

# Clustering

Aarti Singh

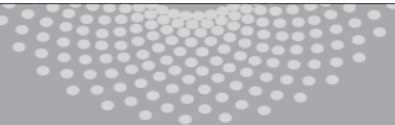
Machine Learning 10-315

Nov 18, 2020

Some slides courtesy of Eric Xing, Carlos Guestrin



**MACHINE LEARNING** DEPARTMENT



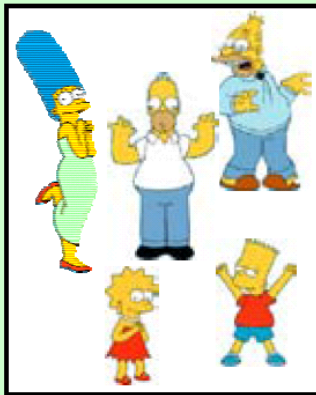
**Carnegie Mellon.**  
School of Computer Science

# What is clustering?

$x_i$  →  $i^{\text{th}}$  data point / object

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

## Clustering is subjective



Simpson's Family



School Employees



Females



Males

# 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 approach - think in terms of a distance (rather than similarity) between vectors or correlations between random variables.

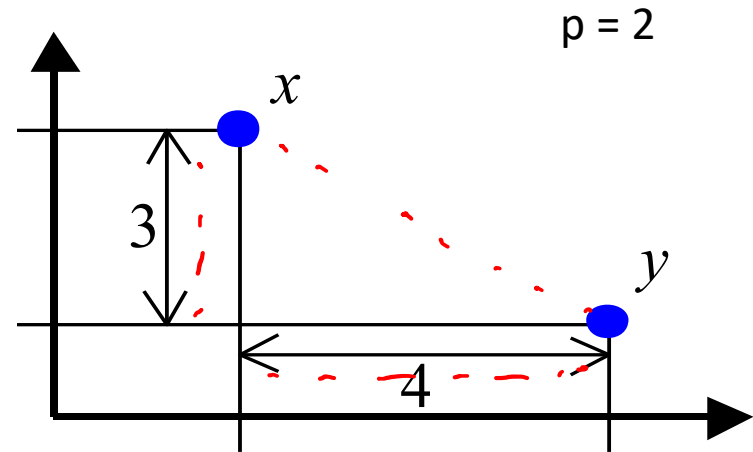
$$x_i \in \mathbb{R}^d$$

# Distance metrics

*= d dimension*

$$x = (x_1, x_2, \dots, x_p)$$

$$y = (y_1, y_2, \dots, y_p)$$



Euclidean distance

$$d(x, y) = \sqrt{\sum_{i=1}^p |x_i - y_i|^2}$$

5

Manhattan distance

$$d(x, y) = \sum_{i=1}^p |x_i - y_i|$$

7

Sup-distance

$$d(x, y) = \max_{1 \leq i \leq p} |x_i - y_i|$$

4

# Correlation coefficient

$$x = (x_1, x_2, \dots, x_p)$$

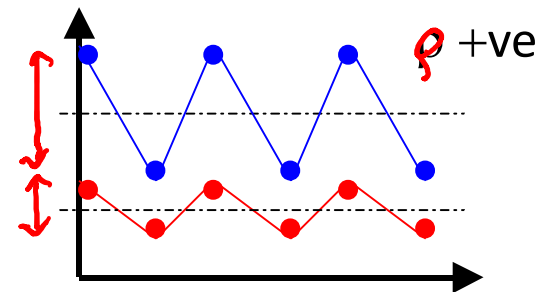
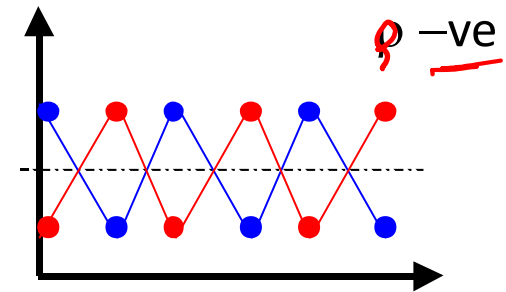
$$y = (y_1, y_2, \dots, y_p)$$

Random vectors (e.g. expression levels of two genes under various drugs)

Pearson correlation coefficient

$$\rho(x, y) = \frac{\sum_{i=1}^p (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^p (x_i - \bar{x})^2 \times \sum_{i=1}^p (y_i - \bar{y})^2}}$$

$$\text{where } \bar{x} = \frac{1}{p} \sum_{i=1}^p x_i \text{ and } \bar{y} = \frac{1}{p} \sum_{i=1}^p y_i.$$



# Partitioning Algorithms

- Partitioning method: Construct a partition of  $n$  objects into a set of  $K$  clusters
- Given: a set of objects and the number  $K$
- Find: a partition of  $K$  clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions
  - Effective heuristic method: K-means algorithm

