is an **existing** (data)point that is "closest" to all other points in the cluster.

Defining “Nearness” of Clusters

- **(2) How do you determine the “nearness” of clusters?**
  - **Approach 2:**
    Intercluster distance = minimum of the distances between any two points, one from each cluster
  - **Approach 3:**
    Pick a notion of “**cohesion**” of clusters, *e.g.*, maximum distance from the clustroid
    - Merge clusters whose *union* is most cohesive

Cohesion

- **Approach 3.1:** Use the **diameter** of the merged cluster = maximum distance between points in the cluster
- **Approach 3.2:** Use the **average distance** between points in the cluster
- **Approach 3.3:** Use a **density-based approach**
  - Take the diameter or avg. distance, e.g., and divide by the number of points in the cluster
  - Perhaps raise the number of points to a power first, e.g., square-root

Implementation

- **Naïve implementation of hierarchical clustering:**
  - At each step, compute pairwise distances between all pairs of clusters, then merge
  - $O(N^3)$
- Careful implementation using priority queue can reduce time to $O(N^2 \log N)$
  - **Still too expensive for really big datasets that do not fit in memory**

k-means clustering

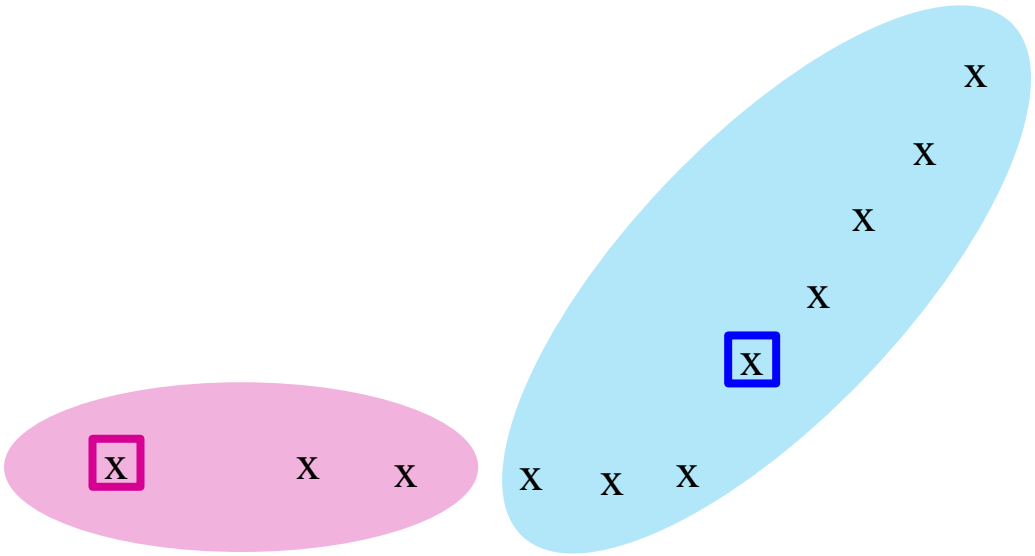
k–means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking ***k***, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then ***k*-1** other points, each as far away as possible from the previous points

Populating Clusters

- **1)** For each point, place it in the cluster whose current centroid it is nearest
- **2)** After all points are assigned, update the locations of centroids of the ***k*** clusters
- **3)** Reassign all points to their closest centroid
  - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

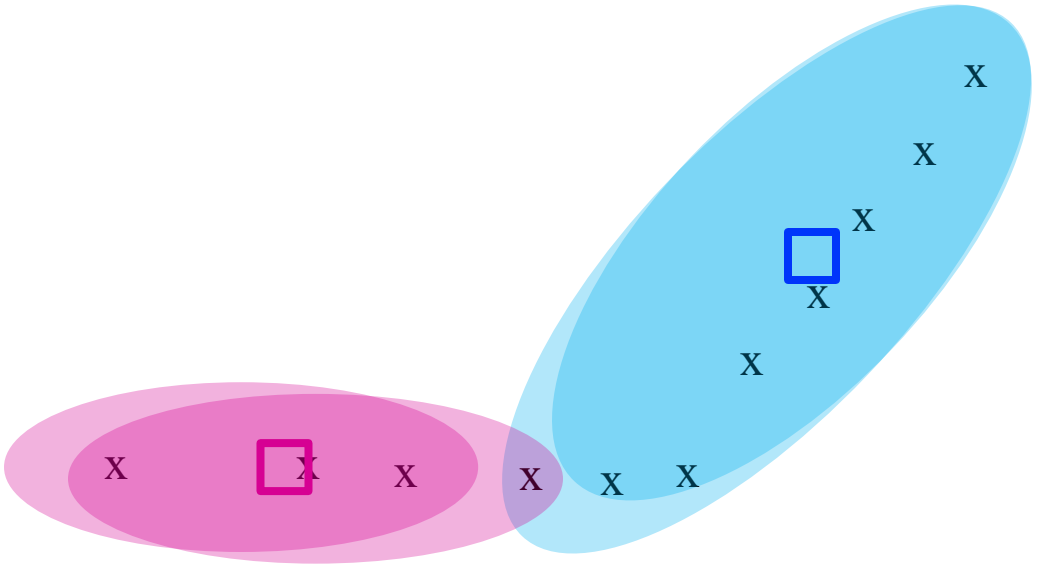
Example: Assigning Clusters



x … data point
□ … centroid

Clusters after round 1

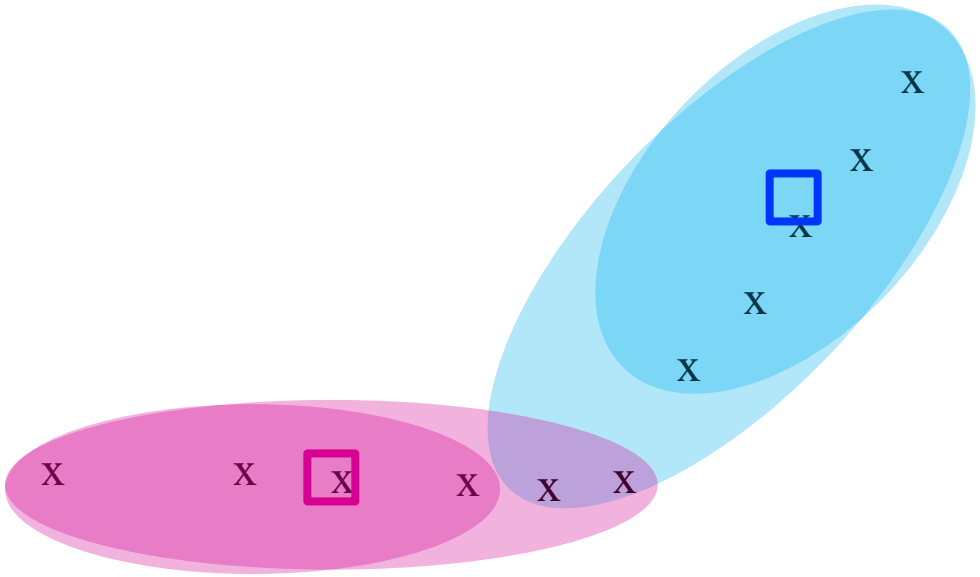
Example: Assigning Clusters



x … data point
□ … centroid

Clusters after round 2

Example: Assigning Clusters



x … data point
□ … centroid

Clusters at the end

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

Mean of the points in the cluster:

$$\mu(\mathrm{C}) = \frac{1}{|C|}\sum_{x \in C} x$$

K-means loss function

K-means tries to minimize what is called the “k-means” loss function:

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

that is, the sum of the squared distances from each point to the associated cluster center

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Does each step of k-means move towards reducing this loss function (or at least not increasing)?

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Sketch of proof/ argument:

1. Any other assignment would end up in a larger loss

1. The mean of a set of values minimizes the squared error

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Does this mean that k-means will always find the minimum loss/clustering?

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

NO! It will find *a minimum*.

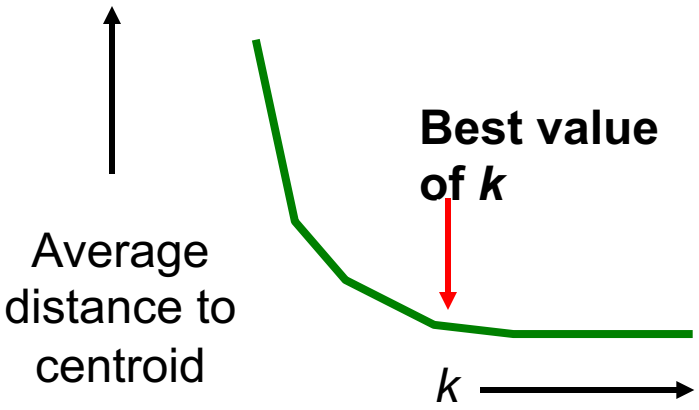
Unfortunately, the k-means loss function is generally not convex and for most problems has many, many minima

We're only guaranteed to find one of them
=> It's therefore important to try different random seeds

Getting the *k* right

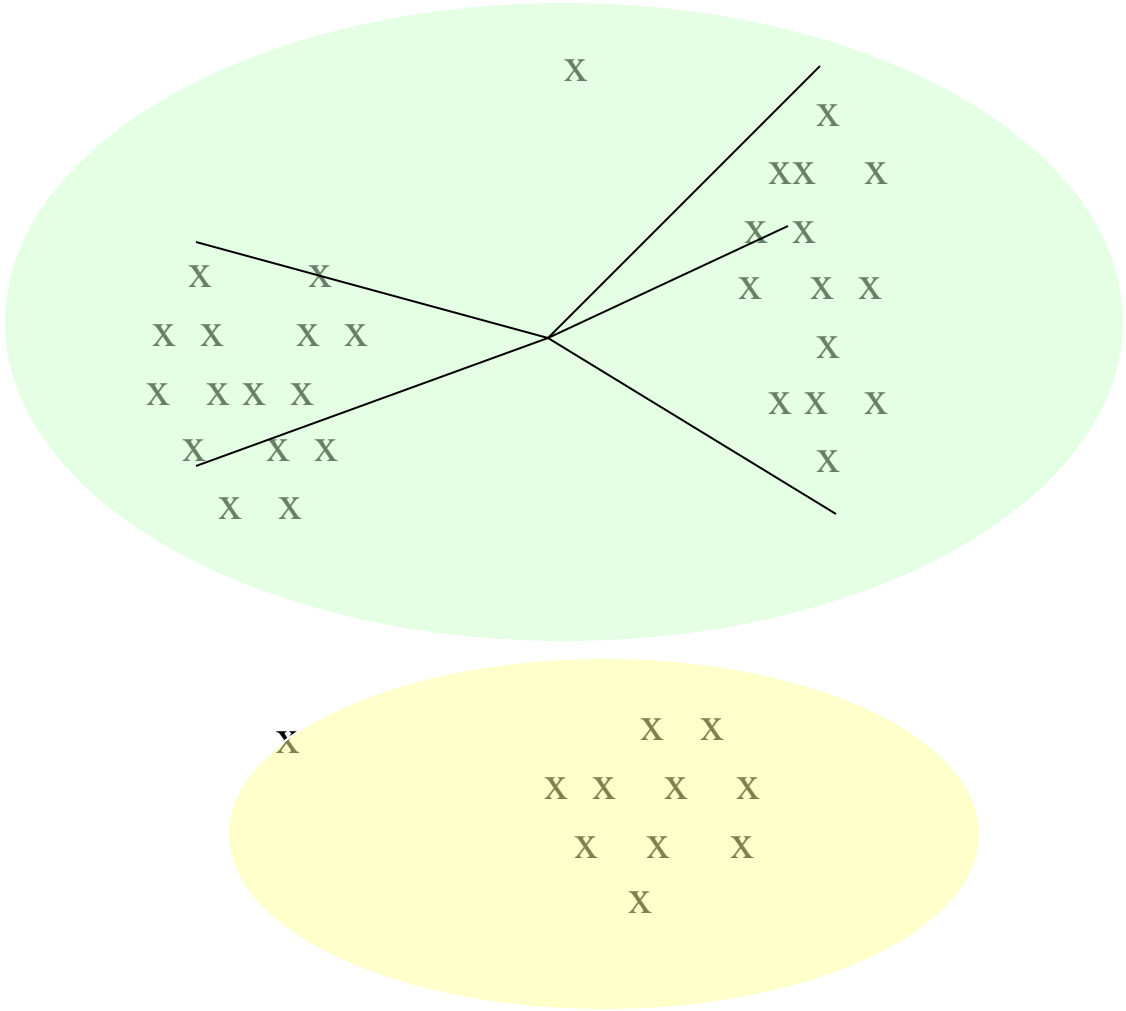
How to select *k*?

- Try different *k*, looking at the change in the average distance to centroid, as *k* increases.
- Average falls rapidly until right *k*, then changes little



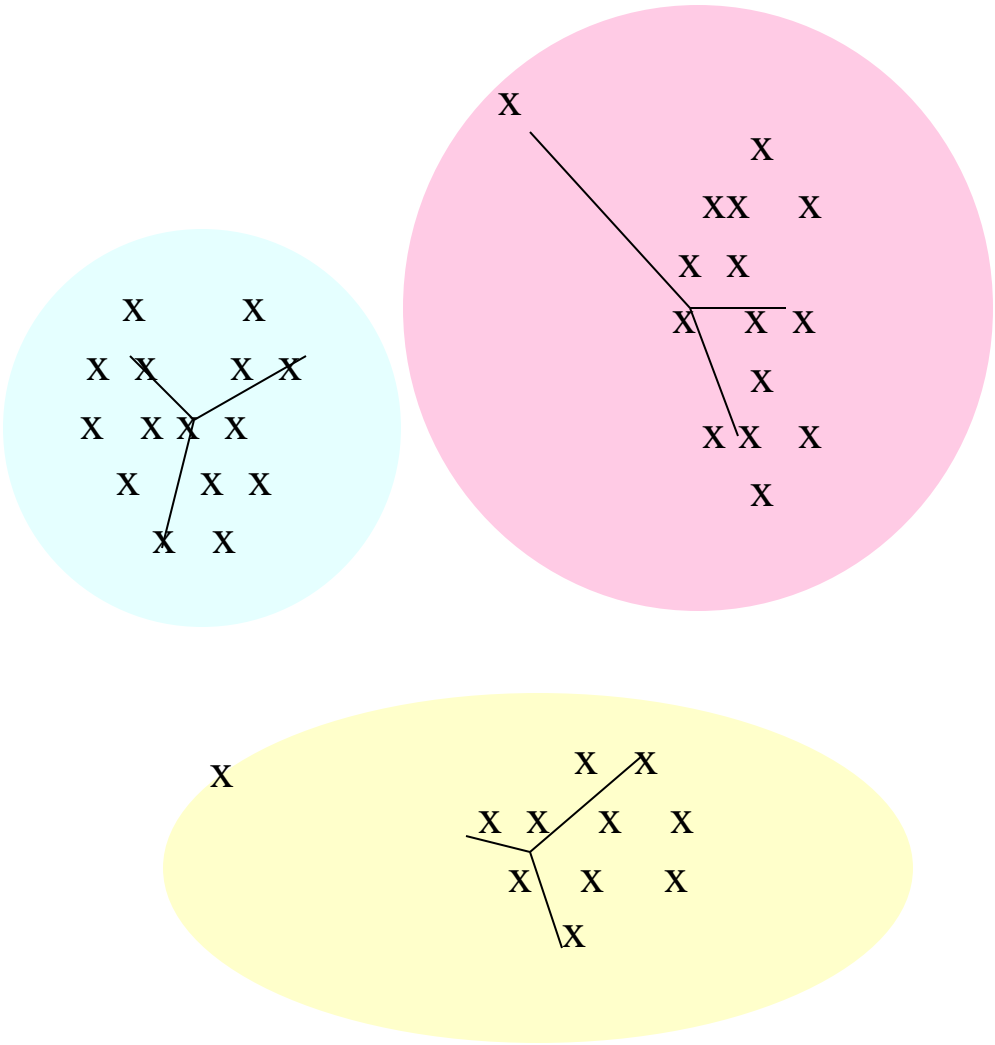
Example: Picking *k*

Too few;
many long
distances
to centroid.



