Unsupervised Learning

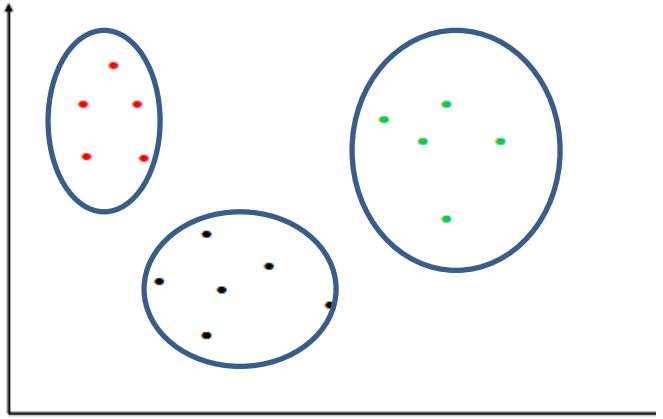
Given data, discover self-similar groups within the data (clustering)
얘는 supervised랑 다르게 data에 대한 information이 없어

CS534

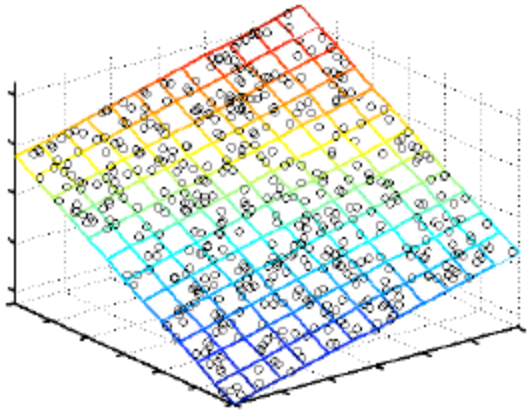
What can we learn from unlabeled data?

- Finding Group of clusters

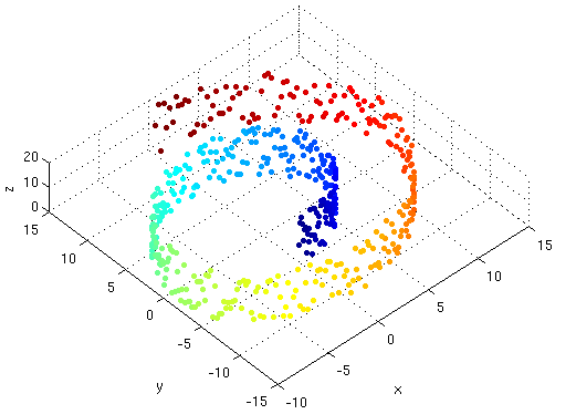
in the data



- Finding low dimensional representation – dimension reduction

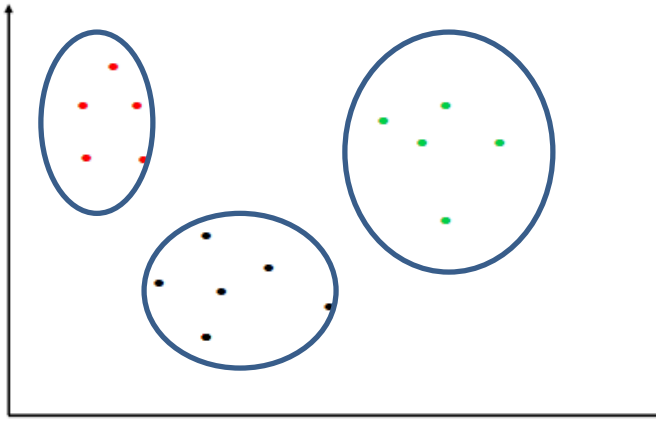


PCA



Nonlinear embedding

Clustering



- Clustering: the process of grouping a set of objects into classes of similar objects
  - high within-class similarity
  - low cross-class similarity
- It is the most common form of unsupervised learning

Example Applications

- Find genes that are similar in their functions
- Group documents based on topics
- Categorize customers based on their purchase history
- Group images based on their contents
- ……

Critical Issues in Clustering

- What makes objects similar?
  - Definition of "similarity/distance“
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
  - Avoid “trivial” clusters - too large or small
- Clustering Algorithms
  - Flat vs hierarchical
  - Hard vs soft

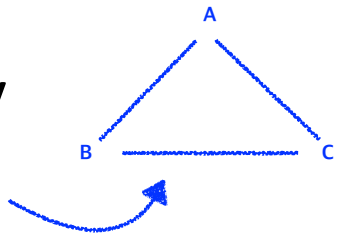
What is similarity

Hard to define but
We know it when we see it

- The real meaning of similarity is a philosophical question. We will take a more pragmatic (실용/실제적인) approach
  - Depends on representation and algorithm. For many rep./alg., easier to think in terms of a distance (rather than similarity) between vectors

What properties should a distance measure have?

- D must be Symmetric
  - $D(A, B) = D(B, A)$
  - Otherwise, we can say A looks like B but B does not look like A
- Positivity, and self-similarity
  - Distance is non-negative
  - $D(A, B) \geq 0$, and $D(A, B) = 0$ iff $A = B$
  - Otherwise there will be different objects that we cannot tell apart
- Must satisfy triangle inequality
  - $D(A, B) + D(B, C) \geq D(A, C)$
  - Otherwise one can say "A is like B, B is like C, but A is not like C at all"



얘네는 not for binary features

Distance Measures: Minkowski Metric

- Suppose two object x and y both have d features
  - $x = (x_1, \cdots, x_d), y = (y_1, \cdots, y_d)$
- The Minkowski metric of order r is defined by

summing over all features

$$d(x, y) = \sqrt[r]{\sum_i |x_i - y_i|^r}$$

- Common Minkowski metrics:

Distance 계산하는 방법 : Minkowski Metric
근데 r이 1 이냐 2 냐 +inf 냐에 따라 달라짐

  - Euclidean(r=2): $d(x, y) = \sqrt[2]{\sum_i (x_i - y_i)^2}$, also called L_2 distance
  - Manhattan distance(r=1) : $d(x, y) = \sum_i |x_i - y_i|$, also called L_1 distance
  - "Sup" distance(r = $+\infty$): $d(x, y) = \max_i |x_i - y_i|$, also called L_∞ distance

Other Distances

non-binary feature는 hamming distance쓰고
binary feature는 맨해튼 디스턴스써락

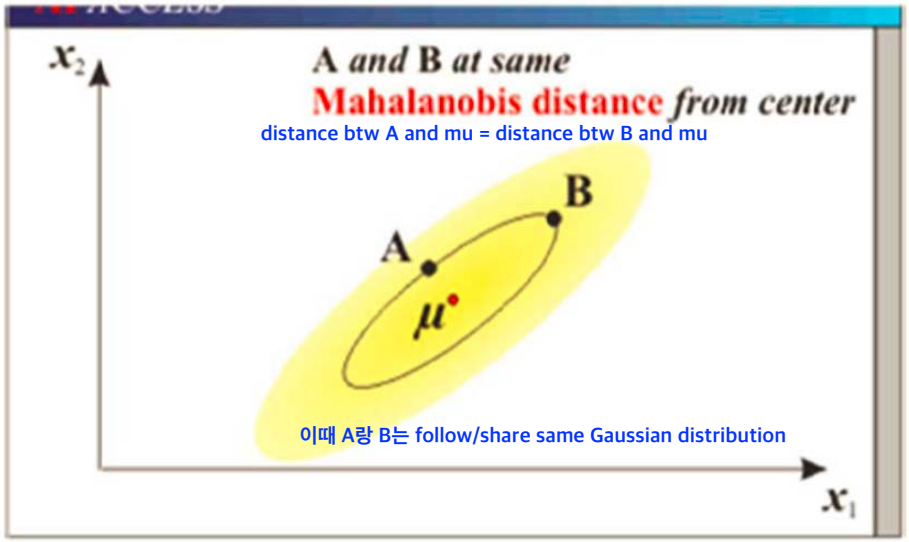
- Hamming distance (Manhattan distance on binary features)
  - # of features that differ
  - e.g.: distance of two sites based on their species composition

	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9
Site A:	1	0	1	1	0	0	1	0	1
Site B:	0	0	1	0	1	1	1	0	1

D(A, B) = 4

- Mahalanobis distance (assuming $\mathbf{x}, \mathbf{y}$ follows a Gaussian distribution with covariance matrix Σ

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$



Similarities

- Cosine similarities – commonly used to measure document similarity

$$cos(\mathrm{x}, \mathrm{x}') = \frac{<\mathrm{x} \cdot \mathrm{x}'>}{|\mathrm{x}| \cdot |\mathrm{x}'|}$$