# K-Means

## Algorithm

**Input** – Desired number of clusters,  $k$

*and data points  $\{x_i\}_{i=1}^n$*

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

**Iterate** –

*$\{\mu_k\}$*

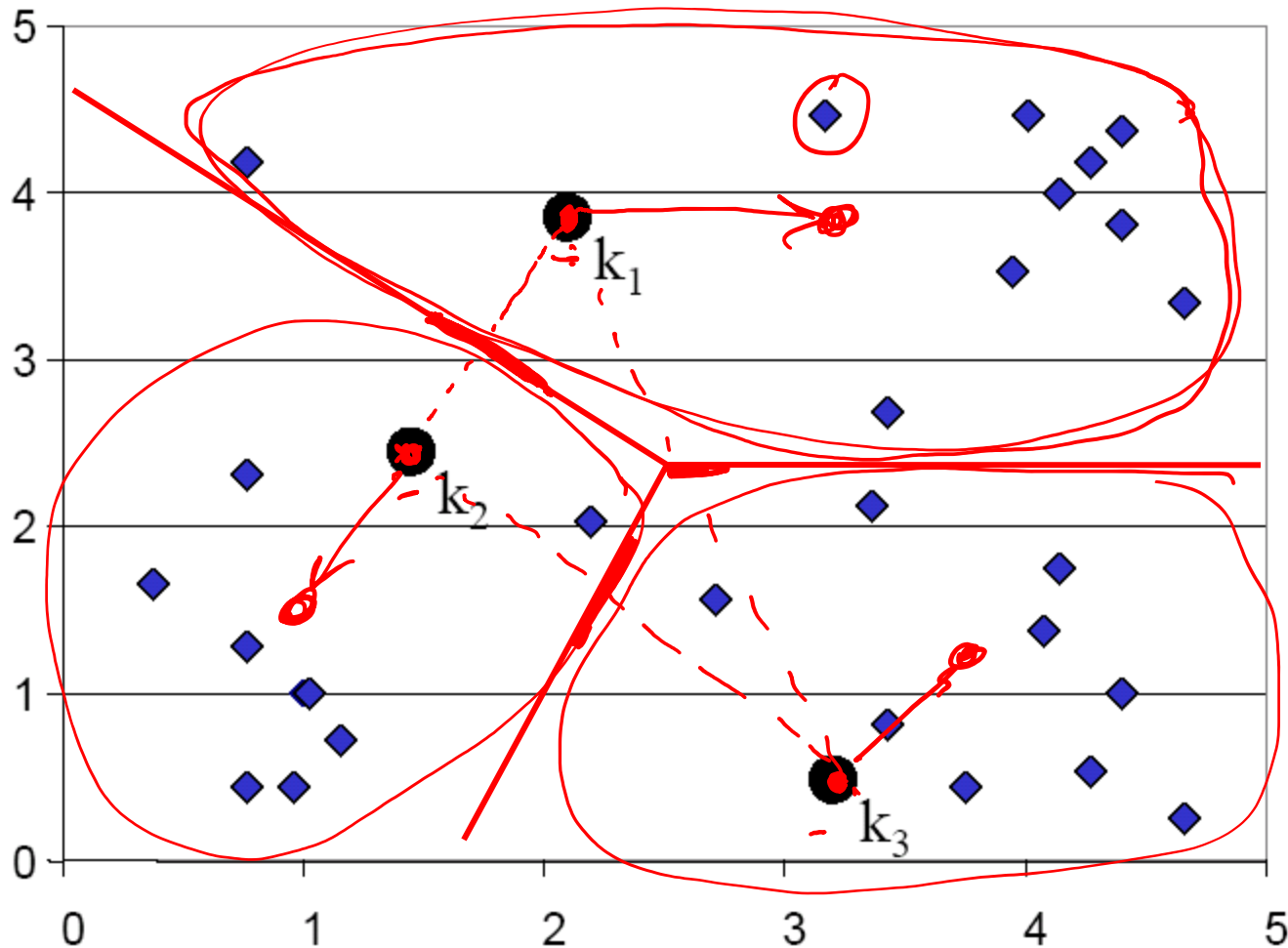
1. Assign points to the nearest cluster centers
2. Re-estimate the  $k$  cluster centers (aka the **centroid** or **mean**), by assuming the memberships found above are correct.

$$\vec{\mu}_k = \frac{1}{C_k} \sum_{i \in C_k} \vec{x}_i$$

**Termination** –

If none of the objects changed membership in the last iteration, exit. Otherwise go to 1.