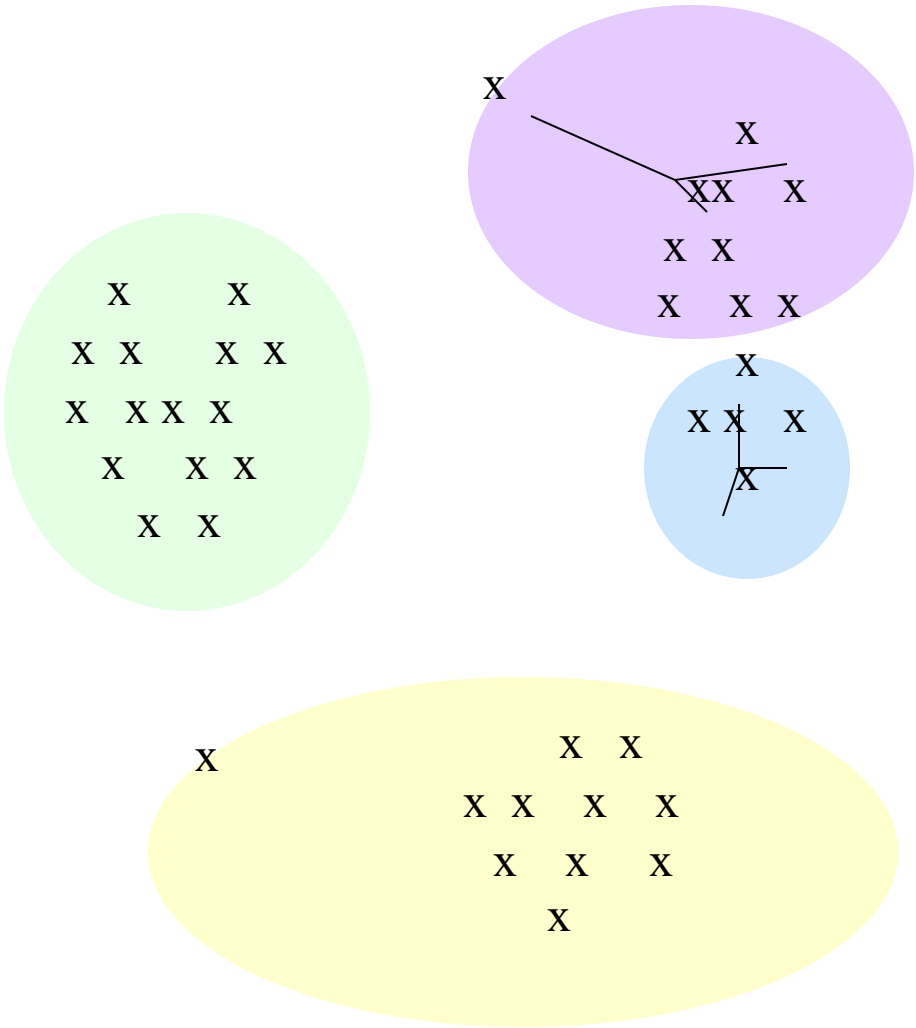
Example: Picking *k*

Just right;
distances
rather short.



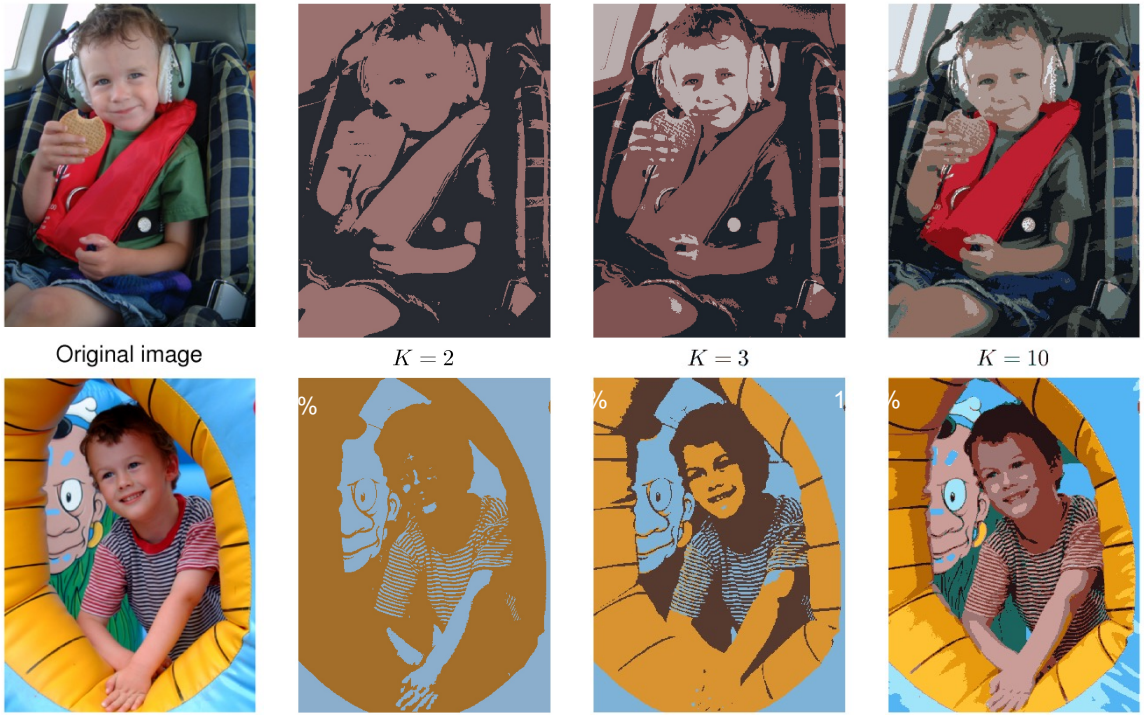
Example: Picking *k*

Too many;
little improvement
in average
distance.



Examples of k-means clustering

- Clustering RGB vectors of pixels in images
- Compression of image file: N x 24 bits
  - Store RGB values of cluster centers: K x 24 bits
  - Store cluster index of each pixel: N x log K bits



K-Means time complexity

Variables: *K* clusters, *n* data points,
m features/dimensions, *I* iterations

What is the runtime complexity?

- Computing distance between two points (e.g. Euclidean)
- Reassigning clusters
- Computing new centers
- Iterate…

K-Means time complexity

Variables: *K* clusters, *n* data points,
m features/dimensions, *I* iterations

What is the runtime complexity?

- Computing distance between two points is O*(m)* where *m* is the dimension of the vectors/number of features.
- Reassigning clusters: O*(Kn)* distance computations, or O*(Knm)*
- Computing centroids: Each points gets added once to some centroid: O*(nm)*
- Assume these two steps are each done once for *I* iterations: O*(IKnm)*

In practice, K-means converges quickly and is fairly fast

Limitations of K-means

- Need to determine "K" via domain knowledge or heuristics (as stated before)
- Only converge to local optimal
  - Need to try multiple starting points
  - Nice and Practical Research results on "Careful Randomized Seeding" in available, e.g. :
    - K-means++ and
    - K-means // (aka K-means parallel)

    Refer to the "More on K-means" supplementary lecture notes in course website.
- "Hard" assignment of each data point to a single cluster:
  - Each data point can only be assigned to 1 cluster (class)
  - What about points that lie in between groups ? e.g. Jazz + Classical
- Overall results can be affected by a few Outliners

Can we do better ?

The famous "GMM" – The Gaussian Mixture Model

Key Idea:

Assume Gaussian distribution of data points

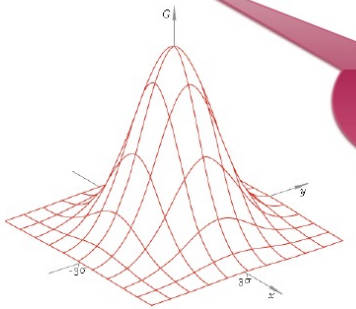
within each cluster

$$p(X) = N(X \mid \mu, \Sigma)$$

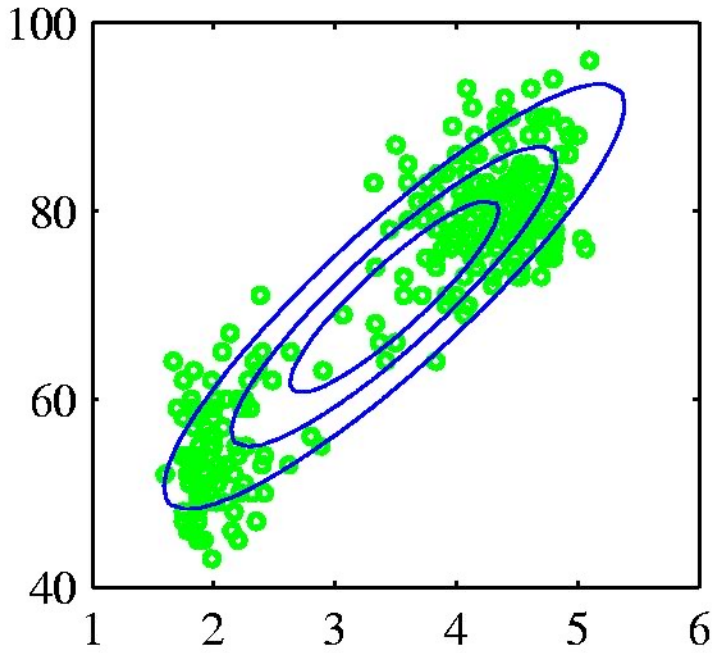
Mean

Variance

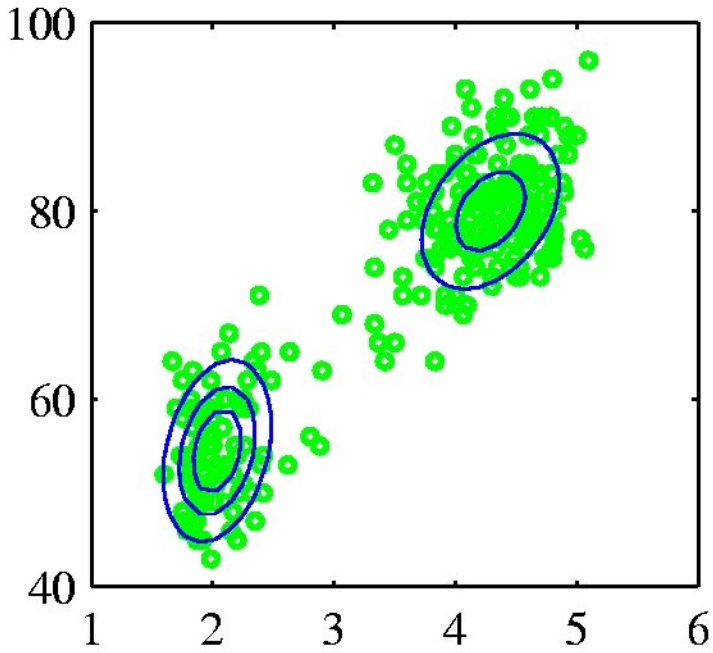
Gaussian == "Normal" distribution



- A sample data set



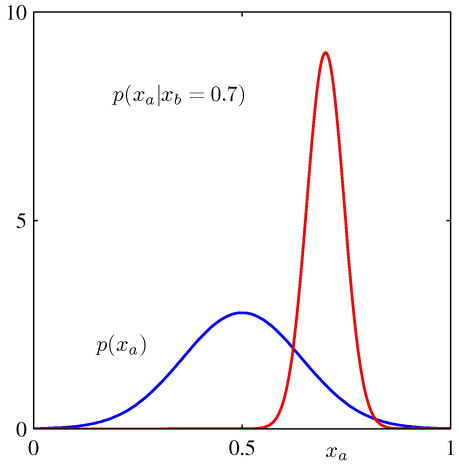
Single Gaussian

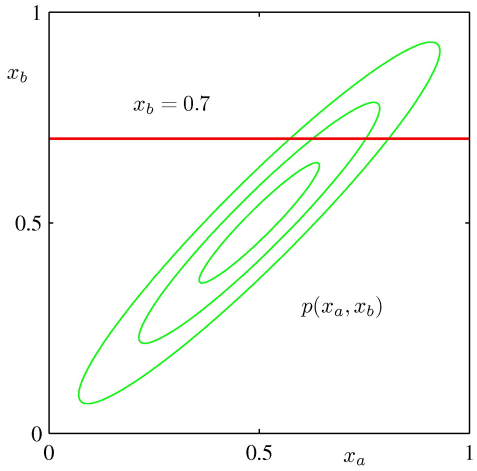


Mixture of two Gaussians

Clustering with Gaussian mixture density

- Each cluster represented by Gaussian density
  - Center, as in K-means
  - Covariance matrix: cluster spread around center





$$p(x) = N(x|\mu,\Sigma) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

Data dimension d

Determinant of covariance matrix Σ

Quadratic function of point x and mean μ

The Gaussian Mixture Model (GMM) (cont'd)