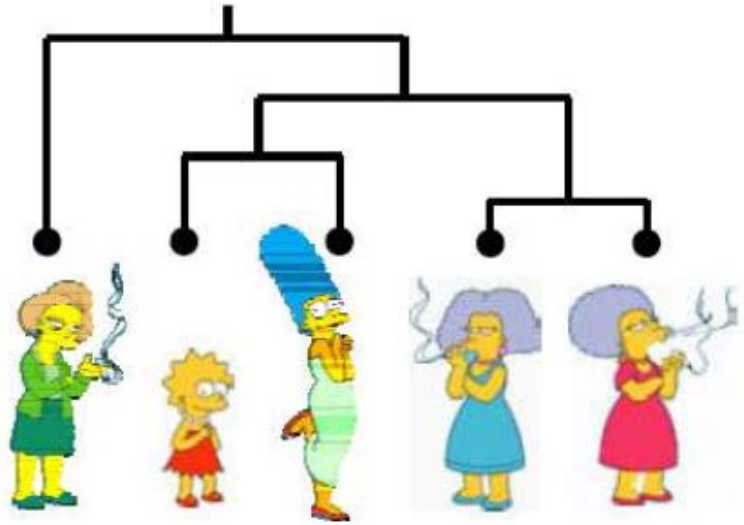
똑같으면 1
완전다르면 0

- Kernels – e.g., RBF (Gaussian) Kernel

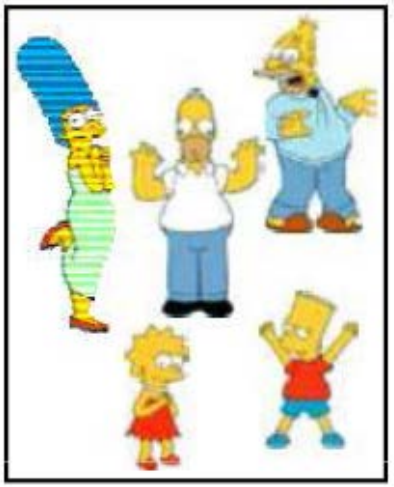
$$S(X, X') = \exp \frac{-|X - X'|^2}{2\sigma^2}$$

Clustering algorithms

- Hierarchical algorithms (not covered)
  - Bottom up – agglomerative
  - Top down – divisive

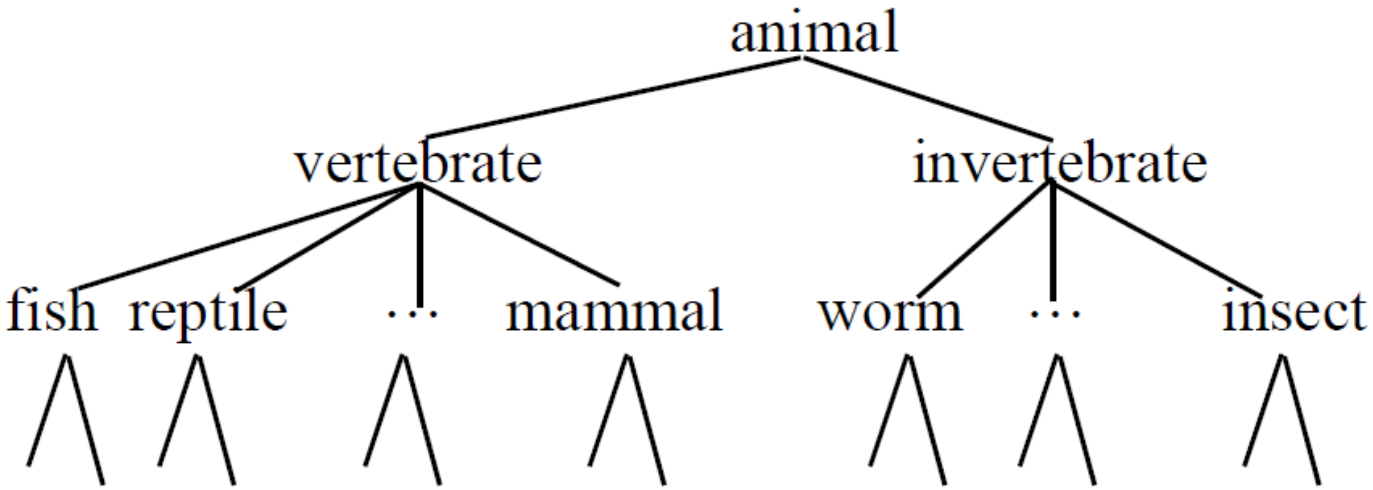


- Flat clustering algorithms
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering (not covered)



Hierarchical Clustering

- Given a set of objects, build a tree-based taxonomy



- Hierarchies are convenient way for organizing information

Not covered

Hierarchical Agglomerative Clustering (HAC)

- Starts with each object in a separate cluster
- Repeatedly joins the closest pair of clusters
- until there is only one cluster

The history of merging forms a tree of hierarchy

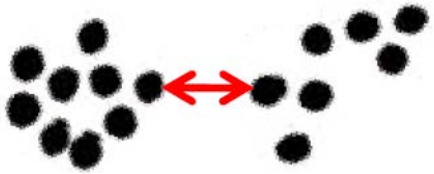
Question: how to measure the **"closeness"** of two clusters?

Not covered

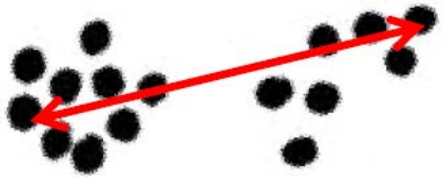
Closest Pair of Clusters

The distance between two clusters is defined as the distance between:

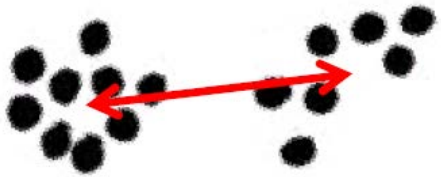
- Single-link
  - The nearest pair of points



- Complete-link
  - The furthest pair of points



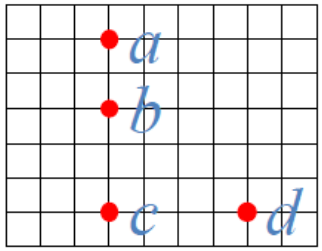
- Centroid
  - The center of gravity

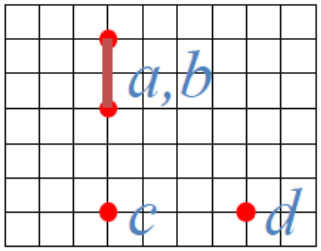


- Average-link
  - Average of all cross-cluster pairs

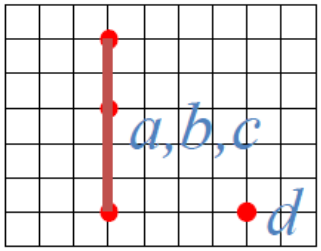
Not covered

Single Link Method

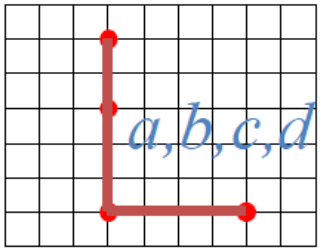




(1)



(2)



(3)

	b	c	d
a	2	5	6
b		3	5
c			4

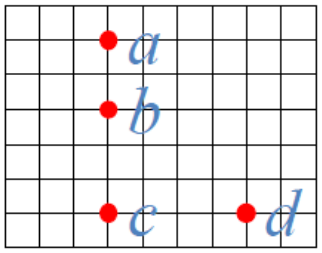
	b	c	d
a	2	5	6
b		3	5
c			4

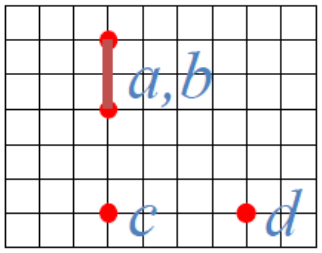
	c	d
a, b	3	5
c		4

	d
a, b, c	4

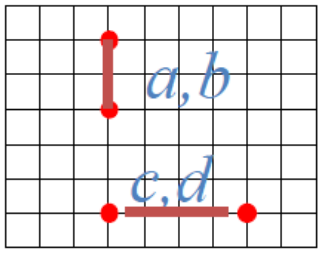
Not covered

Complete Link Method

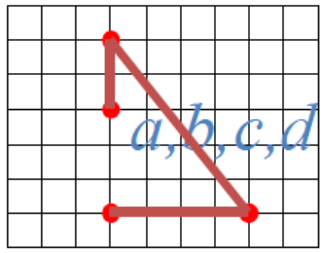




(1)



(2)