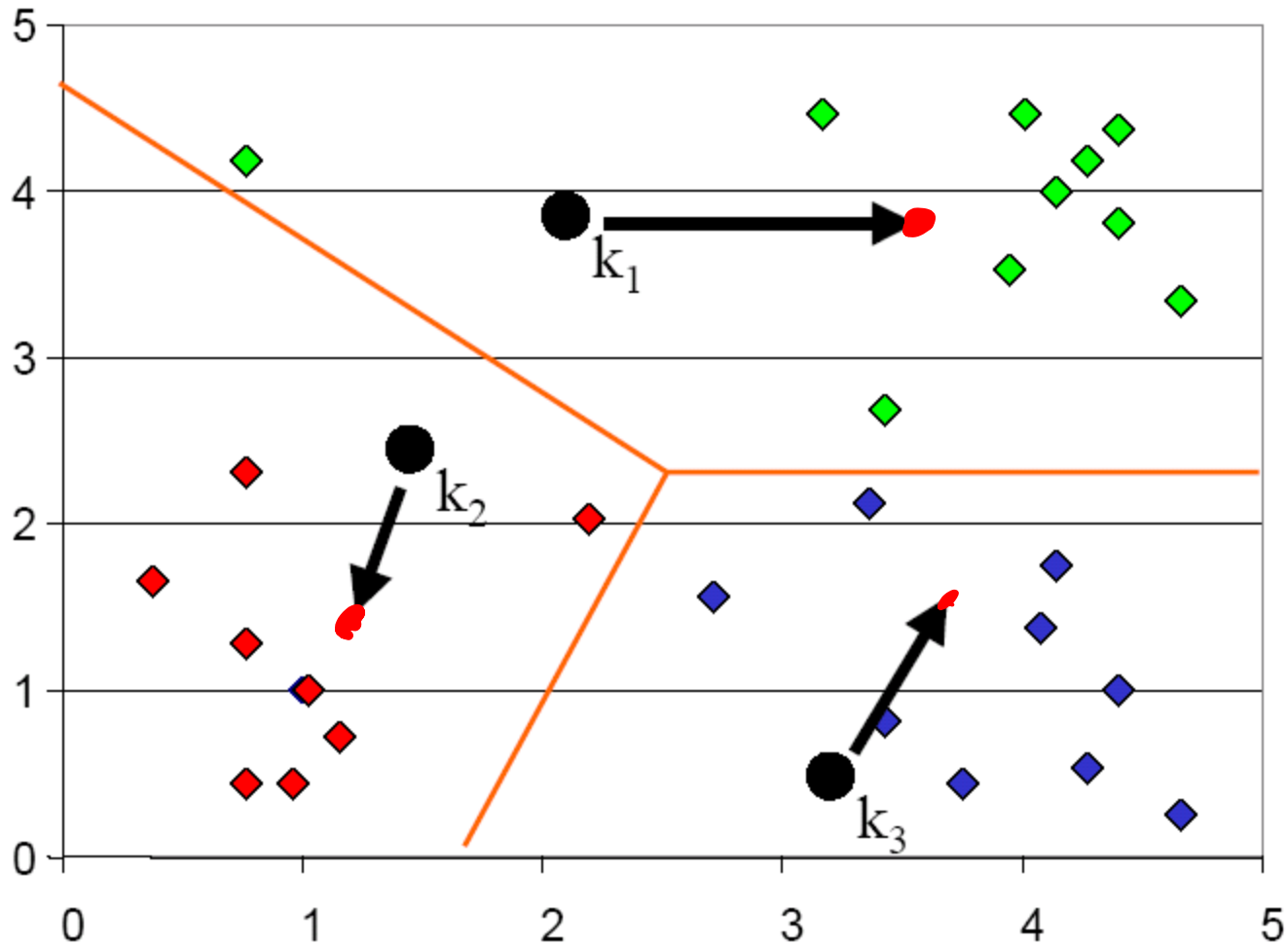
# K-means Clustering: Step 1



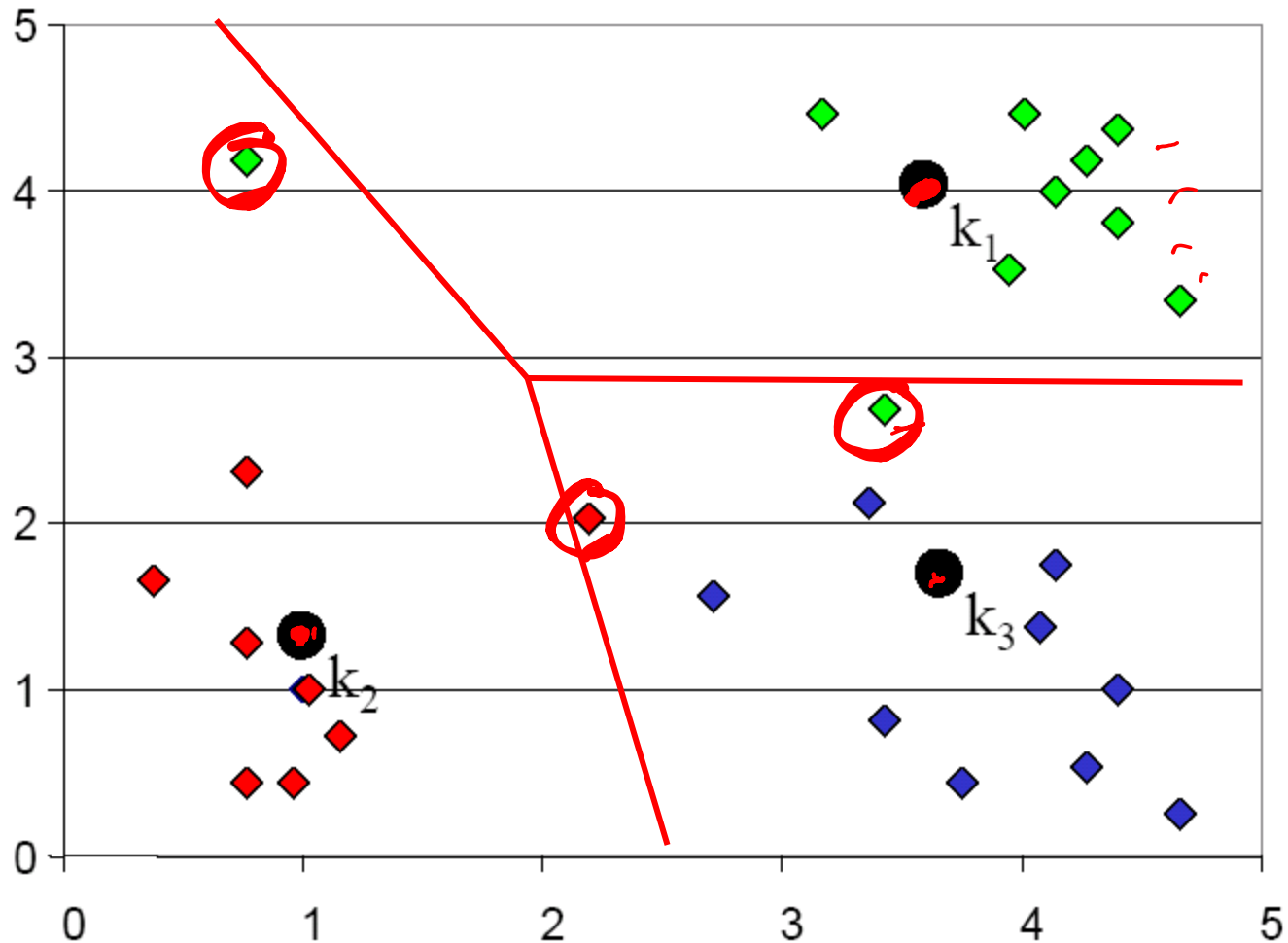
Voronoi  
diagram



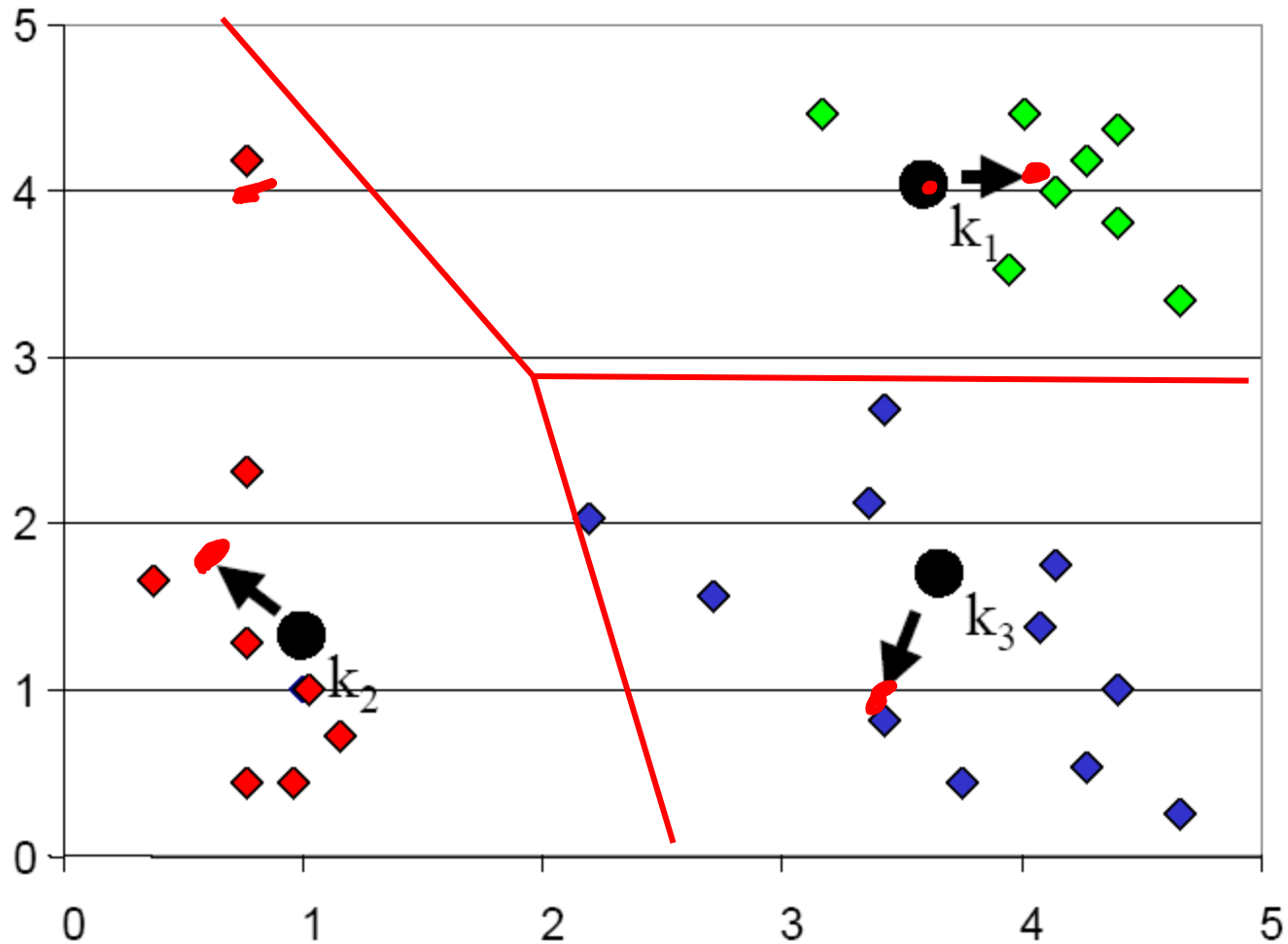
# K-means Clustering: Step 2



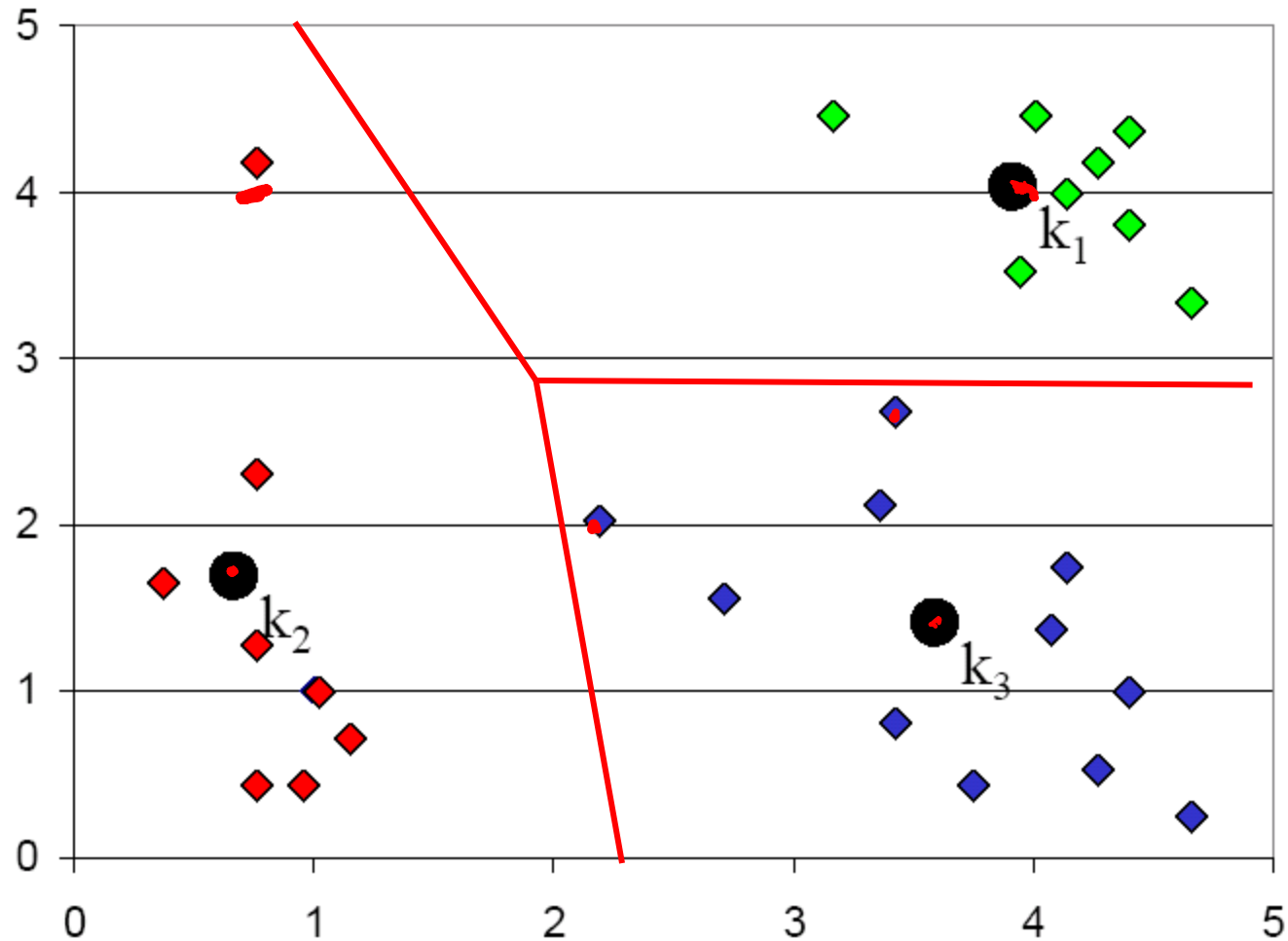
# K-means Clustering: Step 3



# K-means Clustering: Step 4



# K-means Clustering: Step 5



# K-means Recap ...

- Randomly initialize  $k$  centers
  - $\mu^{(0)} = \mu_1^{(0)}, \dots, \mu_k^{(0)}$

# K-means Recap ...

- Randomly initialize  $k$  centers

- $\mu^{(0)} = \mu_1^{(0)}, \dots, \mu_k^{(0)}$

Iterate  $t = 0, 1, 2, \dots$

- **Classify:** Assign each point  $j \in \{1, \dots, m\}$  to nearest center:

- $C^{(t)}(j) \leftarrow \arg \min_{i=1, \dots, k} \|\mu_i^{(t)} - x_j\|_2^2$

# K-means Recap ...

- Randomly initialize  $k$  centers

- $\mu^{(0)} = \mu_1^{(0)}, \dots, \mu_k^{(0)}$

Iterate  $t = 0, 1, 2, \dots$

- **Classify:** Assign each point  $j \in \{1, \dots, m\}$  to nearest center:

*cluster assignment of data point  $j$  at time  $t$*

- $C^{(t)}(j) \leftarrow \arg \min_{i=1, \dots, k} \|\mu_i^{(t)} - x_j\|_2^2$



- **Recenter:**  $\mu_i$  becomes centroid of its points:

- $\mu_i^{(t+1)} \leftarrow \arg \min_{\mu} \sum_{j: C^{(t)}(j)=i} \|\mu - x_j\|_2^2$   $i \in \{1, \dots, k\}$

*$\frac{1}{n} \sum_{j: C^{(t)}(j)=i} x_j$*

- Equivalent to  $\mu_i \leftarrow$  average of its points!

# What is K-means optimizing?

*Objective*

- Potential function  $F(\underline{\mu}, C)$  of centers  $\underline{\mu}$  and point allocations  $C$ :

$$\begin{aligned} \underline{F}(\underline{\mu}, C) &= \sum_{j=1}^m \|\underline{\mu}_{C(j)} - x_j\|^2 \\ &= \sum_{i=1}^k \sum_{j:C(j)=i} \|\mu_i - x_j\|^2 \end{aligned}$$

- Optimal K-means:

$$\square \min_{\underline{\mu}} \min_C F(\underline{\mu}, C)$$



# K-means algorithm

- Optimize potential function:

$$\min_{\mu} \min_C F(\mu, C) = \min_{\mu} \min_C \sum_{i=1}^k \sum_{j:C(j)=i} \| \mu_i - x_j \|^2 \quad \leftarrow \text{non-convex}$$

- K-means algorithm:** (coordinate descent on  $F$ )