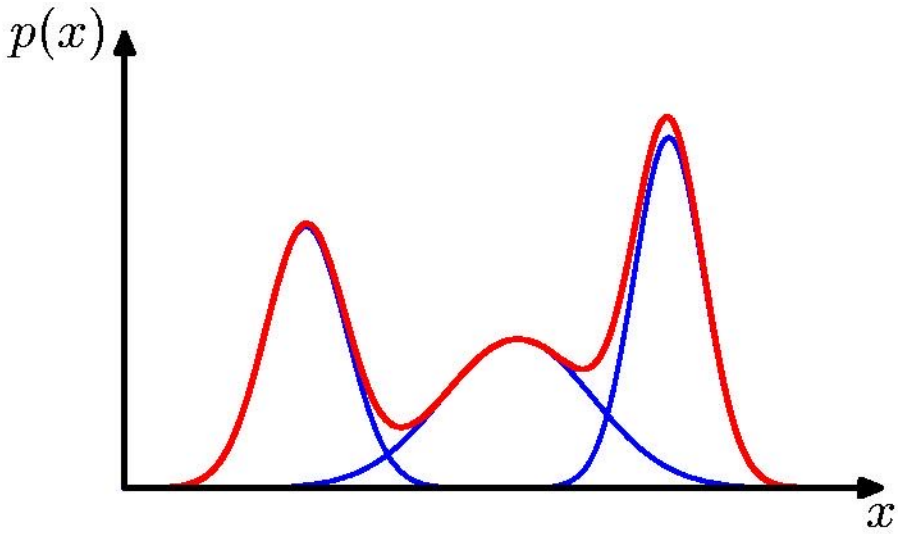
Combine simple models into a complex model:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

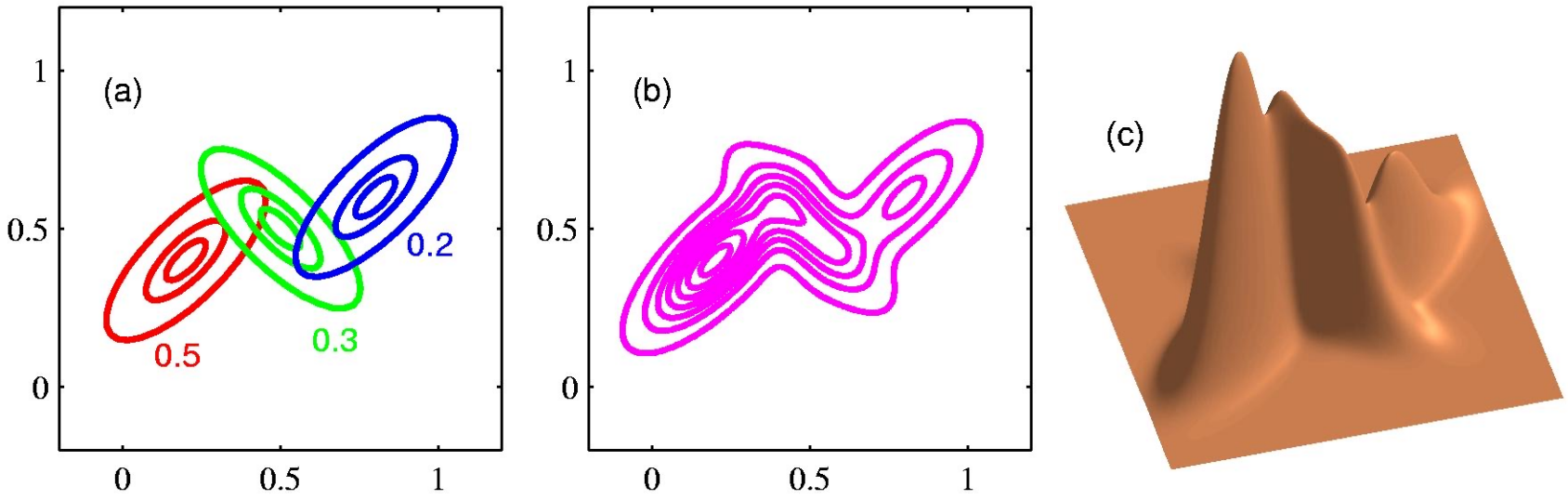
Component

Mixing coefficient: can be seen as the "contribution" from Gaussian component k

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



The Gaussian Mixture Model (GMM) (cont'd)

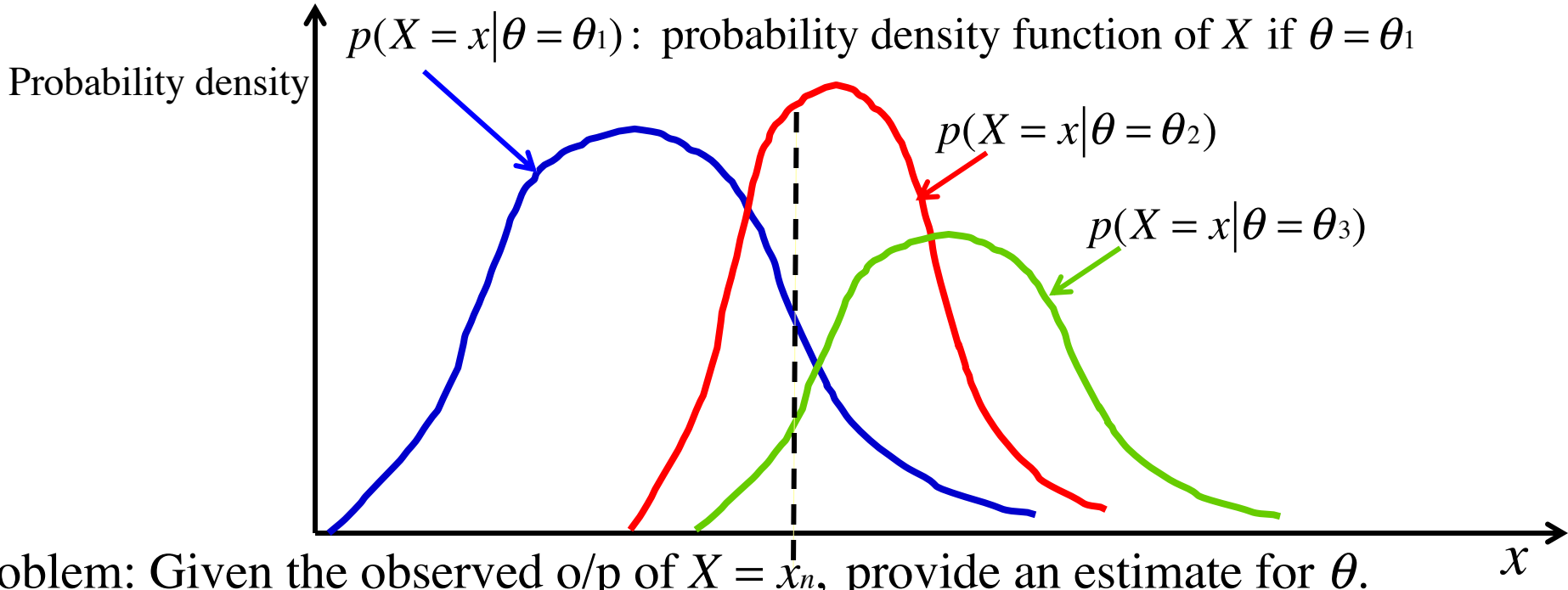


The Maximum Likelihood Estimator (MLE)

Input parameter: θ

A system to generate a r.v. X based on some probability distribution with parameter θ

Observed Output: $X = x_n$

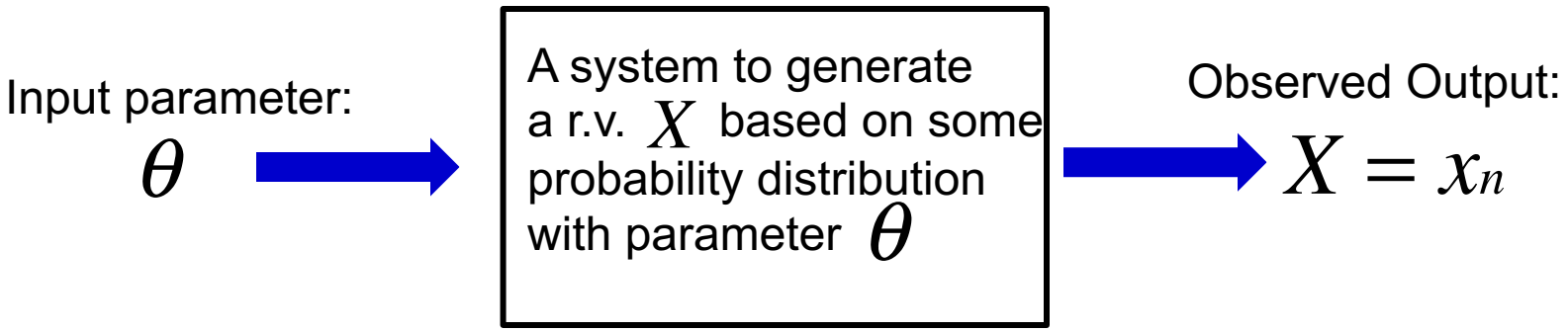


Problem: Given the observed o/p of $X = x_n$, provide an estimate for θ.

θ^*_{MLE} = Max. Likelihood Estimator (MLE) of $\theta = \arg\{\max_{\theta i}[p(X = x_n|\theta = \theta_i)]\}$

and in the above example, $\theta^*_{MLE} = \theta_2$

An Alternative Estimator: The Maximum A Posteriori (MAP) Estimator



$$\theta^*_{MAP} = \text{Max. a posteriori estimator (MAP) of } \theta = \arg\{\max_{\theta i}[p(\theta = \theta_i | X = x_n)]\}$$

$$= \arg\{\max_{\theta i}[\frac{p(X = x_n \text{ and } \theta = \theta_i)}{p(X = x_n)}]\} = \arg\{\max_{\theta i}[\frac{p(X = x_n | \theta = \theta_i)p(\theta = \theta_i)}{p(X = x_n)}]\}$$

$$= \arg\{\max_{\theta i}[p(X = x_n | \theta = \theta_i)p(\theta = \theta_i)]\}$$

The "true" probability distribution, $p(\theta)$, of the parameter, i.e. θ, to be estimated is the so-called prior distribution.

If no prior knowledge about the distribution of θ is available,

we can assume $p(\theta) \sim$ uniform distribution, i.e. $p(\theta = \theta_i) =$ some constant.

In this case, we have $\theta^*_{MAP} = \theta^*_{MLE}$

The Gaussian Mixture Model (GMM) (cont'd)

- Given the set of N data points represented by a matrix $\mathbf{X}$, we find the model parameters π , μ and Σ (which are vectors and a matrices) which maximize the probability of the occurrence of the observed data, via the Maximum (log) Likelihood Estimation (MLE) approach

$$p(X|\pi,\mu,\Sigma)=\prod_{n=1}^{N}p(x_n)=\prod_{n=1}^{N}\sum_{k=1}^{K}\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)$$

$$\ln p(\mathbf{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})=\sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right\}$$

Log of a sum; no closed form maximum

- Solution: use standard, iterative, numeric optimization methods or the *Expectation Maximization (EM)* algorithm (see next slides and Chapter 9 of PRML of C.M.Bishop).

How to Maximizing Log Likelihood for GMM

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Maximum of log likelihood:
derivatives of $\ln p(\mathbf{X}|\pi, \mu, \Sigma)$ w.r.t parameters to 0.

$$\gamma(z_{k,n}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \qquad N_k = \sum_{n=1}^{N} \gamma(z_{k,n})$$

$\gamma(z_{k,n})$ is called a "responsibility": how much is this Gaussian k responsible for this point X_n?

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) \mathbf{x}_n \qquad \pi_k^{\text{new}} = \frac{N_k}{N}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}}$$

The Gaussian Mixture Model (GMM) (cont'd)

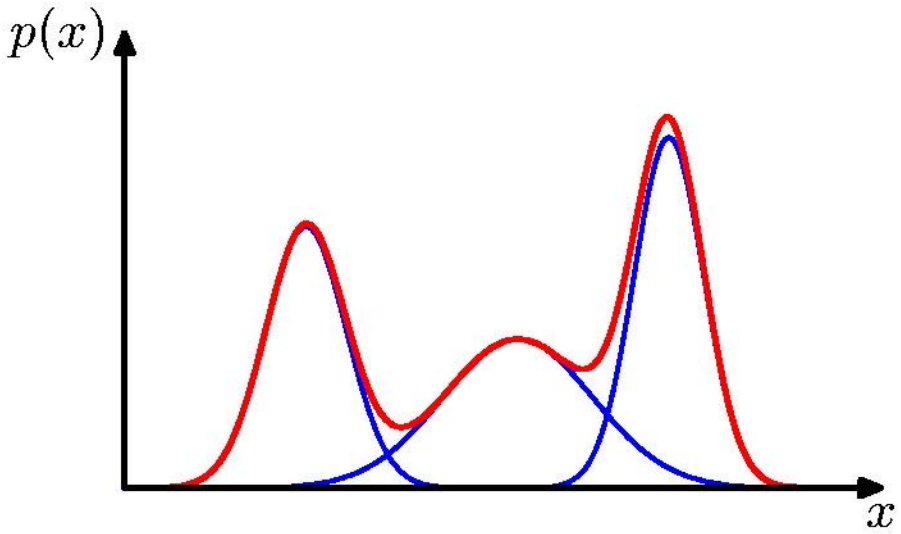
Combine simple models into a complex model:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Component

Mixing coefficient: can be seen as the "contribution" from Gaussian component k

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



Derivation of MLE for GMM

Maximum of log likelihood:
derivatives of $\ln p(\mathbf{X}|\pi,\mu,\Sigma)$ w.r.t parameters to 0.

$$\ln p(\mathbf{X}|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k) \right\}$$

For the μ_k:

Recall that $N(x|\mu,\Sigma) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \right)$

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k,\Sigma_k)}{\sum_k \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k,\Sigma_k)}}_{\gamma(z_{k,n})} \Sigma_k^{-1}(\mathbf{x}_n - \mu_k)$$

$$\mu_k = \frac{1}{\sum_n \gamma(z_{k,n})} \sum_n \gamma(z_{k,n}) \mathbf{x}_n$$

$\gamma(z_{k,n}) \equiv \text{Prob}(z_{k,n} = 1 \mid x_n)$ where $z_{k,n} = 1$ iff the n^{th} data point x_n is generated by cluster k

$N_k \equiv \sum_n \gamma(z_{k,n}) =$ effective number of data points in the k^{th}cluster.

Derivation of MLE for GMM (cont'd)

For Σ_k:

$$\Sigma_k = \frac{1}{\sum_n \gamma(z_{kn})} \sum_n \gamma(z_{kn})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T$$

For the π_k:

- Take into account constraint $\sum_k \pi_k = 1$
- Lagrange multiplier

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) + \lambda(\sum_k \pi_k - 1)$$

$$0 = \sum_n \frac{\mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_k \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)} + \lambda$$

Multiply by π_k and summing over k, we get $\lambda = -N \Rightarrow \pi_k = \frac{\sum_n \gamma(z_{kn})}{N}$

The Gaussian Mixture Model (GMM)

- Given the set of N data points represented by a matrix $\mathbf{X}$, we find the model parameters π , μ and Σ (which are vectors and a matrices) which maximize the probability of the occurrence of the observed data, via the Maximum (log) Likelihood Estimation (MLE) approach

$$p(X|\pi,\mu,\Sigma) = \prod_{n=1}^{N} p(x_n) = \prod_{n=1}^{N}\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)$$

$$\ln p(\mathbf{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \right\}$$

Log of a sum; no closed form maximum

- Solution: use standard, iterative, numeric optimization methods or the *Expectation Maximization (EM)* algorithm (see next slides and Chapter 9 of PRML of C.M.Bishop).