(3)

	b	c	d
a	2	5	6
b		3	5
c			4

	b	c	d
a	2	5	6
b		3	5
c			4

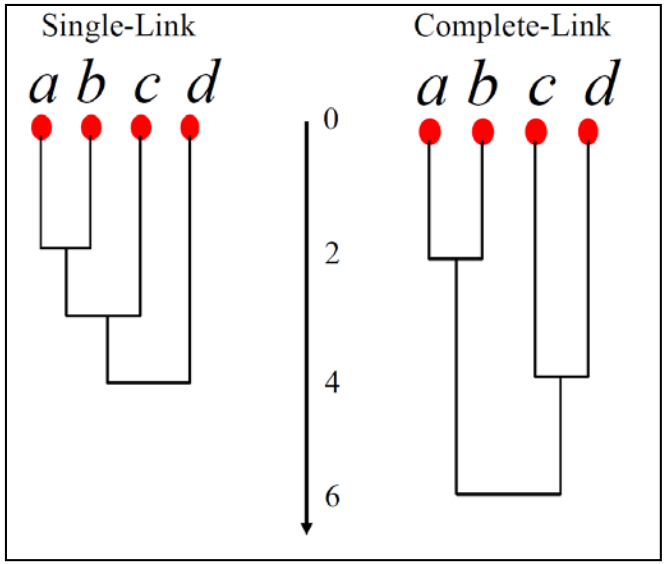
	c	d
a,b	5	6
c		4

	c,d
a,b	6

Not covered

Visualization: Dendrogram

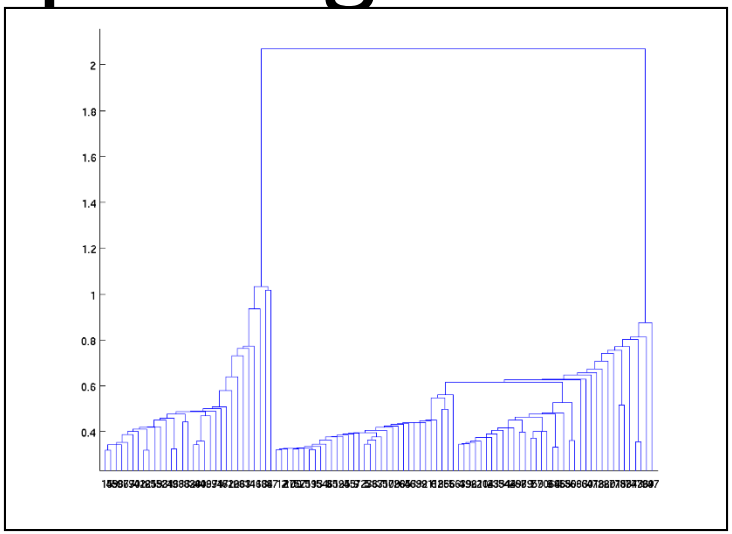
- Height of the joint = the distance between the two merge clusters
- The merge distance monotonically increases as we merge more and more for
  - Single, complete and average linkage methods
  - Not for the centroid method



This example is shown upside down.

Not covered

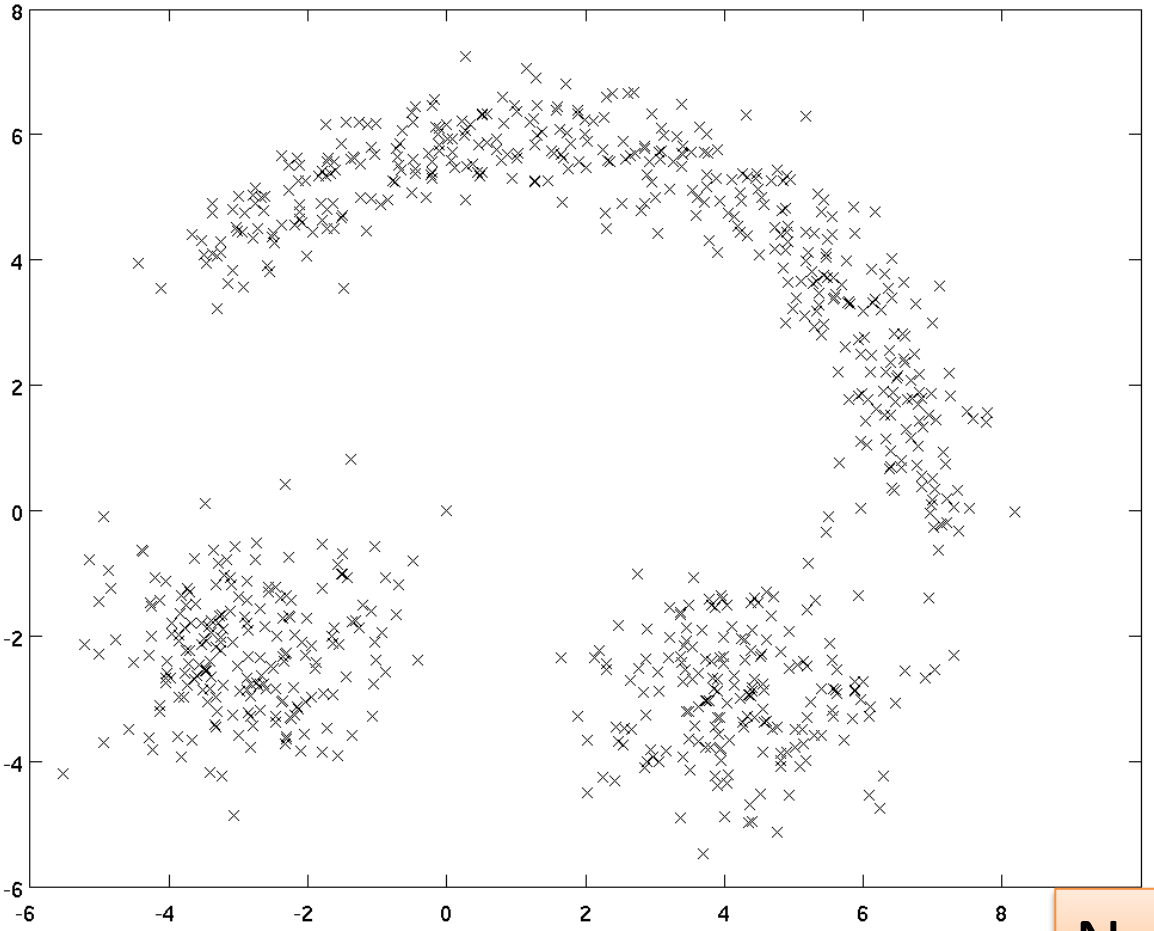
Interpreting Dendrogram



- Dendragram can be used to visually identify
  - the number of clusters in data
    - A horizontal cut creates a partition
    - Moving the cut from root down creates more clusters
    - Large gaps indicate good cutting points
  - well-formed clusters
    - Some clusters are better formed than the other
    - This can be easy to see on a dendrogram