**(1)** Fix  $\mu$ , optimize  $C$

**Expected** cluster assignment

**(2)** Fix  $C$ , optimize  $\mu$

**Maximum** likelihood for center

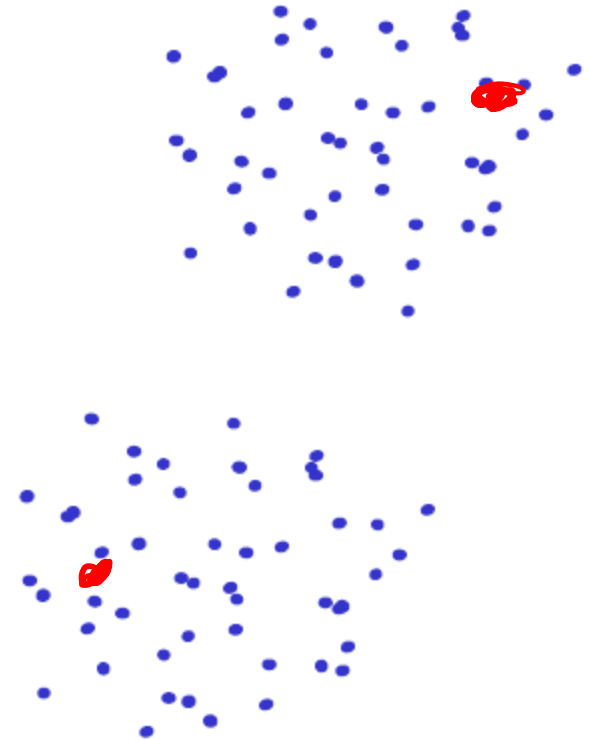
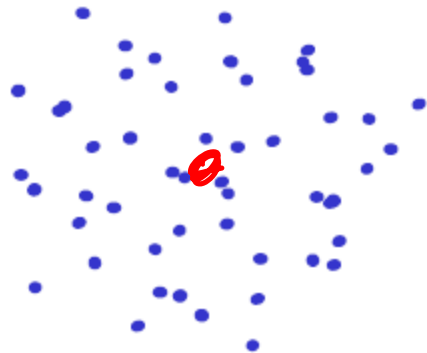
Next class, we will see a generalization of this approach:

**EM algorithm**

# Seed Choice

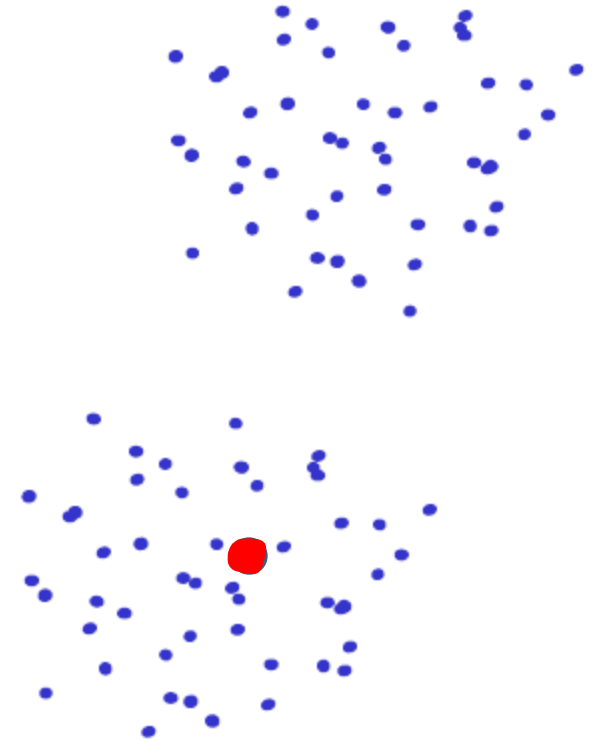
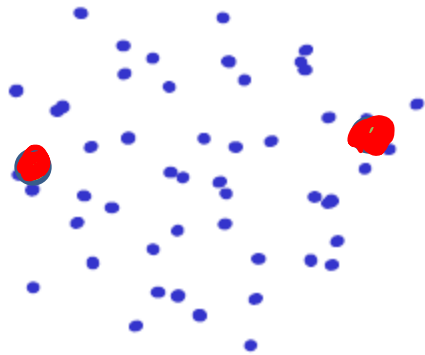
random choice  
of cluster centers

- Results are quite sensitive to seed selection.