Summary: How to Maximizing Log Likelihood for GMM

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Maximum of log likelihood:
derivatives of $\ln p(\mathbf{X}|\pi, \mu, \Sigma)$ w.r.t parameters to 0.

$$\gamma(z_{k,n}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \qquad N_k = \sum_{n=1}^{N} \gamma(z_{k,n})$$

$\gamma(z_{k,n})$ is called a "responsibility": how much is this Gaussian k responsible for this point X_n?

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) \mathbf{x}_n \qquad \pi_k^{\text{new}} = \frac{N_k}{N}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}}$$

Derivation of MLE for the Mixture of Gaussian model (cont'd)

- No closed form solutions: $\gamma(z_{kn})$ depends on parameters
- But these equations suggest simple iterative scheme for finding maximum likelihood:
  Alternate between estimating the current $\gamma(z_{kn})$ and updating the parameters $\{\mu_k, \Sigma_k, \pi_k\}$.

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

1. Initialize the parameters: means μ_k , covariances $\sum_k$ and mixing coeff. π_k of the K Gaussian components.
2. E Step: Assign each point x_n an assignment score $\gamma(z_{k,n})$ for each cluster k
3. M Step: Given scores, adjust $\mu_k, \sum_k, \pi_k$ for each cluster k
4. Evaluate Likelihood value. If likelihood value or parameters converge, stop ; Otherwise Goto Step 2. (the E Step)

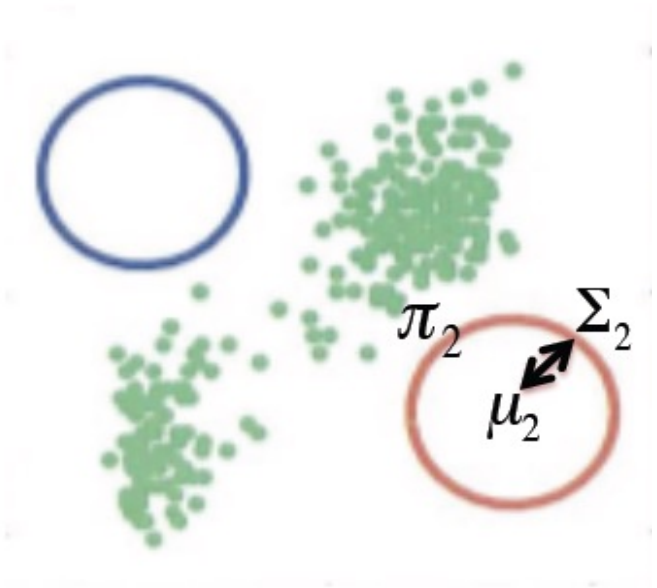
where

$z_{k,n}$ is the 0-1 indicator r.v. showing whether x_n belongs to cluster k

$\gamma(z_{k,n})$ is an estimate of the posterior probability that data point x_n is "contributed" by cluster k , i.e. the conditional probability that x_n belongs to cluster k given the value of x_n and parameters $\mu_k, \sum_k, \pi_k$

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

1. Initialize μ_k , Σ_k π_k , one for each Gaussian k



- Tip! Use K-means result to initialize:

$$\mu_k \leftarrow \mu_k$$

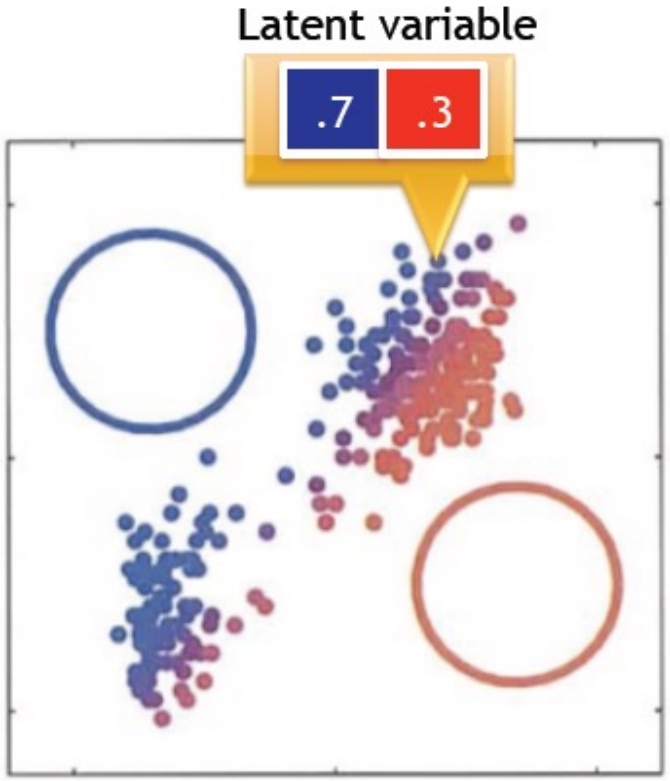
$$\Sigma_k \leftarrow \text{cov}(cluster(K))$$

$$\pi_k \leftarrow \frac{\text{Number of points in k}}{\text{Total number of points}}$$

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

2. **E Step:** For each point X_n, determine its assignment score to each Gaussian k:

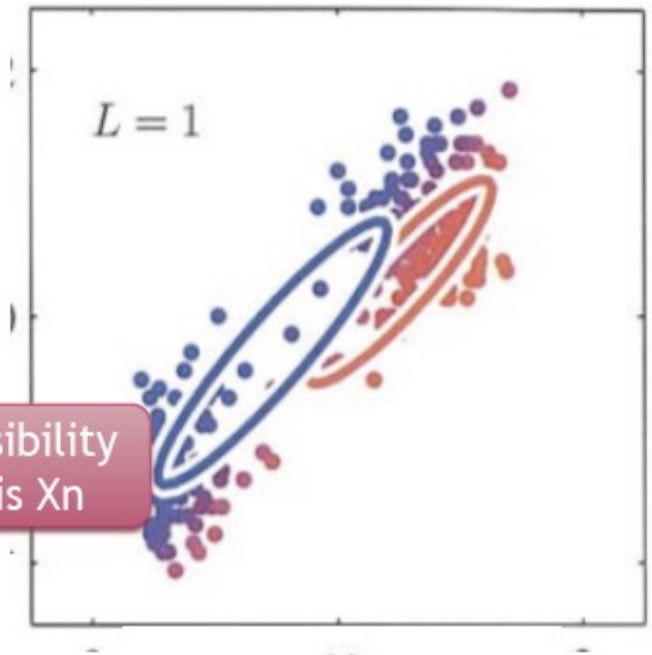
$$\gamma(z_{k,n}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$



$\gamma(z_{k,n})$ is called a "responsibility": how much is this Gaussian k responsible for this point X_n?

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

3. **M Step:** For each Gaussian k, update parameters using new $\gamma(z_{k,n})$



Responsibility for this Xn

Mean of Gaussian k

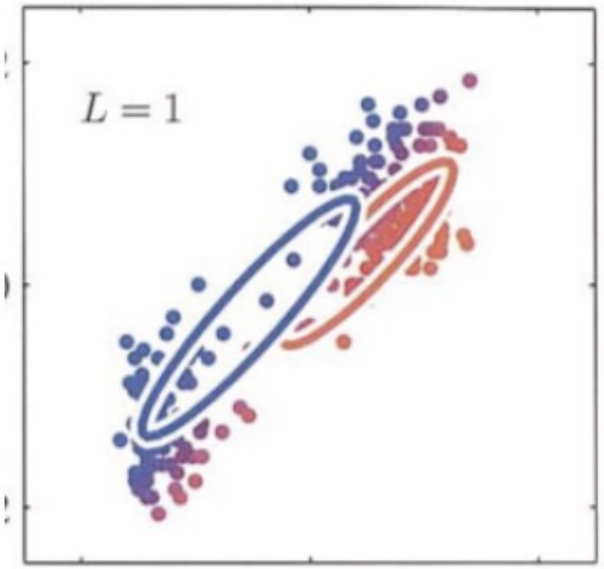
$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) \mathbf{x}_n$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{k,n})$$

Find the mean that "fits" the assignment scores best

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

3. **M Step:** For each Gaussian k, update parameters using new $\gamma(z_{k,n})$



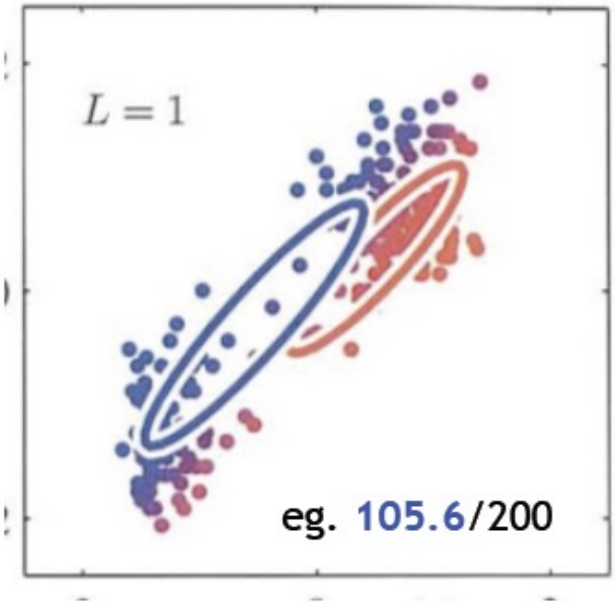
Covariance matrix of Gaussian k

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\text{T}}$$

Just calculated this!

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

3. **M Step:** For each Gaussian k, update parameters using new $\gamma(z_{k,n})$



Mixing Coefficient for Gaussian k

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

Total # of points

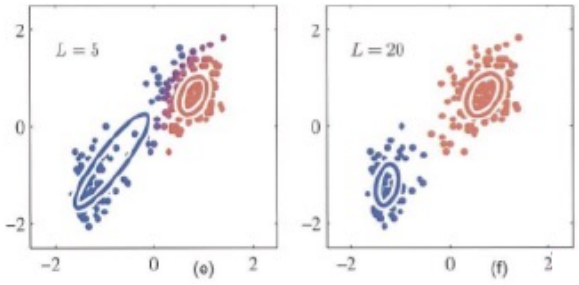
$$N_k = \sum_{n=1}^{N} \gamma(z_{k,n})$$

Expectation Maximization (EM) approach for the Gaussian Mixture Model (GMM)

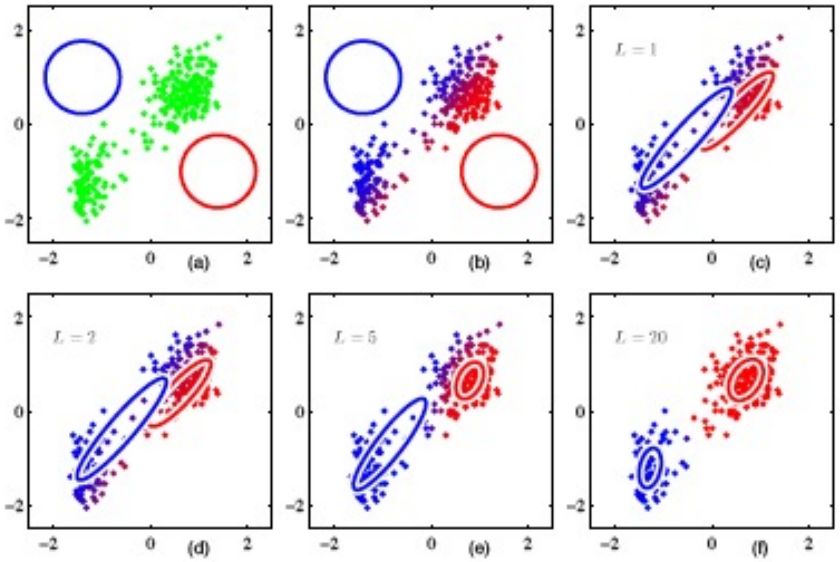
4. Evaluate log **likelihood**. If likelihood or parameters converge, stop. Else go to Step 2 (**E step**).

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Likelihood is the probability for the observed data-set X to occur (i.e to be generated) given the current values of the parameters.



The EM approach for GMM (cont'd)



- More iterations needed to converge than K-means algorithm, and each cycle requires more computation
- Common, initialise parameters based K-means run.