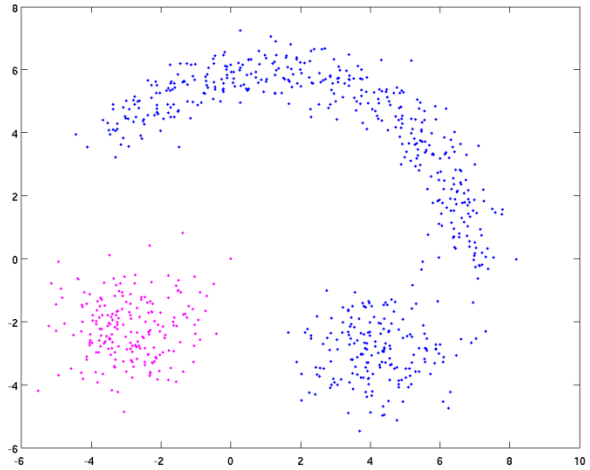
Not covered

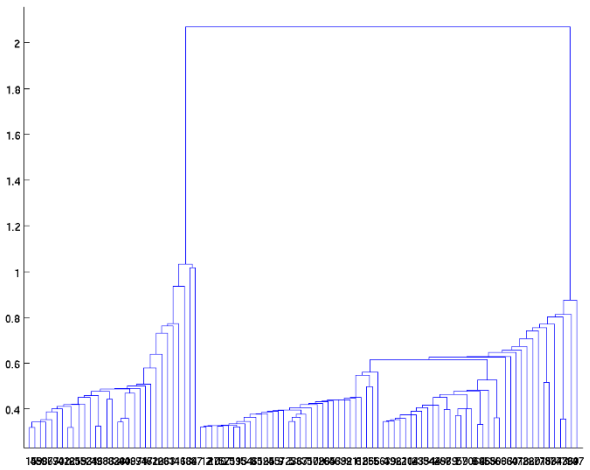
Example

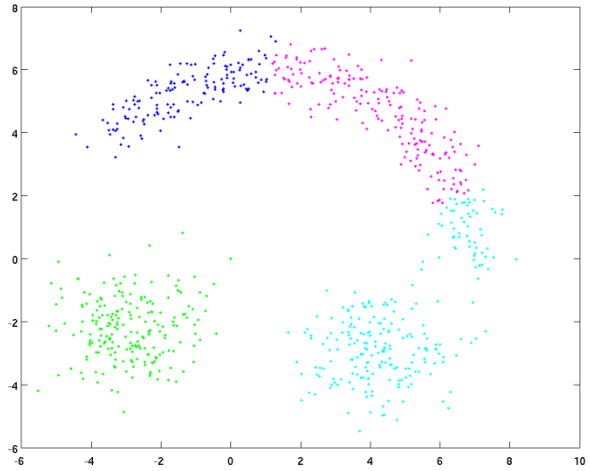


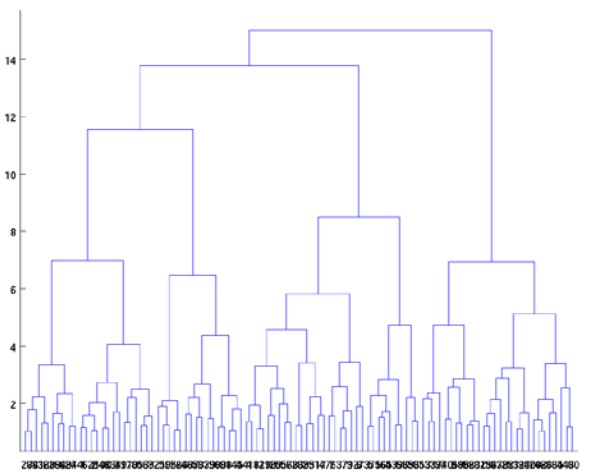
Not covered

Single Link vs. Complete Link





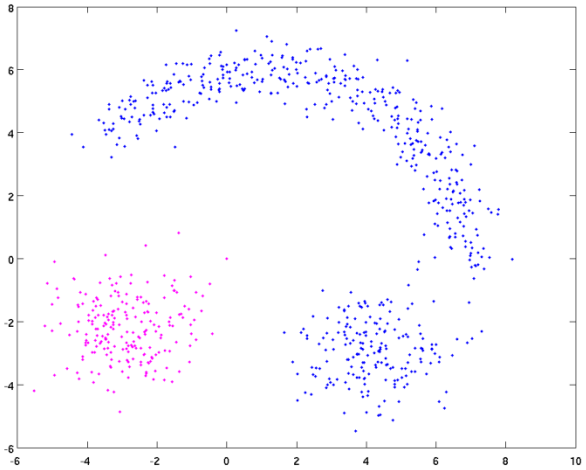


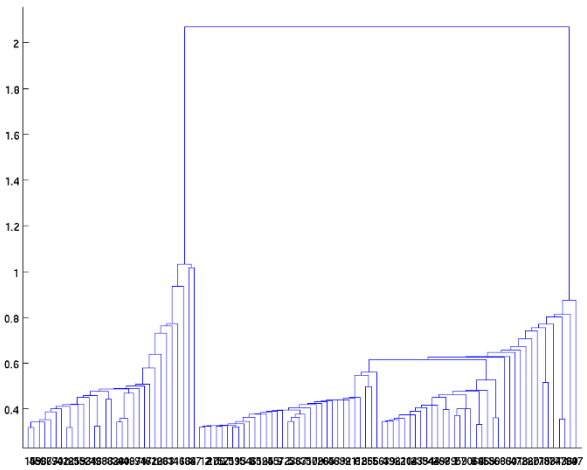


Not covered

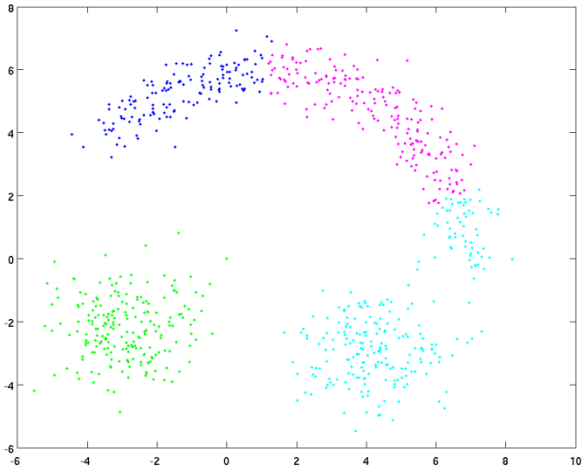
Which is single link? Which is complete link?

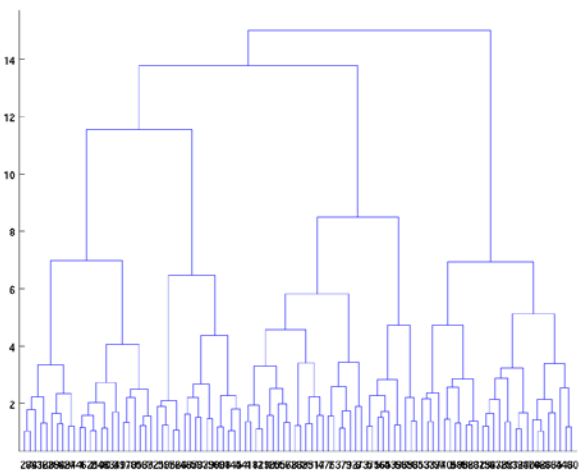
Single Link vs. Complete Link





Single





Complete

Not covered

- Single-link creates straggly clusters due to chaining effect

Flat Clustering

- Given D, a data set of n points, we want to find k clusters
- For each cluster we would like to represent all points of that cluster with one representative, and the representation error should be minimized 각 cluster에 대표하는 representative가 있어야겠징 그리고 개랑 비슷한애들은 걔의 cluster로
- This leads to the following objective function

sum up over all clusters 얘가 error고

$$\min_{\mu, C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

representative cluster

Where $C = \{C_1, C_2, \ldots, C_k\}$ is a partition of D such that

$$C_i \cap C_j = \emptyset, \forall i \neq j \text{ and } D = \cup_i C_i$$

D = all C들의 union

and μ_i is the representative of C_i

- A combinatorial optimization problem
  - Discrete solution space
  - Exhaustive search for an optimal solution is not feasible cuz too expensive

An Iterative Solution

- ***Initialization:*** Start with a random partition of the data
- ***Iterative step***: the cluster assignments and cluster centers are updated to improve the objective
- ***Stopping criterion***: if no improvement can be achieved. til converge

K-Means

Algorithm

Input – Desired number of clusters, *k*

Initialize – the *k* cluster centers (randomly if necessary)

Iterate –

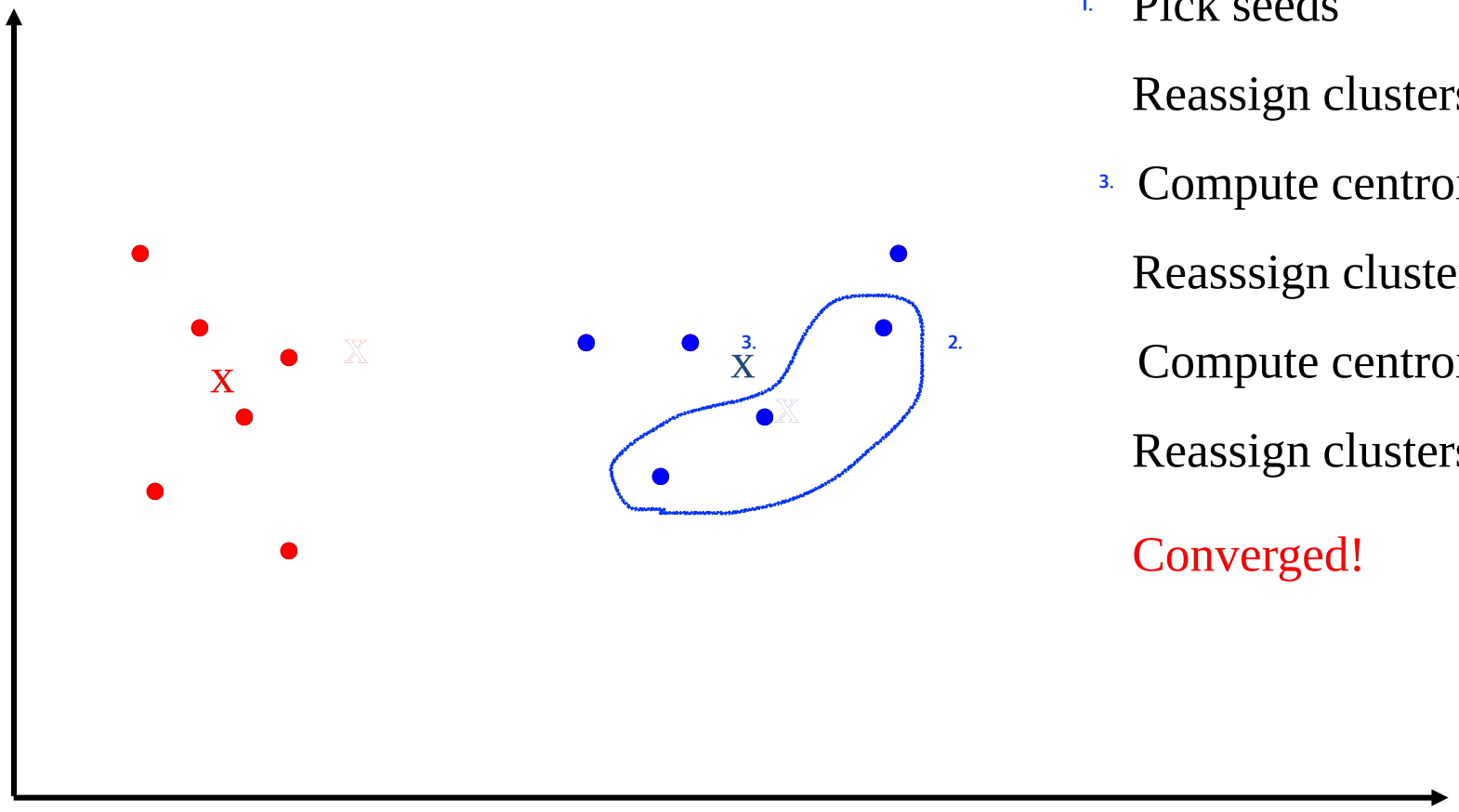
1. Assigning each of the ~~N~~ n data points to its nearest cluster centers
2. Re-estimate the cluster center by assuming that the current assignment is correct