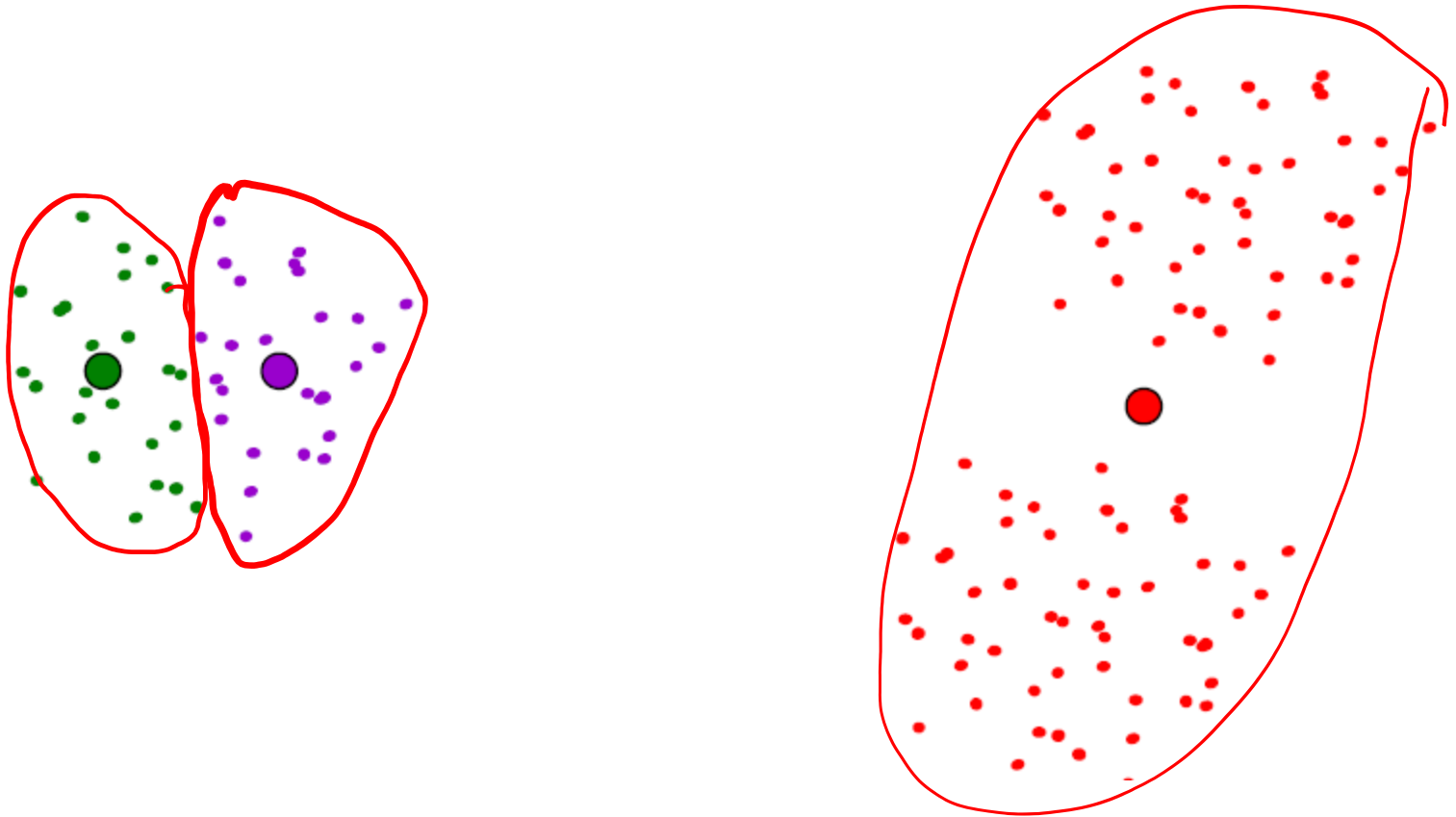
# Seed Choice

- Results are quite sensitive to seed selection.




# Seed Choice

- Results are quite sensitive to seed selection.



# Seed Choice

- Results can vary based on random seed selection.
  - Some seeds can result in poor convergence rate, or convergence to sub-optimal clustering.
    - Try out multiple starting points (very important!!!) 
    - k-means ++ algorithm of Arthur and Vassilvitskii
- key idea: choose centers that are far apart
- (probability of picking a point as cluster center  $\propto$  distance from nearest center picked so far)

# Other Issues

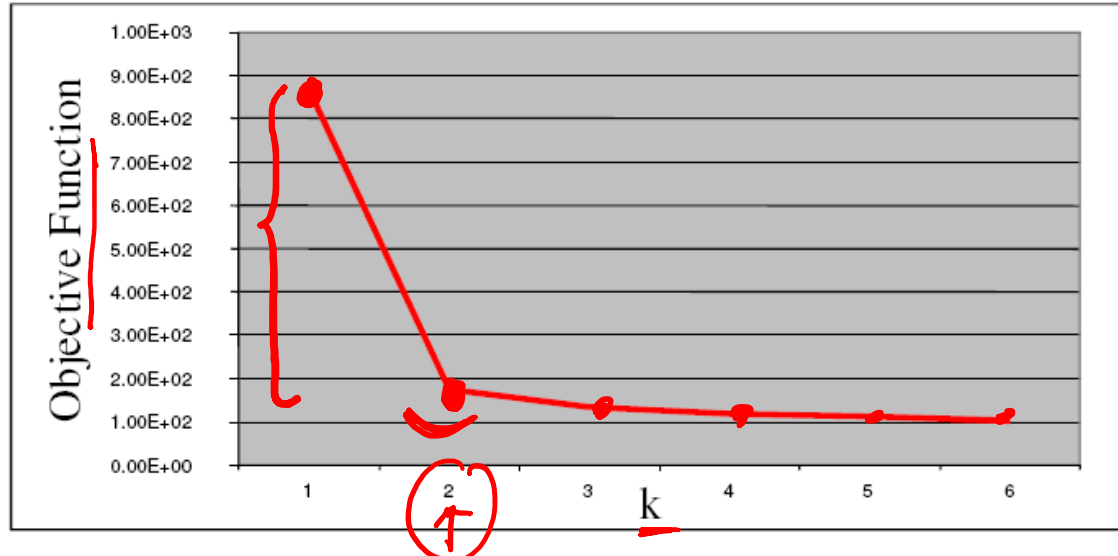
- Number of clusters K

- Objective function

$$\sum_{j=1}^m \|\mu_{C(j)} - x_j\|^2$$



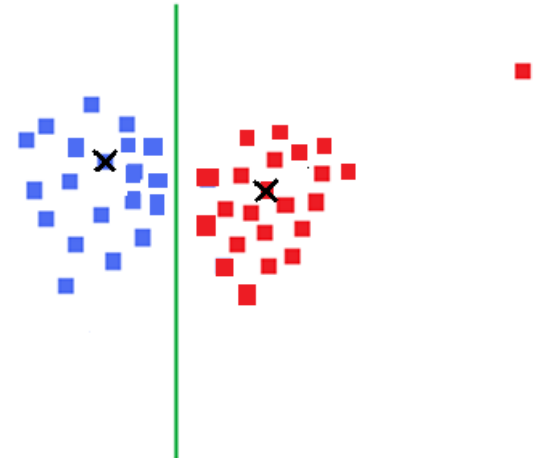
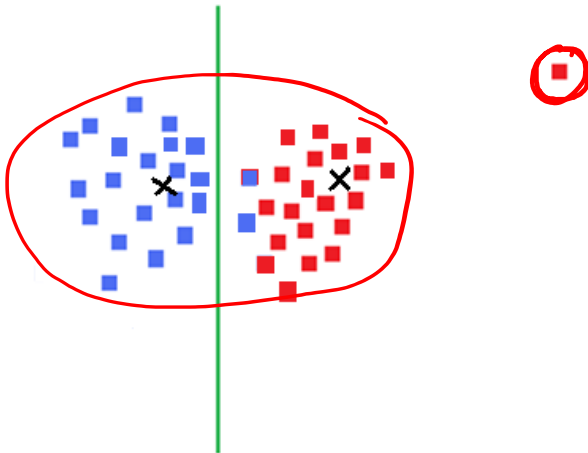
- Look for “Knee” in objective function