Recap: EM for GMM

The EM Algorithm

1. Initialize the parameters: means μ_k , covariances Σ_k and mixing coeff. π_k of the K Gaussian components.
2. E Step: Assign each point x_n an assignment score $\gamma(z_{k,n})$ for each cluster k
3. M Step: Given scores, adjust μ_k, Σ_k, π_k for each cluster k
4. Evaluate Likelihood value. If likelihood value or parameters converage, stop ; Otherwise Goto Step 2. (the E Step)

$\gamma(z_{k,n})$ is an estimate of the posterior probability that data point x_n is "contributed" by (or belongs to) cluster k, i.e. the conditional probability given the parameters μ_k, Σ_k, π_k and the observation x_n

Comparing to the K-means algorithm

1. Initialize means μ_k
2. E Step: Assign each point to a cluster
3. M Step: Given clusters, refine mean μ_k of each cluster k
4. Stop when change in means is small

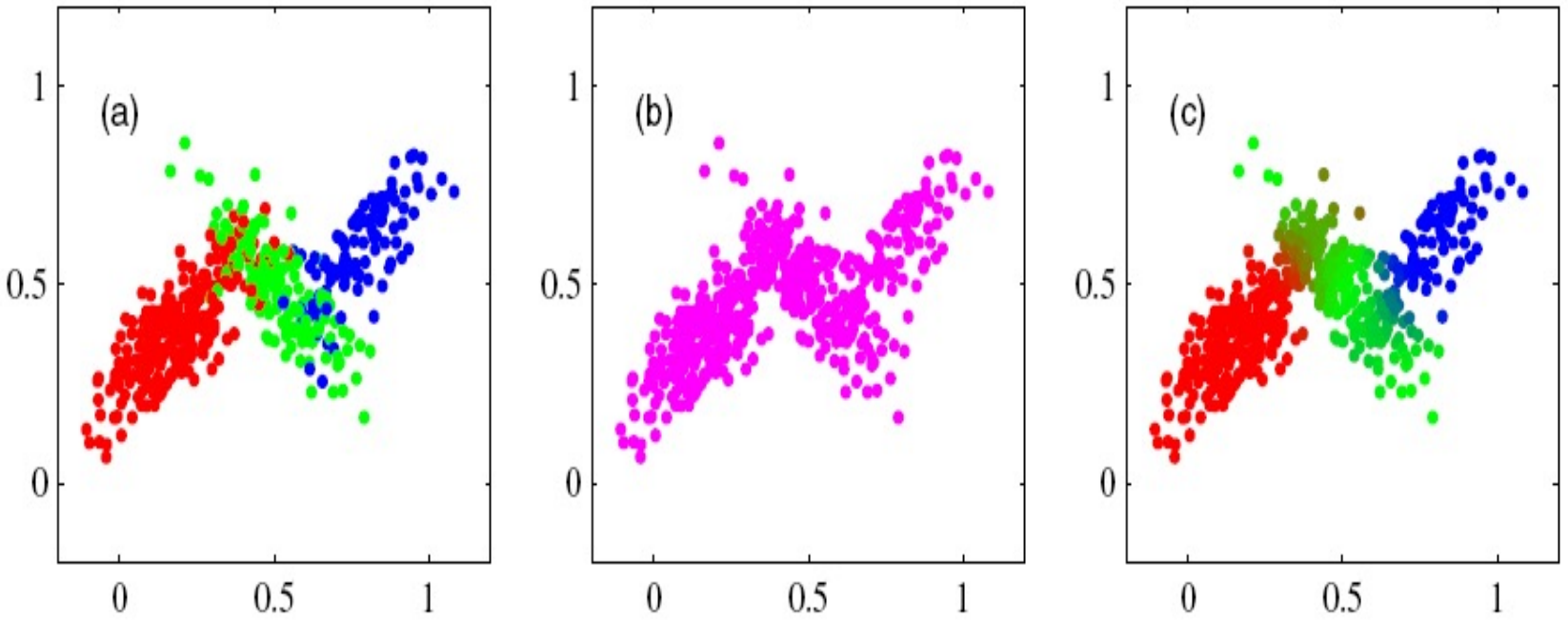


Figure 9.5 Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ in which the three states of $\mathbf{z}$, corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution $p(\mathbf{x})$, which is obtained by simply ignoring the values of $\mathbf{z}$ and just plotting the $\mathbf{x}$ values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point $\mathbf{x}_n$, obtained by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively

Application: Using GMM for Image Segmentation

Source:
https://kittipatkampa.wordpress.com/2011/02/17/image-segmentation-using-gaussian-mixture-models/

Original Image

Segmentation results using GMM with 3 components

Input Features:

x-y pixel locations & pixel lightness/color in L*a*b color space

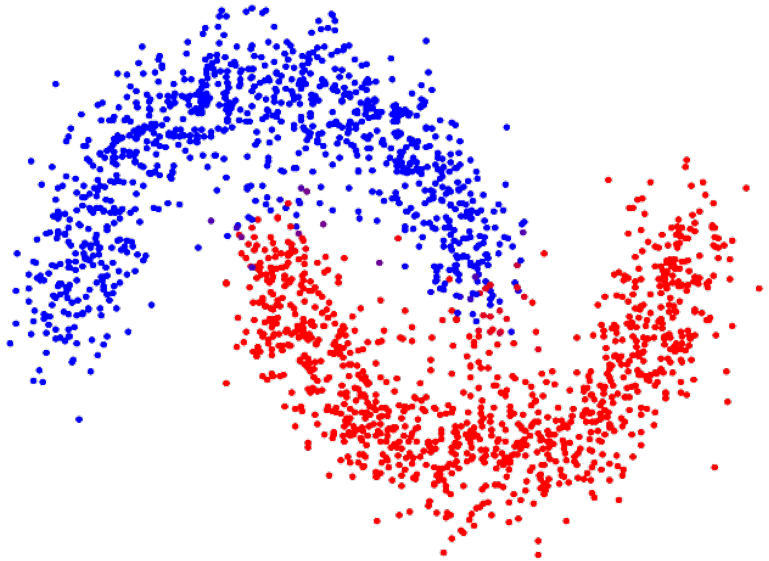
Output Results:

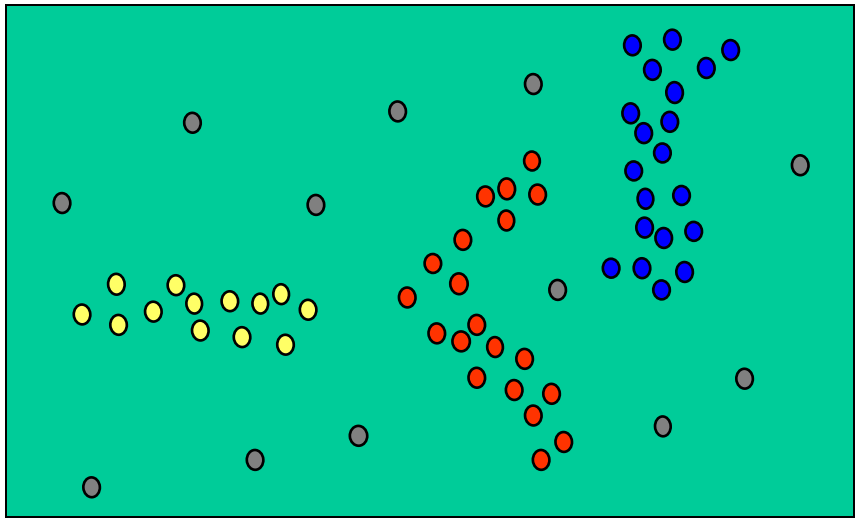
Each color represents a class ; The brightness represents the posterior probability – darker pixels represent high uncertainty of the posterior distribution.

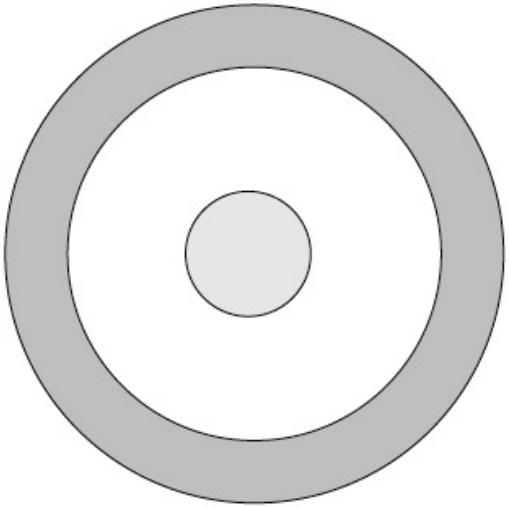
From K-means to GMM

- K-means is a 0-1 classifier, which assigns every data point to one and only one cluster
  - As shown previously, K-means can also be formulated as EM
- Mixture of Gaussians is a Probability Model to describe/characterize a given set of data
  - GMM can be used as a "Soft" classifier
  - For every point we can quantify the **likelihood** that it belongs to a particular cluster
  - Once you have established the model for a given data set, you can use it for other applications, e.g. Prediction, Density Estimator, statistical inference etc.
- The EM-approach can be generalized to other non-Gaussian distributions.
- K-means, GMM and the general EM-approach are all widely used in practice for large-scale problems.

K-means and GMM CANNOT deal with Clusters of Random Shapes !!



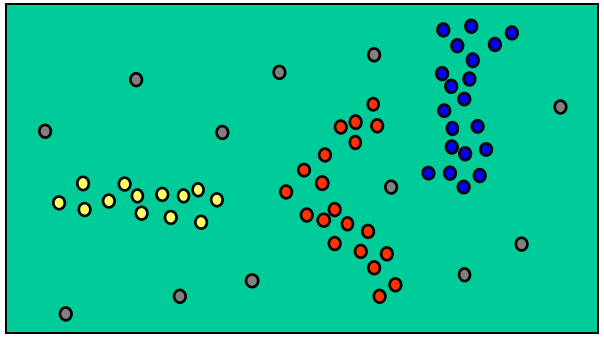




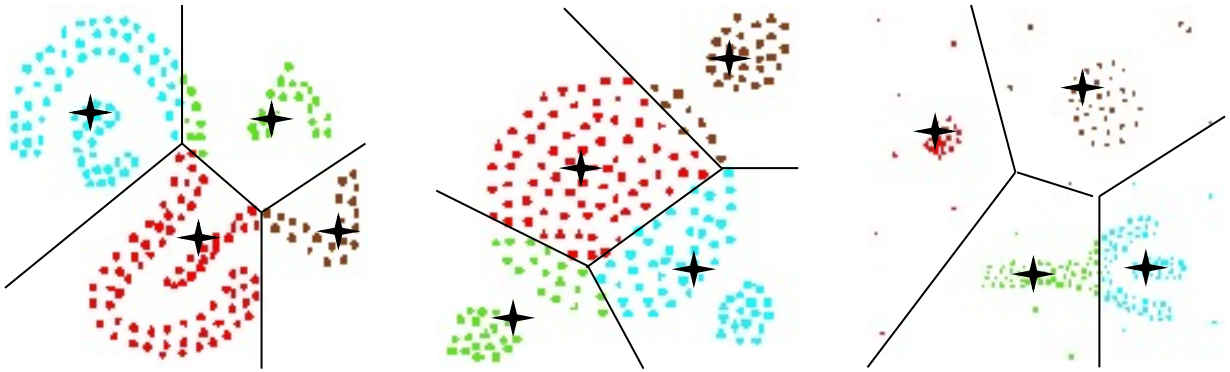
Density-based Clustering

- *Basic Idea*:

  Clusters are dense regions in the data space, separated by regions of lower object density



Why Density-Based Clustering?



Results of a *k*-medoid algorithm for *k*=4

Density-based Clustering

- Density-based Clustering locates regions of high density that are separated from one another by regions of low density.
- Density = number of points within a specified radius (ε) (in d-dimensional space)
- Why Density-Based Clustering methods?
  - Discover clusters of arbitrary shape.
  - Clusters – Dense regions of objects separated by regions of low density
- DBSCAN – the first density based clustering
  - To be covered in ESTR4300 ! All are welcome

Backup Slides

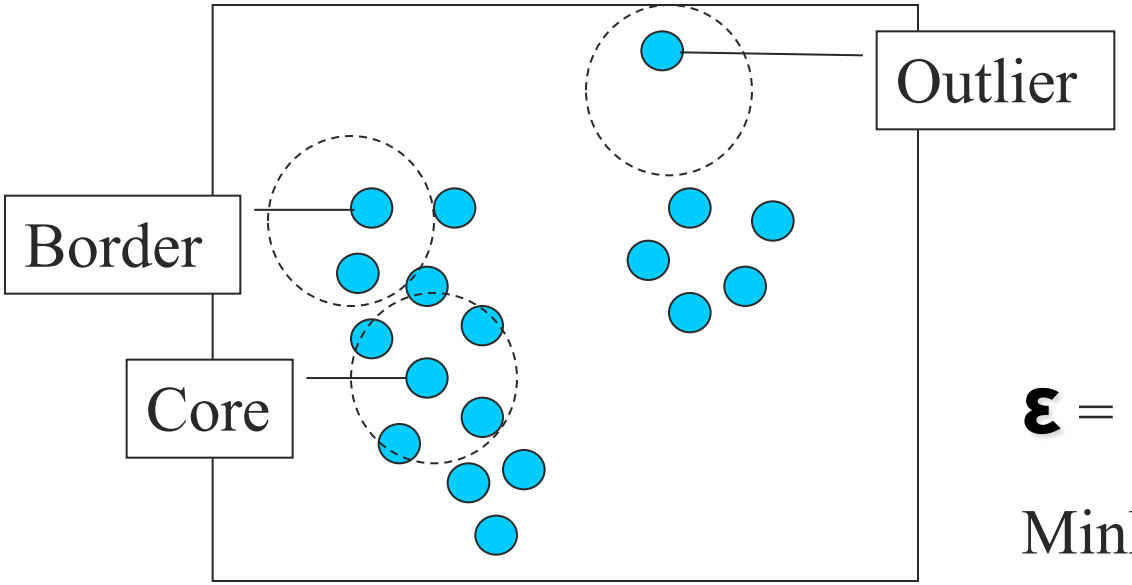
DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Proposed by Ester, Kriegel, Sander, and Xu (KDD1996) – 2014 SIGKDD Test of Time Award
- Relies on a density-based notion of cluster: A cluster is defined as a maximal set of density-connected points.
- Discovers clusters of arbitrary shape in spatial databases with noise

DBSCAN Terminology

- A point is a core point if it has more than a specified number of points (MinPts) within a radius of **ε**
  - These are points that are at the interior of a cluster
- A border point has fewer than MinPts within **ε**, 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.

Border & Core



ε = 1 unit

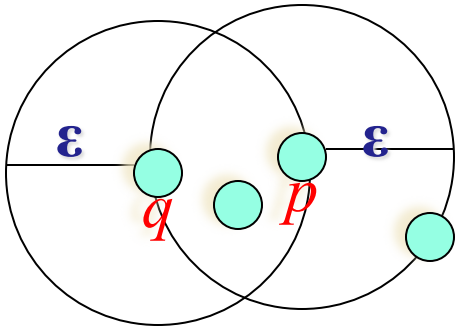
MinPts = 5

Key Idea for Cluster Formation under DBSCAN

- Any two core points are close enough– within a distance ε of one another – are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

Concepts: ε-Neighborhood

- **ε-Neighborhood** - Points within a radius of ε from a point. (epsilon-neighborhood)
- "High density" - ε-Neighborhood of a point contains at least **MinPts** of points



ε-Neighborhood of *p*

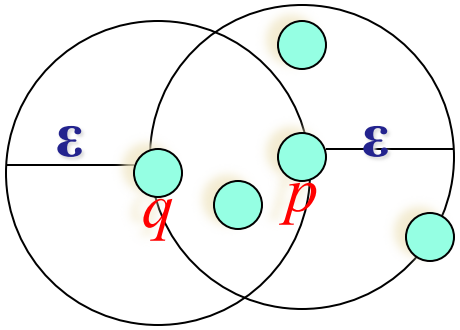
ε-Neighborhood of *q*

Density of *p* is "high" (w.r.t. MinPts = 4)

Density of *q* is "low" (w.r.t. MinPts = 4)

Concepts: Reachability

- **Directly Density-Reachable**
  - A point q is directly density-reachable from a point p if q is within the ε-Neighborhood of p and p is a core point.

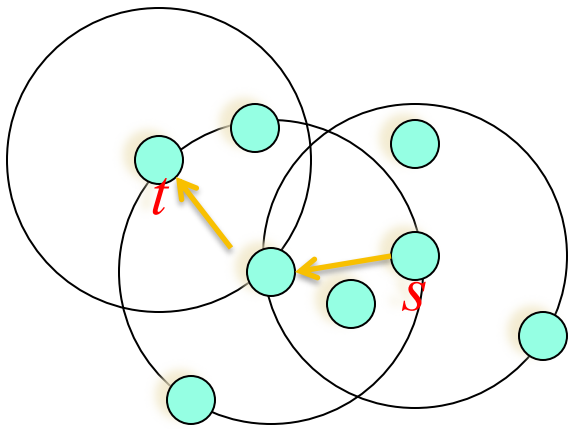


- q is directly density-reachable from p
- p is not directly density-reachable from q !