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Termination –

If none of the data points changed membership =iterating until converge in the last iteration, exit. Otherwise, go to 1

Demo: K-Means Example (K=2)



1. Pick seeds

   Reassign clusters

3. Compute centroids

   Reasssign clusters

   Compute centroids

   Reassign clusters

   Converged!

Does KMeans Converge?

Objective

$$\min_{\mu,C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ, optimize C
   - Assign each point to its closet center

Step 1 of kmeans

2. Fix C, optimize μ:

$$\min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

   - Take partial derivative of μ_i and set to zero, we have

$$2 \sum_{x \in C_i} (x - \mu_i) = 0 \Rightarrow \mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

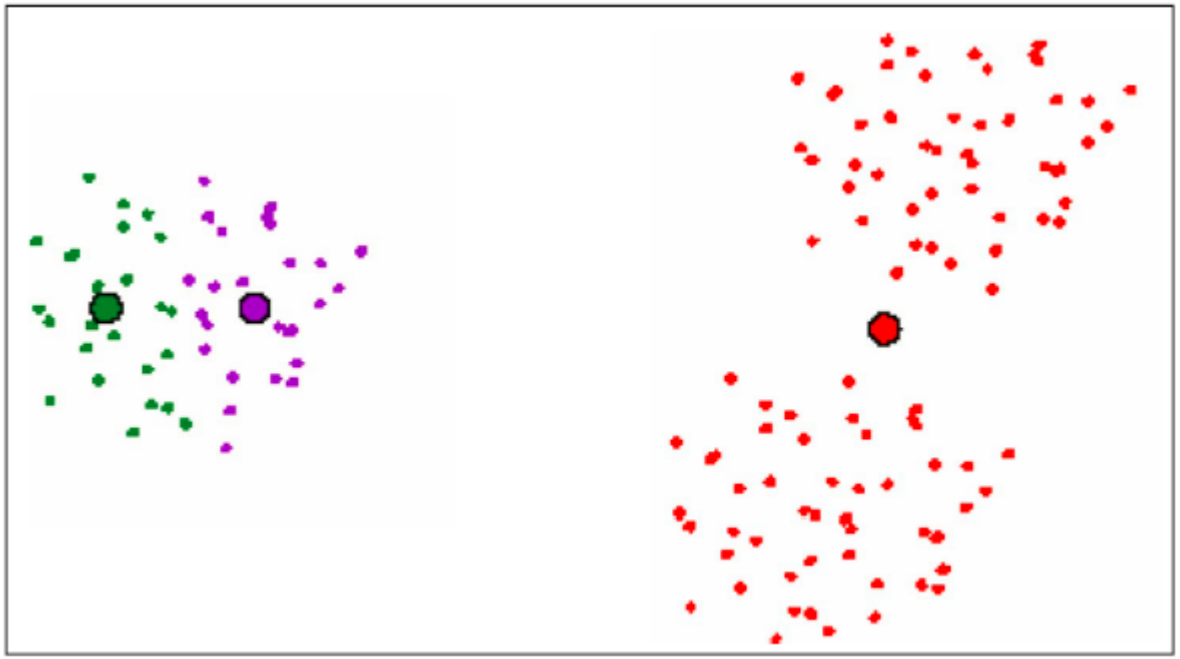
derivative of mu_i,해서 set to 0 하면,
We get mu_i = mean of i cluster

Step 2 of kmeans

Each iteration is guaranteed to decrease the objective and there are finite possible C's – thus guaranteed to converge --- but only to local minimum

Impact of Initial Seeds

- Highly sensitive to the initial seeds initial seed가 매우 중요 in K means
  initial seed를 그지같이 선택하면 원하는 결과를 가질 수 없음 at the end



- Multiple random trials: choose the one with best sum of squared loss (important!)
- Heuristics for choosing better centers
  - choose initial centers to be far apart – furthest first traversal (kmeans ++)

More Comments

- K-Means is exhaustive:
  - Cluster every data point, no notion of outlier
  - Outliers cause problems
    - Become singular clusters 아웃라이어 하나가 하나의 클러스터가 된다
    - Bias the centroid estimation while we want unbias
- K-medoids methods is more robust to outliers but more expensive than K means
  - Cluster medoid: must be one of the data points, one that has the minimum sum squared distance to all data points in the cluster
  - More expensive to compute 왜냠 you should compute all pairwise distance
    - For each pt: sum squared dist with all other pts in cluster $O(|C|^2)$
- Need to specify k: difficult in practice

Hard vs. Soft Clustering

- Hard clustering:
  - Data point is deterministically assigned to one and only one cluster
  - But in reality clusters may overlap
- Soft-clustering:
  - Data points are assigned to clusters with certain probabilities
- Often referred to as model-based clustering for the use of probabilistic model for clusters

Each data point is assigned to

Side track: Gaussian Bayes Classifier

After alg. each class has its own specific Gaussian distribution

- We have k classes in our data
- Each class contains data generated from a particular Gaussian distribution $N(\mu_c, \Sigma_c), c = 1, \dots, k$
- The data is generated as follows:
  - first randomly select the class label according to prior $p(y)$
  - draw a random sample from the Gaussian distribution of that particular class
  - Repeat n times to generate n points
- In supervised learning, we observe the resulting (x, y) pairs and we simply need to estimate $p(y = c)$ and $p(x|y = c)$ for $c = 1, \dots, k$

MLE Estimates of Mean and Covariance for Gaussian

Given that $p(x|y=c) \sim N(\mu_c, \Sigma_c)$, and given a set of n_c training examples with $y = c$:

$$\mu_c = \frac{1}{\sum_i I(y_i = c)} \sum_{y_i = c} x_i$$

$$\Sigma_c = \frac{1}{\sum_i I(y_i = c)} \sum_{y_i = c} (x_i - \mu_c)(x_i - \mu_c)^T$$

normalize

Back to Unsupervised Learning

- Now assume we know our data is generated in the same way
- But for unsupervised learning, we don't have the y's, and only observe $x's$
  - We observe a mixture of Gaussians
- How can we learn the correct model from the incomplete data?
- We will still use Maximum likelihood estimation

Gaussian Mixture Model

because we only observe x, the likelihood function will be about x

$$P(\mathbf{x}) = \sum_{i=1}^{k} P(\mathbf{x}, y = i)$$

이때 y는 모르지만, y가 어떻게 generate되는지 아니까

$$= \sum_{i=1}^{k} P(\mathbf{x} \mid y = i) P(y = i)$$

$$= \sum_{i=1}^{k} \alpha_i P(\mathbf{x} | \theta_i)$$

probability of y=i

$\theta_i = \{\mu_i, \Sigma_i\}$

위에서배운거 gaussian

$\alpha_i = P(y = i)$: class priors

Mixing parameter

Goal of learning:

- Given a set of x's, find the values of $\underbrace{\{\alpha_1, \cdots, \alpha_k, \theta_1, \cdots, \theta_k\}}_{\Theta}$ that maximize the likelihood of observing the $x's$

Maximum Marginal Likelihood

iid

$$\arg\max_\theta \prod_j P(\mathbf{x}^j) = \arg\max_\theta \prod_j \sum_{i=1}^{k} P(\mathbf{x}^j, y^j = i)$$

이때 j는 index in training sets

$$= \arg\max_\theta \sum_{j=1}^{n} \log \sum_{i=1}^{k} P(\mathbf{x}^j, y^j = i)$$

파이[P(Xi)] 에서

P(Xi) = 시그마 for all y_i P(Xi,y_i)
= p1f1(xi) + P2f2(xi) + p3f3(xi)

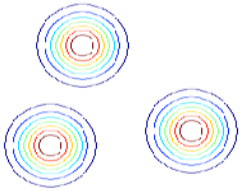
log sum is difficult to optimize !