=> Asymmetric in general

Concepts: Reachability

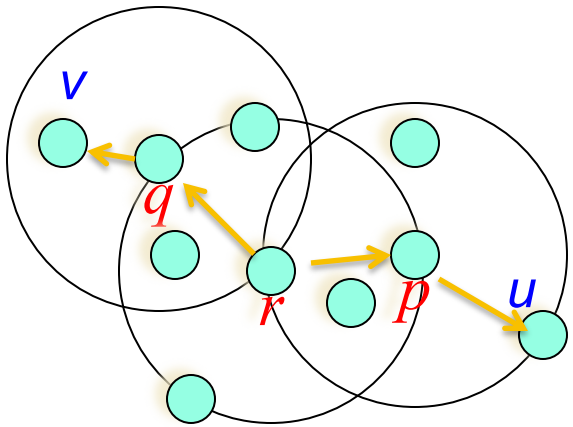
- **Density-Reachable:**
  - A point *q* is density-reachable from *p* w.r.t ε and *MinPts* if there is a chain of points $p_1,\ldots,p_n$, with $p_1=p$, $p_n=q$ such that p_{i+1} is directly density-reachable from p_i w.r.t ε and *MinPts* for all $1 <= i <= n$



- *t* is density-reachable from *s*
- *BUT s* is not density-reachable from *t* !
- Transitive closure of Direct Density-Reachability ; Asymmetric in general !

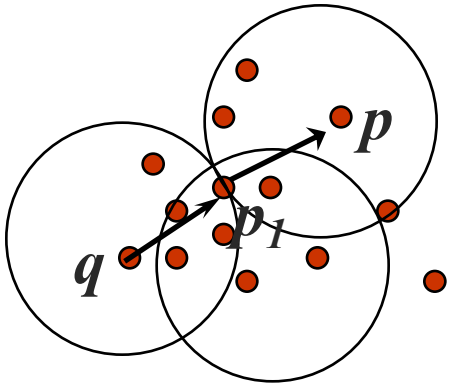
Concepts: Connectivity

- **Density-connectivity**
  - Point *p* is density-connected to Point *q* w.r.t ε and *MinPts* if there is a point *r* such that both *p* and *q* are density-reachable from *r* w.r.t ε and *MinPts*

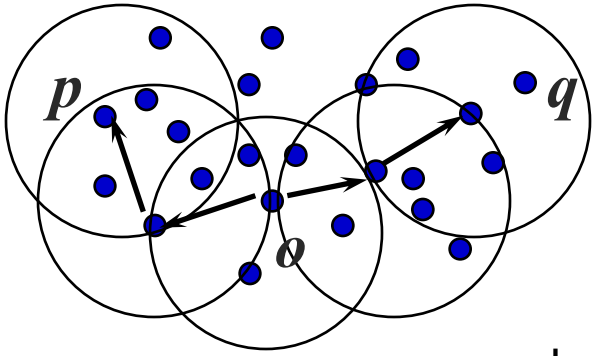


- *p* and *q* are density-connected to each other by *r*
- *v* and *u* are density-connected to each other by *r*
- Density-connectivity is symmetric

Density-reachable vs. Density-Connected



p is Density-reachable from q



p and q are Density-connected via o

Formal Description of a Cluster

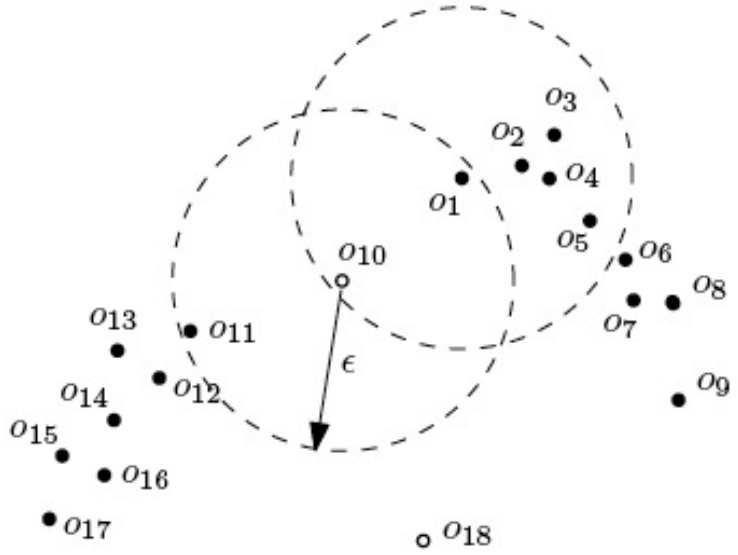
- Given a data set D, parameter ε and a threshold of MinPts.
- A cluster C is a subset of points satisfying two criteria:
  - *Connected:* For all p,q in C: p and q are density-connected.
  - *Maximal:* For all p,q: if p in C and q is density-reachable from p, then q in C (=> p is a core point)

  => Each cluster has at least 1 Core point

  => Each cluster contains at least MinPts points
- Note: a Cluster contains both Core and Border points
- **Noise:** points which are not directly density-reachable from at least one core point.
- The set of Clusters defined as above is ALWAYS Unique

  !...except…=> border point may belong to multiple cluster

Cluster Examples



C1 = {o1, o2,…o10}
C2 = {o10, o11,…,o17}
o18 is a noise point

Figure 2: An example dataset (the two circles have radius ϵ; $MinPts = 4$)

Connected: For all p,q in C: p and q are density-connected.

Maximal: For all p,q: if p in C and q is density-reachable from p, then q in C ;
also => each cluster has at least 1 Core point

- {o1, o10} alone is NOT a cluster by itself because NOT Maximal
- A Border point, e.g. o10, can belong to MULTIPLE clusters
- Clusters produced by DBSCAN is NOT necessarily disjoint !
- However, if p belongs to more than 1 cluster, it MUST BE a Border point
  ⇔ A core point always belongs to a Unique cluster, Why ?

DBSCAN: The Algorithm (Simplified)

```
for each o ∈ D do        /* Result is independent of the order of
                            processing the points EXCEPT… */
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

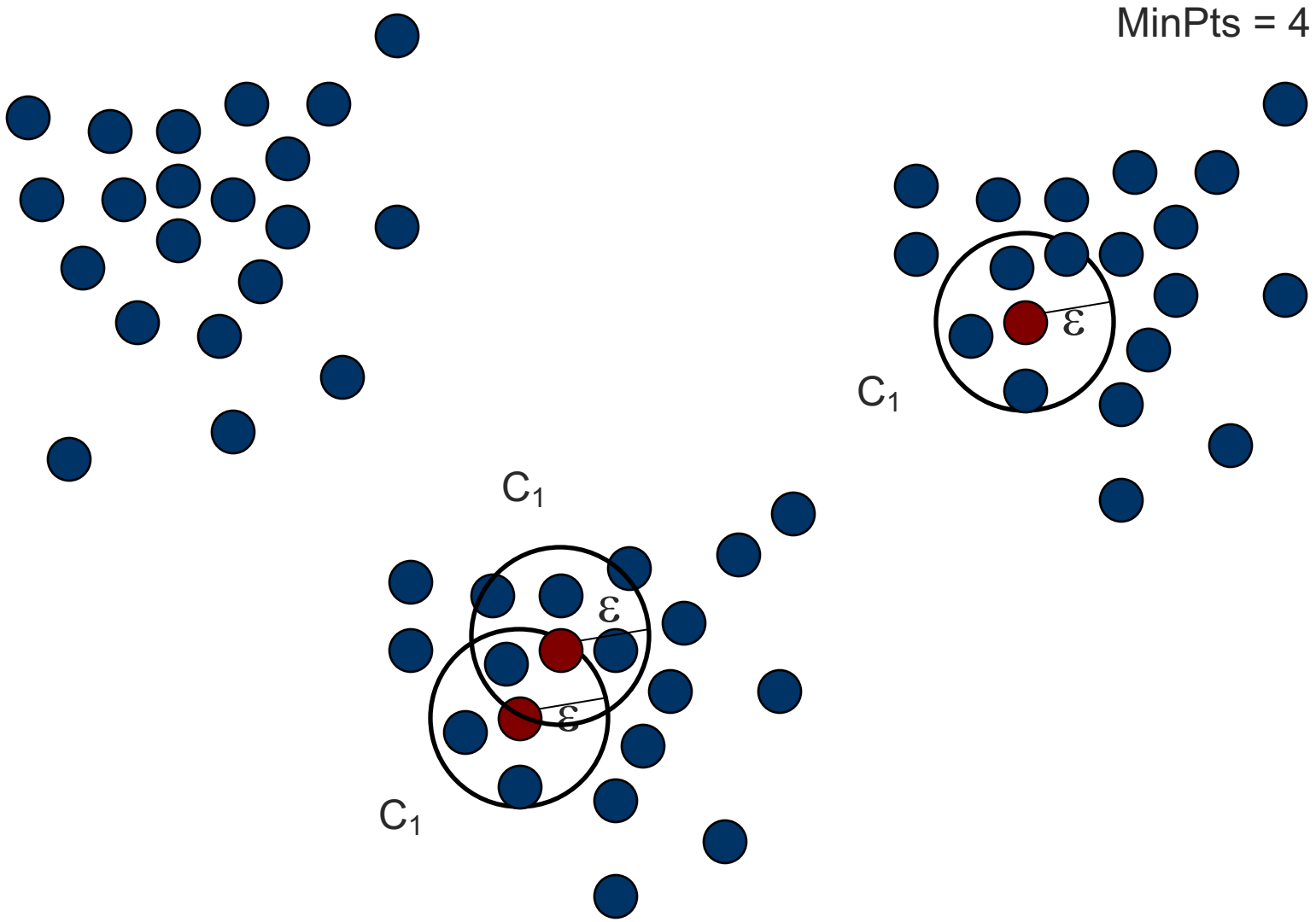
- Worst-case Time Complexity = $O(n^2)$! (trivial ?)
  - NOT O(n log n) Time as initially mis-claimed (for 17 years) !
- Time complexity = $\Omega(n^{4/3})$ for d > 2 [Gan&Tao 2015] ;
- O(n) space complexity
- O(n log n) Time only for 2-dimensional case [Gunawan 2013] ;
- ρ-approx DBSCAN runs in O(n) Time in Expectation for all dimensions [Gan&Tao 2015]

DBSCAN: The Algorithm (a more detail version)

```
DBSCAN(D, eps, MinPts)
   C = 0
   for each unvisited point P in
   dataset D
      mark P as visited
      N = regionQuery(P, eps)
      if sizeof(N) < MinPts
         mark P as NOISE
      else
         C = next cluster
         expandCluster(P, N, C,
                       eps, MinPts)
```

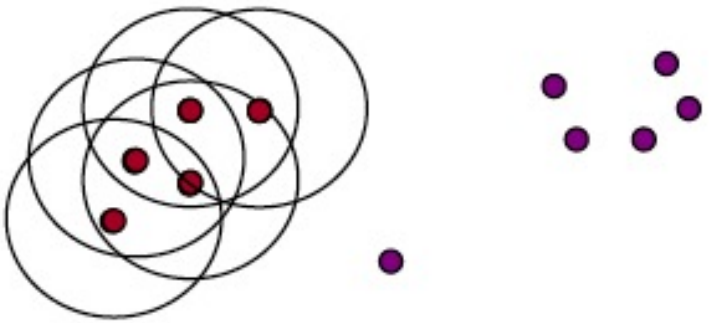
```
expandCluster(P,N,C,eps,MinPts)
   add P to cluster C
   for each point P' in N
      if P' is not visited
         mark P' as visited
         N' = regionQuery(P', eps)
         if sizeof(N') >= MinPts
            N = N joined with N'
      if P' is not yet member of
        any cluster
         add P' to cluster C
```

An Example



Running the DBSCAN Algorithm: An Example

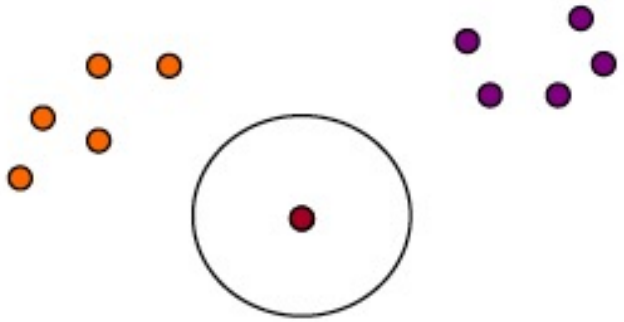
- **Parameter**
  - $\varepsilon = 2$ cm
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

Running the DBSCAN Algorithm: An Example

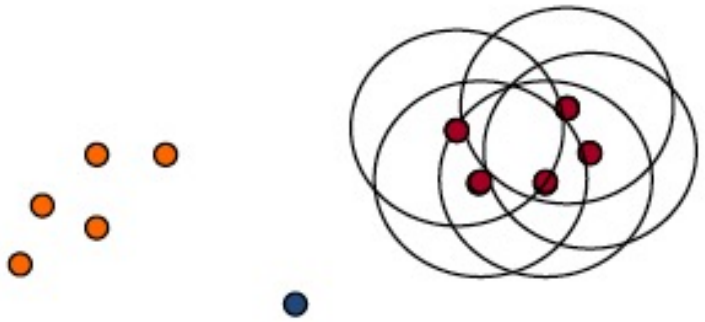
- **Parameter**
  - $\varepsilon = 2$ cm
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

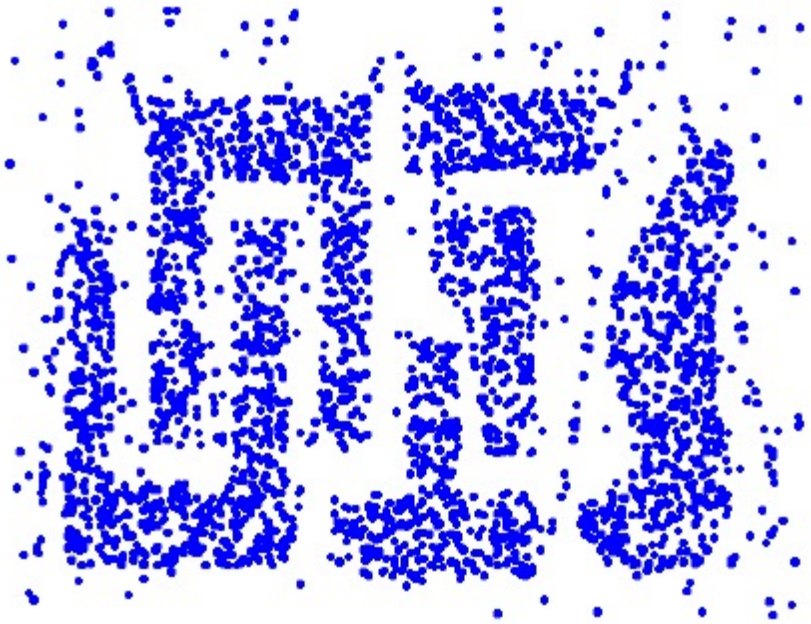
Running the DBSCAN Algorithm: An Example

- **Parameter**
  - $\varepsilon = 2$ cm
  - *MinPts* = 3

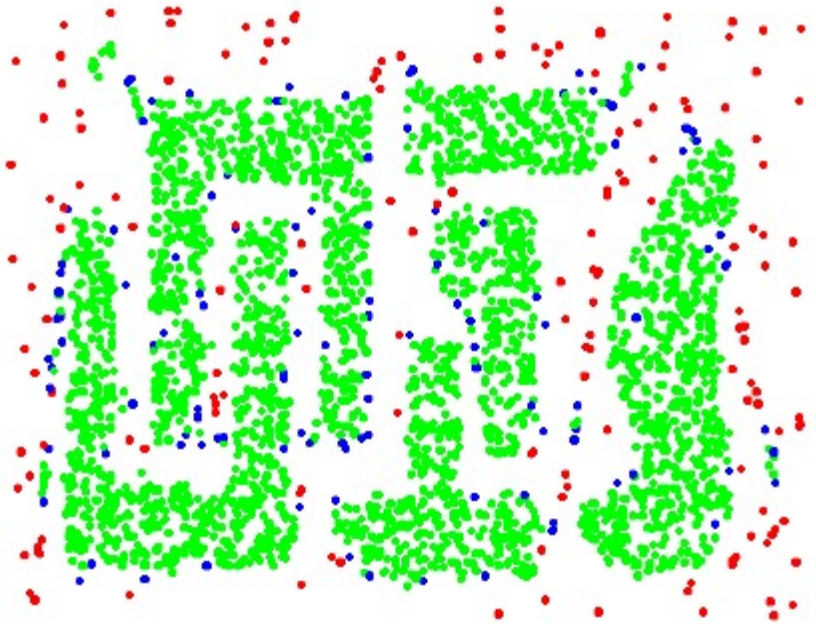