Gradient ascent is doable but very inefficient

Expectation Maximization (EM)

- Commonly used approach for dealing with hidden (missing) data
  - Here the cluster labels are hidden
- Iterative algorithm that starts with an initial guess of the model parameters
- Iteratively performs two linked steps:
  - **Expectation (E-step)**: given current model parameters λ_t, compute the expectation for the hidden (missing) data
  - **Maximization (M-step)**: re-estimate the parameters λ_{t+1} assuming that the expected values computed in the E-step are the true values
- We will first show how it works for mixture of Gaussian

EM – simple case



- A simple case: spherical Gaussians
  - We have unlabeled data $x^1, \cdots, x^m$
  - We know there are K classes
  - We know $\alpha_1 = P(y = 1), \ldots \alpha_K = P(y = K)$
  - We don't know $\mu_1 \cdots \mu_K$, but know the common variance σ^2

Start with an initial guess for $\mu_1, \ldots, \mu_K$,

1. If we know $\mu_1, \ldots, \mu_K$, we can easily compute probability that a point x^j belongs to class k:

even though each class has its own gaussian distribution,
Alpha is given for exam haha. 그럼 걍 그걸로 각각 클래스 계산

$$P(y = k | x^j) \propto \exp\left(-\frac{1}{2\sigma^2}|x^j - \mu_k|^2\right) p(y = k)$$

Simply evaluate this, then normalize

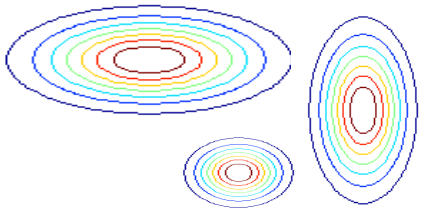
E-step

2. If we know *the* probability that each point belongs to each class, we can estimate the $\mu_1, \ldots, \mu_K$

$$\mu_k = \frac{\sum_{j=1}^{m} P(y=k|x^j) x^j}{\sum_{j=1}^{m} p(y=k|x^j)}$$

M-step

EM – Axis-aligned Gaussian



- We have unlabeled data $x^1, \cdots, x^m$
- We know there are K classes
- We know that the Gaussians are axis aligned

$$\Sigma_i = \begin{pmatrix} \sigma^2_{i,1} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma^2_{i,2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma^2_{i,3} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2_{i,m-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma^2_{i,m} \end{pmatrix}$$

randomly select하는애를 센터로하면 initial guess for mu 는 easya

Start with an initial guess for $\mu_1, \ldots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

알파 하나만 작게하고 다른애들 다 크게하고 이러면 앙대.
왜냐 얘가 posterior를 결정하고 하니까 ㅇㅇ

1. Given parameters, we can easily compute probability that a point x^j belongs to class i:

$$p(y = k|x^j) \propto p(x^j|\mu_k, \Sigma_k)\, p(y = k)$$

Simply evaluate this, then normalize

E-step

2. Given the probability of each point belonging to each class, re-estimate the $\mu_1, \ldots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

$$\mu_k = \frac{\sum_{j=1}^{m} P(y=k \mid x^j)\, x^j}{\sum_{j=1}^{m} P(y=k|x^j)} \qquad \alpha_k = \frac{\sum_{j=1}^{m} P(y=k \mid x^j)}{m}$$

$$\sigma^2_{kl} = \frac{\sum_{j=1}^{m} P(y=k \mid x^j)\left(x^j_l - \mu_{kl}\right)^2}{\sum_{j=1}^{m} P(y=k \mid x^j)}$$

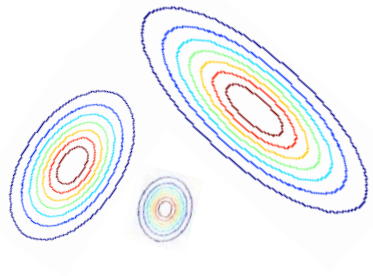
l dimension of mu_k

k라는 class

M-step

EM – General Gaussian

M step만 diff.



Start with an initial guess for $\mu_1, \ldots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

1. If we know the parameters, we can easily compute probability that a point x^j belongs to class k:

$$P(y = k|x^j) \propto p(x^j|\mu_k, \Sigma_k)\, p(y = k)$$

Simply evaluate this, then normalize

E-step

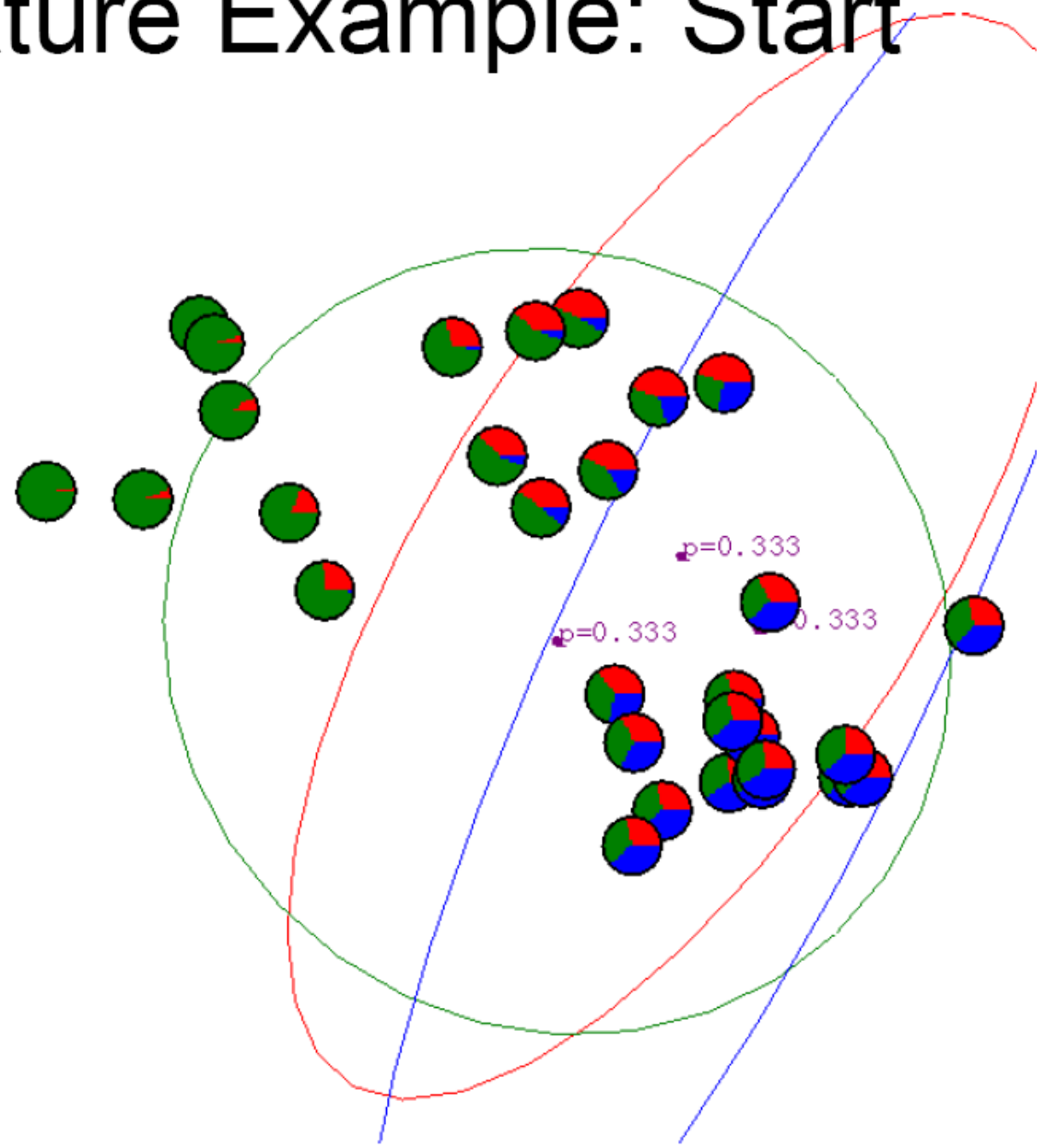
2. If we know *the* probability that each point belongs to each class, we can estimate the $\mu_1, \ldots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

$$\mu_k = \frac{\sum_{j=1}^{m} P(y=k|x^j)x^j}{\sum_{j=1}^{m} P(y=k|x^j)} \qquad \alpha_k = \frac{\sum_{j=1}^{m} P(y = k|x^j)}{m}$$