```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

Another Example



Original Points

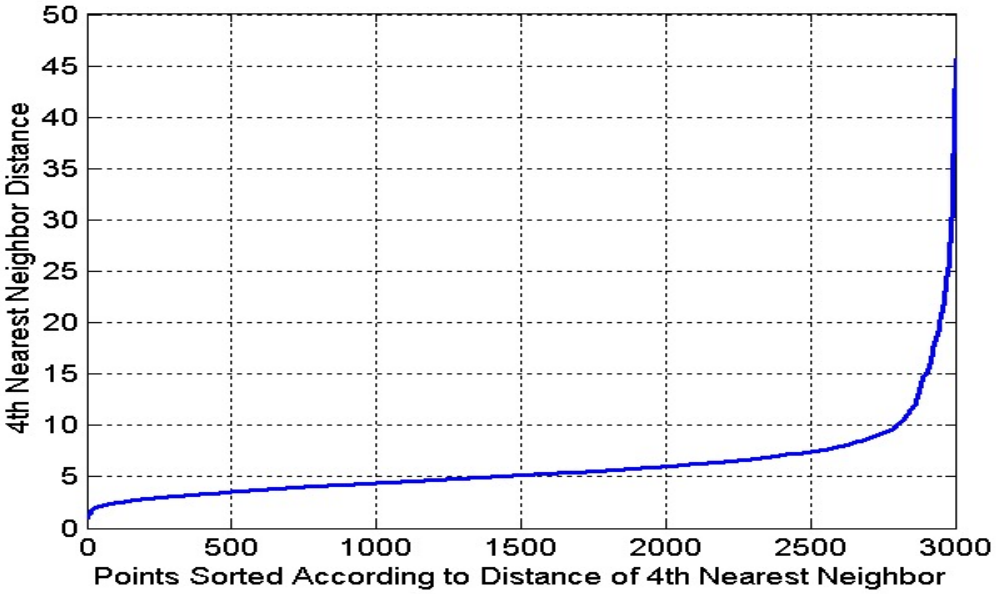


Point types: core, border and outliers

$\varepsilon = 10$, MinPts = 4

DBSCAN: Determining ε and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance (BUT may NOT always be true !!)
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor



DBSCAN: Determining ε and MinPts

- Distance from a point to its k^{th} nearest neighbor=>k-dist
- For points that belong to some clusters, the value of k-dist will be small if k is not larger than cluster size
- For points that are not in a cluster such as noise points, the k-dist will be relatively large
- Compute k-dist for all points for some k
- Sort them in increasing order and plot sorted values
- A sharp change at the value of k-dist that corresponds to suitable value of ε and the value of k as MinPts

DBSCAN: Determining ε and MinPts

- A sharp change at the value of k-dist that corresponds to suitable value of ε and the value of k as MinPts
  - Points for which k-dist is less than ε will be labeled as core points while other points will be labeled as noise or border points.
- If k is too large=> small clusters (of size less than k) are likely to be labeled as noise
- If k is too small=> Even a small number of closely spaced that are noise or outliers will be incorrectly labeled as clusters

Sensitivity of DBSCAN w.r.t. the Choice of ε

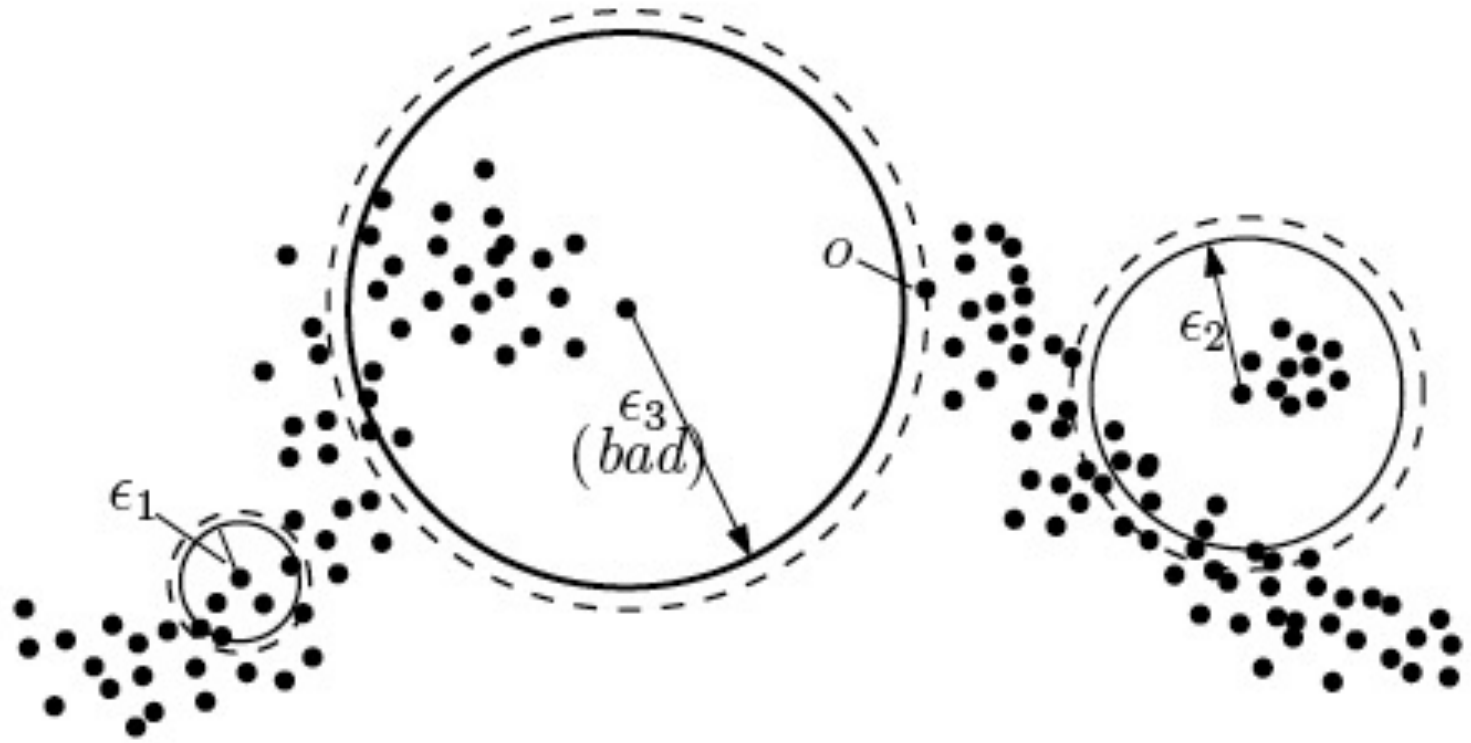


Figure 6: Good and bad choices of ϵ

- Choice of Eps1 will give 3 clusters and Eps2 will give 2 clusters ; these choices of Eps are robust w.r.t. minor perturbation ;

- Eps3 is a bad choice because a slight perturbation of eps3 will change the result from 2 clusters to 1 cluster ; Eps3 is TOO CLOSE to the distance between 2 clusters !!

DBSCAN results Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

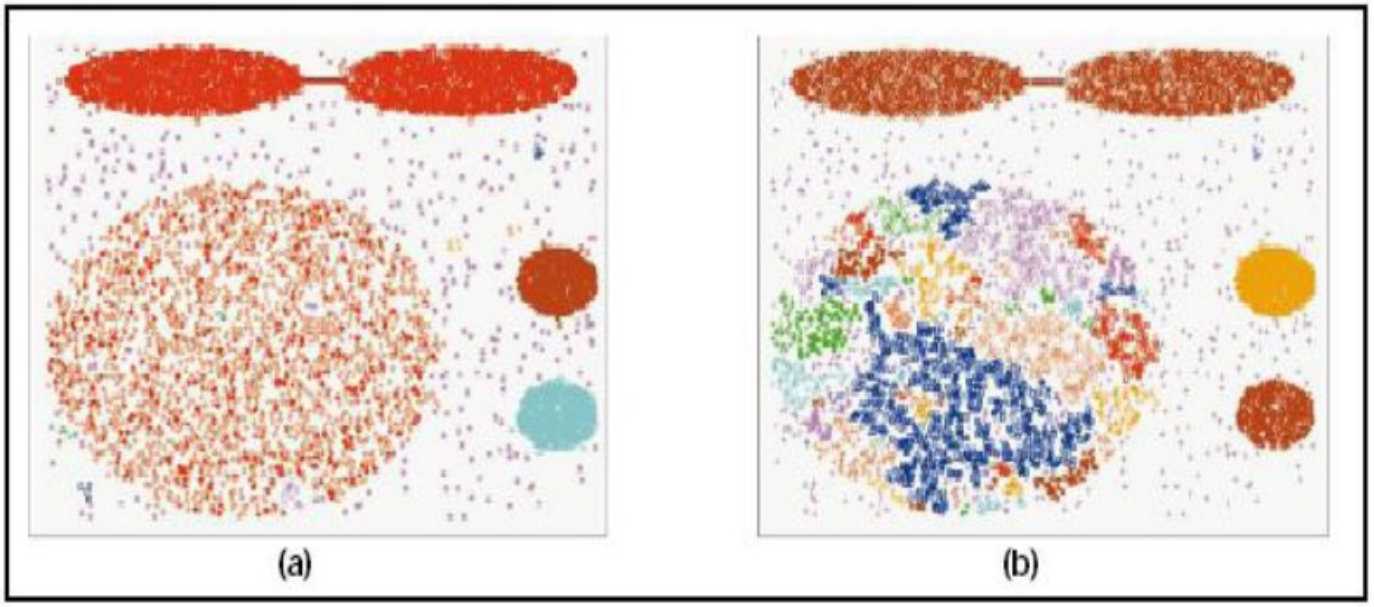
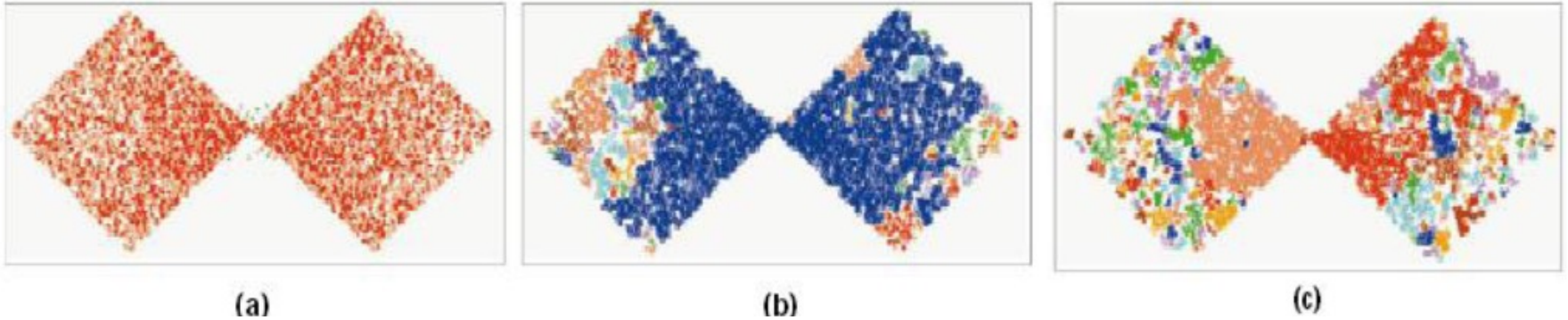
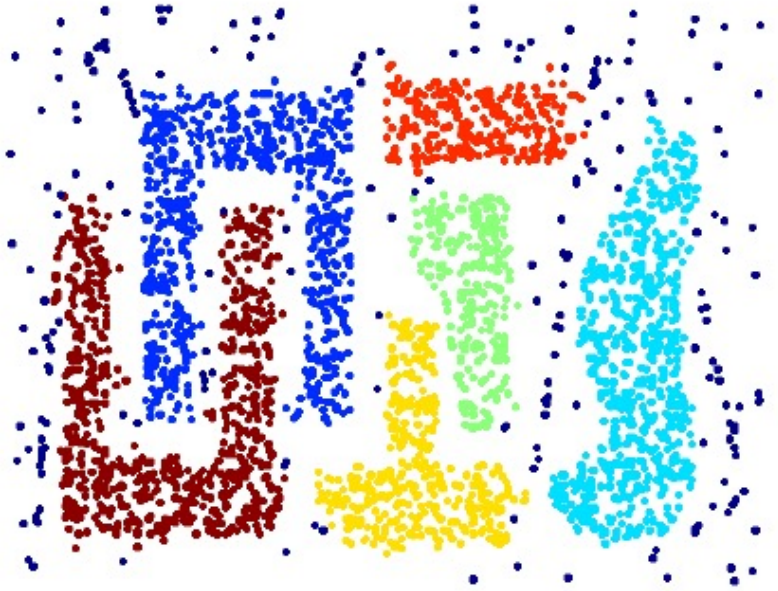