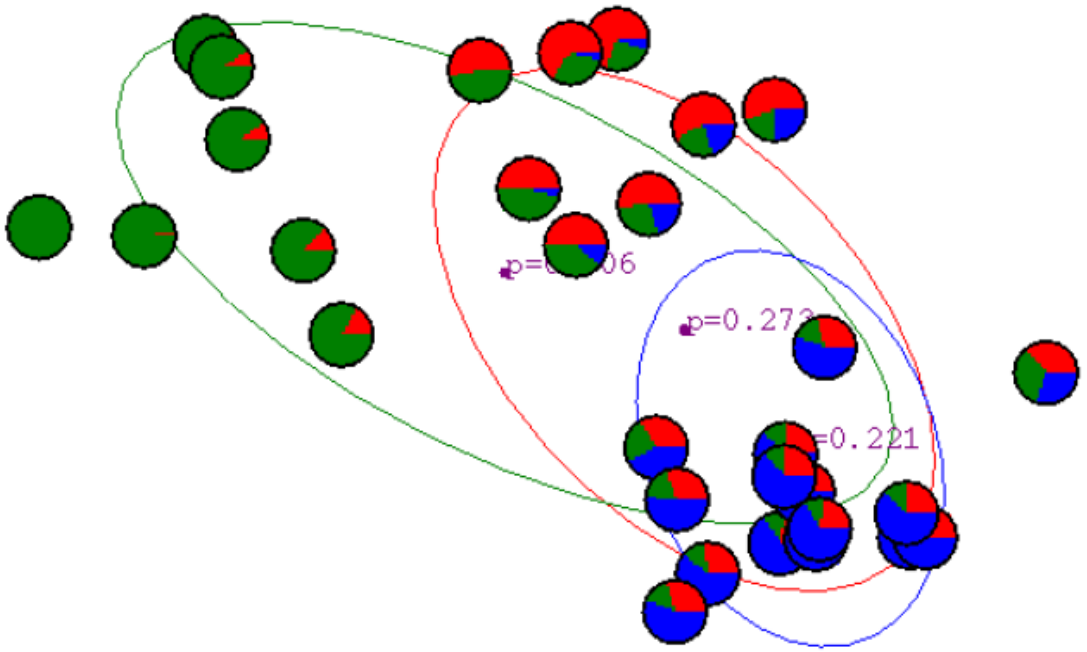
$$\Sigma_k = \frac{\sum_{j=1}^{m} P(y = k|x^j)(x^j-\mu_k)(x^j-\mu_k)^T}{\sum_{j=1}^{m} P(y = k|x^j)}$$

M-step

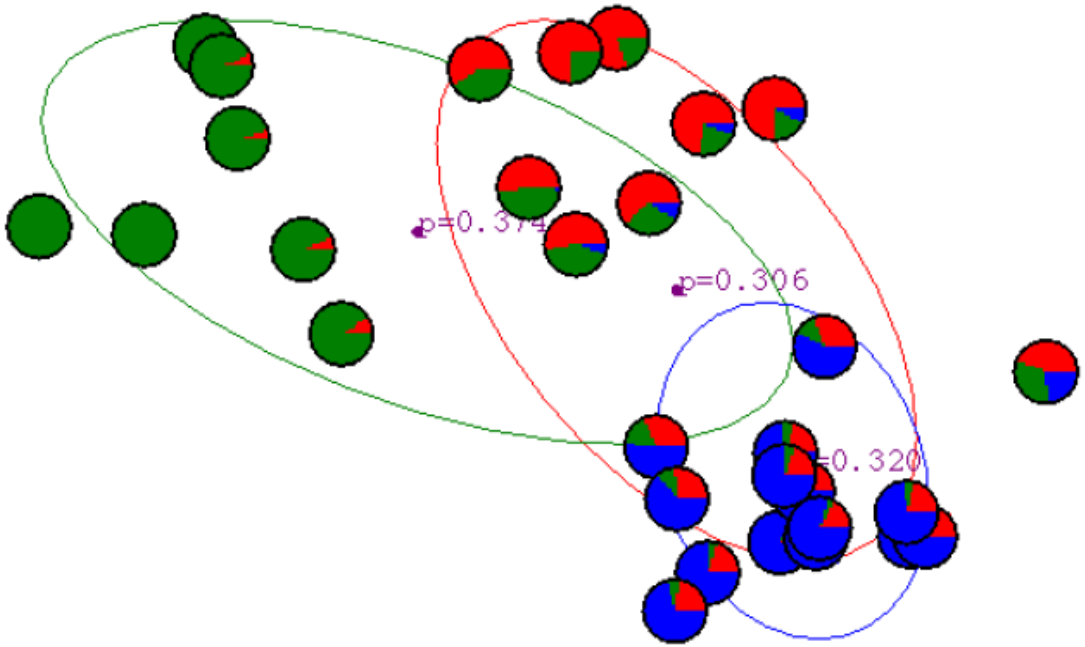
Gaussian Mixture Example: Start



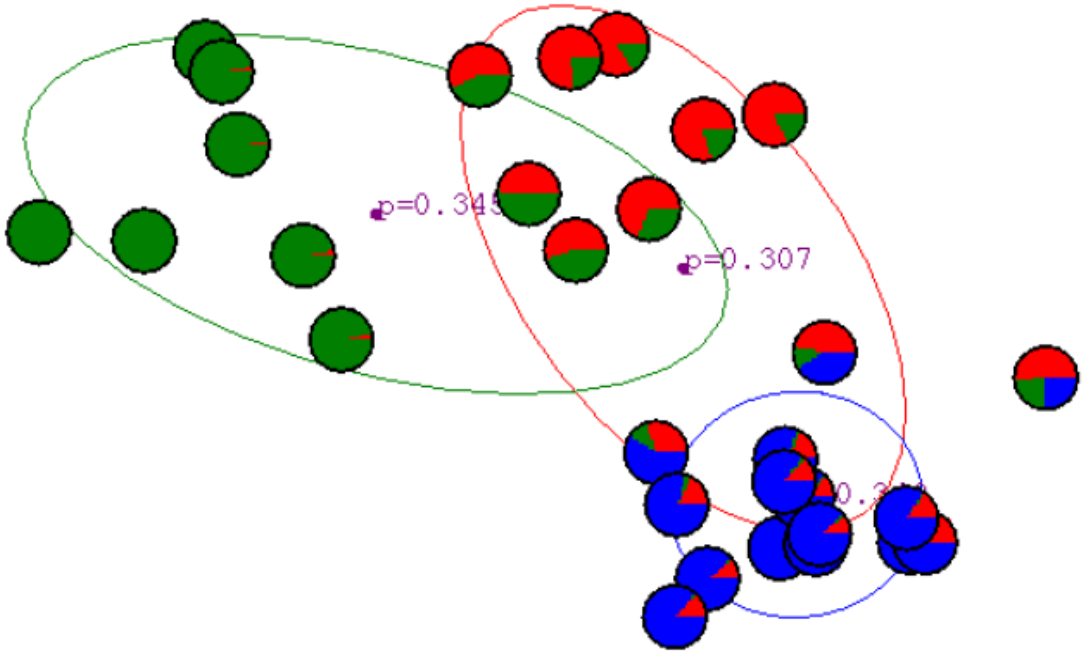
After first iteration



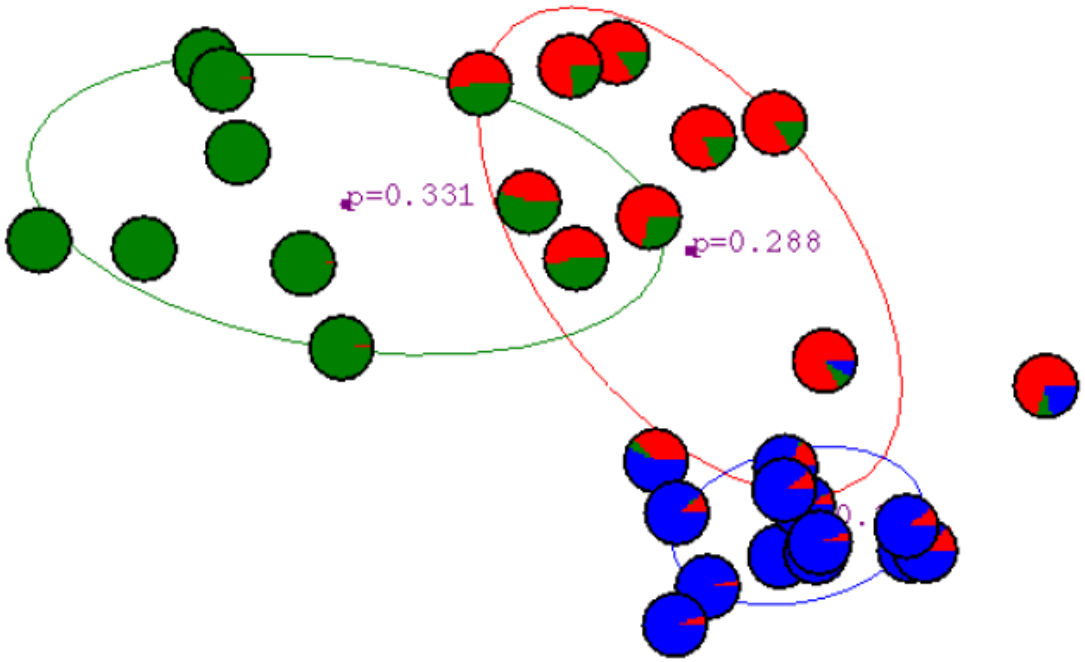
After 2nd iteration



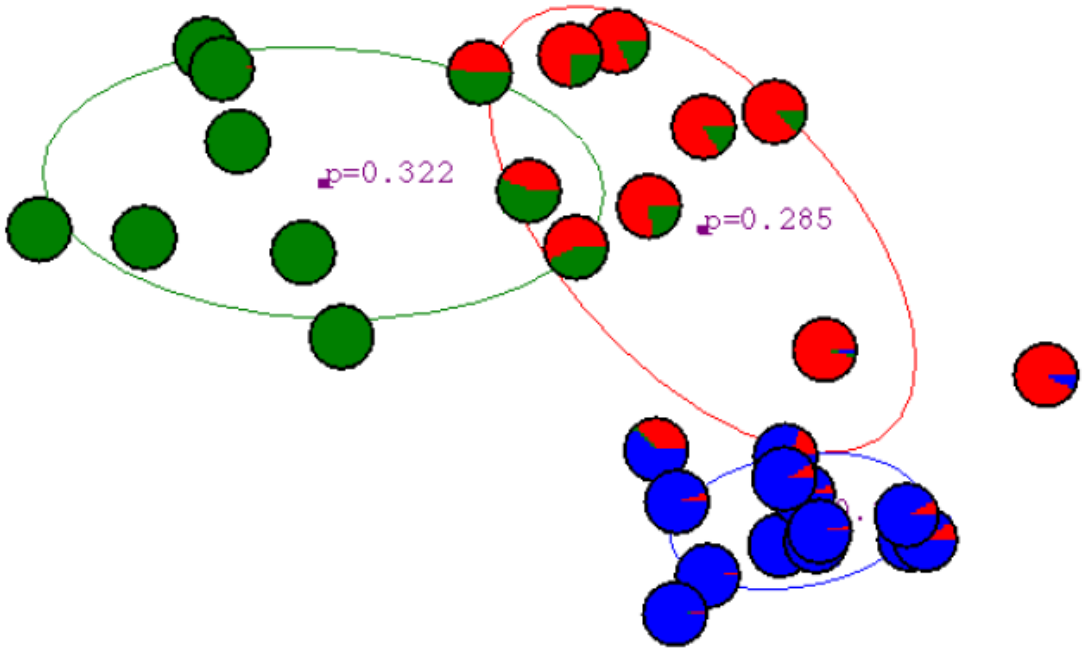
After 3rd iteration



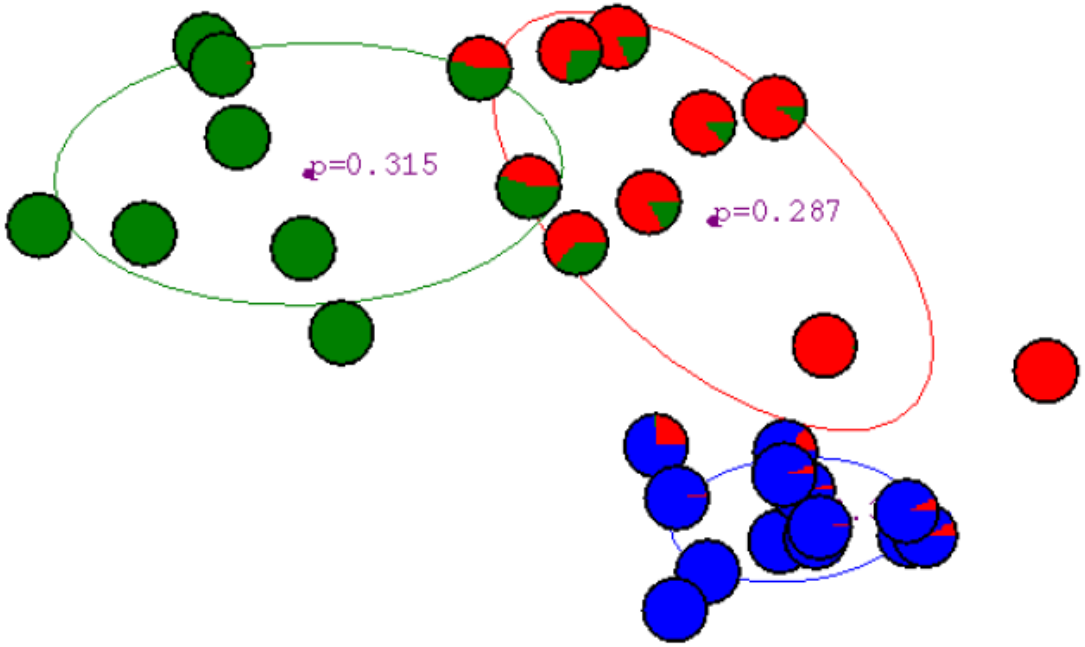
After 4th iteration



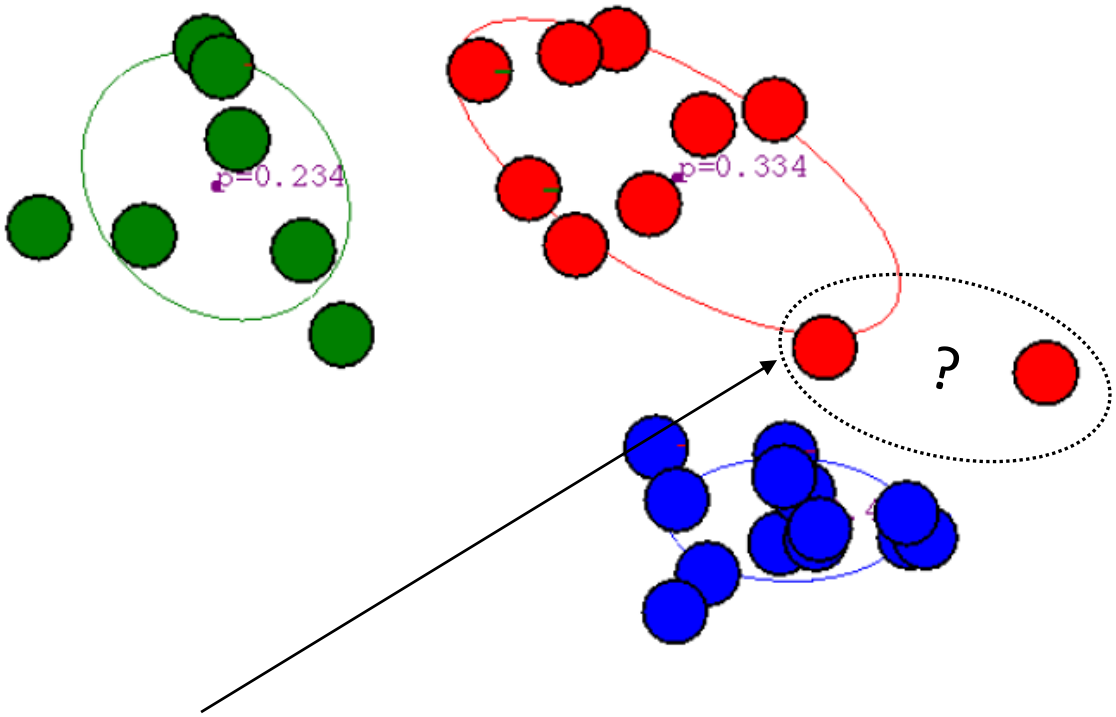
After 5th iteration



After 6th iteration



After 20th iteration



Q: Why are these two points red?

because of the shape of the Gaussian. Blue는 보면, 훨씬 좀더 좁게 돼 있잖아
빨간거는 스트레치 돼있는데
그래서 요 두개가 레드로 되는거양

K mean이면 blue일거야 왜냠 블루센터랑 더 가까우니까 rather than center of the red cluster
이게 difference btw K means and Gaussian

Behavior of EM

- It is guaranteed to converge
  - Convergence proof is based on the fact that the marginal likelihood $P(x; \Theta)$ must increase or remain same between iterations (not obvious)
  - In practice it may converge slowly, one can stop early if the change in log-likelihood is smaller than a threshold
- It converges to a local optimum
  - Multiple restart is recommended
  - Choosing the solution with the highest marginal likelihood