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



When DBSCAN works well

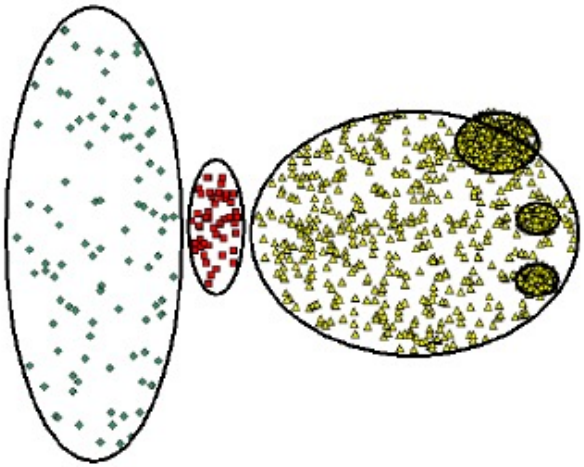
Original Points



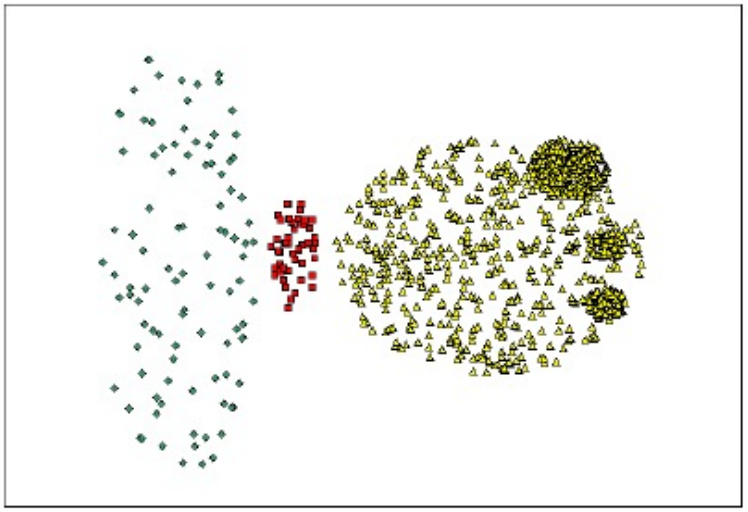
Clusters

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

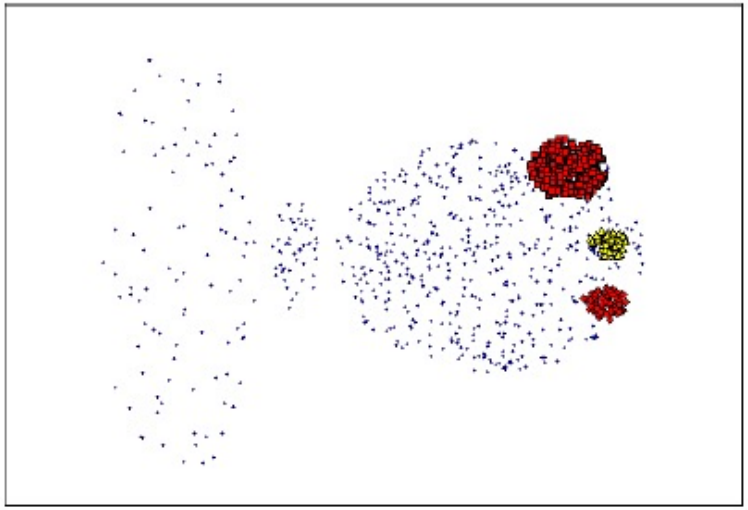
When DBSCAN does NOT work well



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- Cannot handle varying densities
- sensitive to parameters—hard to determine the correct set of parameters

Summary for DBSCAN

- DBSCAN depends on 2 critical parameters:
  - ε and Minpts
- The notion of ε-neighborhood w.r.t. Minpts threshold
- Definition of Core vs. Border vs. Noise (Outliner) points
- Density-Reachability vs. Density-Connectivity
- Defining a Cluster based on Density-Connectivity and MAXIMAL Density-Reachability
- Can be tricky to set the “correct” value of k (size of a cluster)
- Inability to handle highly variable density within the

Summary

- **Clustering:** Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***
- **Algorithms:**
  - Agglomerative **hierarchical clustering**:
    - Centroid and clustroid
  - ***k*-means:**
    - Initialization, picking *k*
  - **EM for GMM**
  - **Density-based Clustering with DBSCAN for Clusters of arbitrary shapes**

Backup Slides

Backup Slides of the Derivation of MLE estimator for GMM

Maximum likelihood estimation of Gaussian

- Given data points x_n, n=1,…,N
- Find a single Gaussian distribution that maximizes data log-likelihood

$$L(\theta)=\sum_{n=1}^{N}\log p(x_n)=\sum_{n=1}^{N}\log N(x_n;m,C)=\sum_{n=1}^{N}\left[-\frac{d}{2}\log(2\pi)-\frac{1}{2}\log|C|-\frac{1}{2}(x_n-m)^T C^{-1}(x_n-m)\right]$$

- Set derivative of data log-likelihood w.r.t. parameters to zero

$$\frac{\partial}{\partial m}L(\theta)=C^{-1}\sum_{n=1}^{N}\left[x_n-m\right]=0$$

$$m=\frac{1}{N}\sum_{n=1}^{N}x_n$$

$$\frac{\partial}{\partial C^{-1}}L(\theta)=\sum_{n=1}^{N}\left[\frac{1}{2}C-\frac{1}{2}(x_n-m)(x_n-m)^T\right]=0$$

$$C=\frac{1}{N}\sum_{n=1}^{N}(x_n-m)(x_n-m)^T$$

- Parameters set as data covariance and mean

Maximum likelihood estimation of Gaussian

$$P(x_n|\theta_k) = P(x_n|\mu_k, \Sigma_k) = \mathcal{N}(x_n|\mu_k, \Sigma_k) = (2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$$

The derivative of a gaussian with respect to its mean

$$\frac{d}{d\mu_k}\{(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\} =$$

$$(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\frac{d}{d\mu_k}\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\} =$$

$$(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\Sigma_k^{-1}(x-\mu_k) =$$

$$\mathcal{N}(x|\mu_k, \Sigma_k)\Sigma_k^{-1}(x-\mu_k)$$

Maximum likelihood estimation of GMM

Maximize for μ_k

$$\frac{d}{d\mu_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{ln(\pi_k) + ln(P(x_n|\mu_k, \Sigma_k))\} = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k, \Sigma_k)}\frac{d}{d\mu_k}P(x_n|\mu_k, \Sigma_k) = 0$$

Use the derivative that we previously found

Maximum likelihood estimation of GMM

Maximize for μ_k

$$\frac{d}{d\mu_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{ln(\pi_k) + ln(P(x_n|\mu_k, \Sigma_k))\} = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k, \Sigma_k)}\frac{d}{d\mu_k}P(x_n|\mu_k, \Sigma_k) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k, \Sigma_k)}P(x_n|\mu_k, \Sigma_k)\Sigma_k^{-1}(x_n - \mu_k) = 0$$

$$\sum_{n=1}^{N} z_{nk}(x_n - \mu_k) = 0$$

$$\mu_k\sum_{n=1}^{N} z_{nk} = \sum_{n=1}^{N} z_{nk}x_n$$

$$\mu_k = \frac{\sum_{n=1}^{N} z_{nk}x_n}{N_k}$$

Maximum likelihood estimation of Gaussian

The derivative of a gaussian with respect to its covariance

$$\frac{d}{d\Sigma_k}\{(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\} =$$

$$(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\frac{d}{d\Sigma_k}\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\}+$$

$$-\frac{1}{2}(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{3}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\frac{d}{d\Sigma_k}|\Sigma_k| =$$

Product rule

Maximum likelihood estimation of Gaussian

The derivative of a gaussian with respect to its covariance

$$\frac{d}{d\Sigma_k}\{(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\} =$$

$$(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{1}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\frac{d}{d\Sigma_k}\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\}+$$

$$-\frac{1}{2}(2\pi)^{-\frac{D}{2}}|\Sigma_k|^{-\frac{3}{2}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}\frac{d}{d\Sigma_k}|\Sigma_k| =$$

$$\mathcal{N}(x|\mu_k,\Sigma_k)\frac{d}{d\Sigma_k}\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\}$$

$$-\frac{1}{2}|\Sigma_k|^{-1}\mathcal{N}(x|\mu_k,\Sigma_k)\frac{d}{d\Sigma_k}|\Sigma_k| =$$

Maximum likelihood estimation of Gaussian

$$\mathcal{N}(x|\mu_k, \Sigma_k)(-\frac{1}{2}(x-\mu_k)^T(x-\mu_k))\frac{d}{d\Sigma_k}\Sigma_k^{-1}$$
$$-\frac{1}{2}|\Sigma_k|^{-1}\mathcal{N}(x|\mu_k, \Sigma_k)|\Sigma_k|\Sigma_k^{-1} =$$

Maximum likelihood estimation of Gaussian

$$\mathcal{N}(x|\mu_k,\Sigma_k)(-\frac{1}{2}(x-\mu_k)^T(x-\mu_k))\frac{d}{d\Sigma_k}\Sigma_k^{-1}$$

$$-\frac{1}{2}|\Sigma_k|^{-1}\mathcal{N}(x|\mu_k,\Sigma_k)|\Sigma_k|\Sigma_k^{-1}=$$

Derivative of inverse

$$\mathcal{N}(x|\mu_k,\Sigma_k)(-\frac{1}{2}(x-\mu_k)^T(x-\mu_k))(-\Sigma_k^{-1}\Sigma_k^{-1})$$

$$-\frac{1}{2}|\Sigma_k|^{-1}\mathcal{N}(x|\mu_k,\Sigma_k)|\Sigma_k|\Sigma_k^{-1}=$$

Maximum likelihood estimation of Gaussian

$$\mathcal{N}(x|\mu_k,\Sigma_k)(-\frac{1}{2}(x-\mu_k)^T(x-\mu_k))\frac{d}{d\Sigma_k}\Sigma_k^{-1}$$

$$-\frac{1}{2}|\Sigma_k|^{-1}\mathcal{N}(x|\mu_k,\Sigma_k)|\Sigma_k|\Sigma_k^{-1}=$$

$$\mathcal{N}(x|\mu_k,\Sigma_k)(-\frac{1}{2}(x-\mu_k)^T(x-\mu_k))(-\Sigma_k^{-1}\Sigma_k^{-1})$$

$$-\frac{1}{2}|\Sigma_k|^{-1}\mathcal{N}(x|\mu_k,\Sigma_k)|\Sigma_k|\Sigma_k^{-1}=$$

$$\mathcal{N}(x|\mu_k,\Sigma_k)\frac{1}{2}(x-\mu_k)^T(x-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1}$$

$$-\frac{1}{2}\mathcal{N}(x|\mu_k,\Sigma_k)\Sigma_k^{-1}=$$

$$\frac{1}{2}\mathcal{N}(x|\mu_k,\Sigma_k)\{(x-\mu_k)^T(x-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1}-\Sigma_k^{-1}\}$$

Maximum likelihood estimation of GMM

Set derivative with respect to Σ_k equal to 0:

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{ln(\pi_k) + ln(P(x_n|\mu_k, \Sigma_k))\} = 0$$

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} ln(P(x_n|\mu_k, \Sigma_k)) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k, \Sigma_k)}\frac{d}{d\Sigma_k}P(x_n|\mu_k, \Sigma_k) = 0$$

Use derivative of gaussian

Maximum likelihood estimation of GMM

Set derivative with respect to Σ_k equal to 0:

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{ln(\pi_k) + ln(P(x_n|\mu_k,\Sigma_k))\} = 0$$

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} ln(P(x_n|\mu_k,\Sigma_k)) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k,\Sigma_k)}\frac{d}{d\Sigma_k}P(x_n|\mu_k,\Sigma_k) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k,\Sigma_k)}\frac{1}{2}P(x_n|\mu_k,\Sigma_k)\{(x_n-\mu_k)^T(x_n-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1} - \Sigma_k^{-1}\} = 0$$

Drop distributions

Maximum likelihood estimation of GMM

Set derivative with respect to Σ_k equal to 0:

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{ln(\pi_k) + ln(P(x_n|\mu_k,\Sigma_k))\} = 0$$

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} ln(P(x_n|\mu_k,\Sigma_k)) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k,\Sigma_k)}\frac{d}{d\Sigma_k}P(x_n|\mu_k,\Sigma_k) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k,\Sigma_k)}\frac{1}{2}P(x_n|\mu_k,\Sigma_k)\{(x_n-\mu_k)^T(x_n-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1} - \Sigma_k^{-1}\} = 0$$

$$\sum_{n=1}^{N} z_{nk}\{(x_n-\mu_k)^T(x_n-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1} - \Sigma_k^{-1}\} = 0$$

Multiply with Σ_k Σ_k

Maximum likelihood estimation of GMM

Set derivative with respect to Σ_k equal to 0:

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{ln(\pi_k) + ln(P(x_n|\mu_k,\Sigma_k))\} = 0$$

$$\frac{d}{d\Sigma_k}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} ln(P(x_n|\mu_k,\Sigma_k)) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k,\Sigma_k)}\frac{d}{d\Sigma_k}P(x_n|\mu_k,\Sigma_k) = 0$$

$$\sum_{n=1}^{N} z_{nk}\frac{1}{P(x_n|\mu_k,\Sigma_k)}\frac{1}{2}P(x_n|\mu_k,\Sigma_k)\{(x_n-\mu_k)^T(x_n-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1} - \Sigma_k^{-1}\} = 0$$

$$\sum_{n=1}^{N} z_{nk}\{(x_n-\mu_k)^T(x_n-\mu_k)\Sigma_k^{-1}\Sigma_k^{-1} - \Sigma_k^{-1}\} = 0$$

$$\sum_{n=1}^{N} z_{nk}\{(x_n-\mu_k)^T(x_n-\mu_k) - \Sigma_k\} = 0$$

$$\sum_{n=1}^{N} z_{nk}\Sigma_k = \sum_{n=1}^{N} z_{nk}(x_n-\mu_k)^T(x_n-\mu_k)$$

$$N_k\Sigma_k = \sum_{n=1}^{N} z_{nk}(x_n-\mu_k)^T(x_n-\mu_k)$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} z_{nk}(x_n-\mu_k)^T(x_n-\mu_k)}{N_k}$$