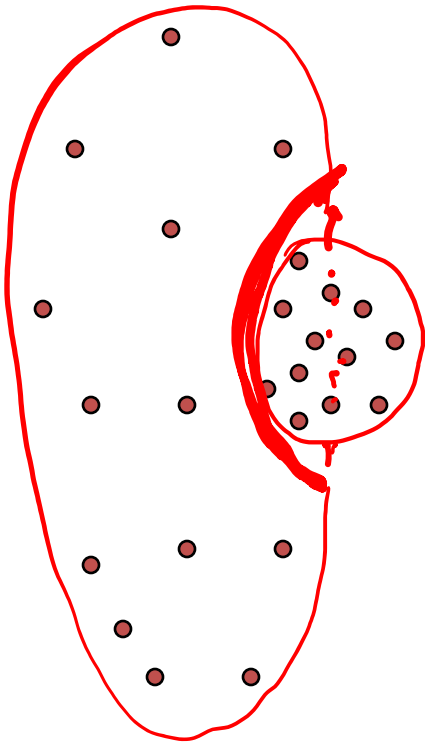
- Can you pick K by minimizing the objective over K? *No!*

# Other Issues

- Shape of clusters
  - Assumes isotropic, equal variance, convex clusters
- Sensitive to Outliers
  - use K-medoids



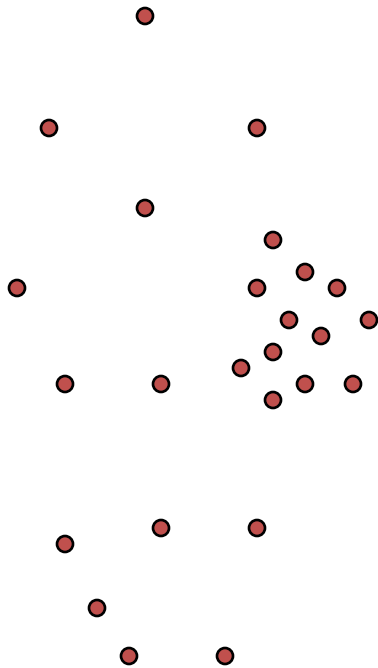
# (One) bad case for K-means



- Clusters may overlap •
- Some clusters may be “wider” than others. •
- Clusters may not be linearly separable •



# (One) bad case for K-means



- Clusters may overlap
- Some clusters may be “wider” than others
- Clusters may not be linearly separable

# Partitioning Algorithms

- K-means
  - **hard assignment**: each object belongs to only one cluster
- Mixture modeling
  - **soft assignment**: probability that an object belongs to a cluster

*Generative approach*

# Gaussian Mixture Model

Mixture of K Gaussian distributions: (Multi-modal distribution)

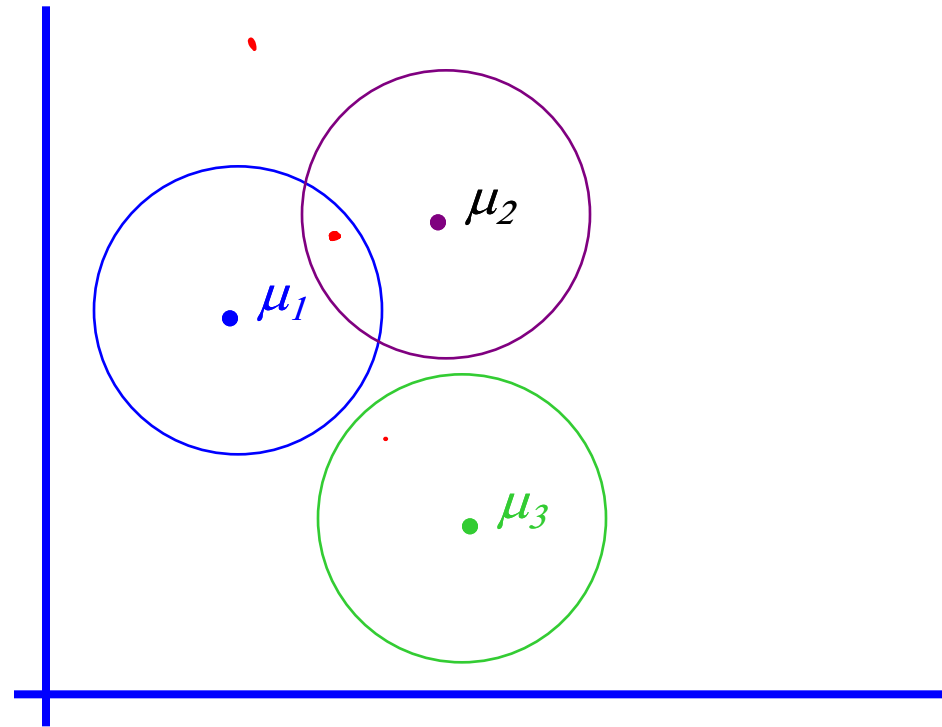
$$p(x|y=i) \sim N(\mu_i, \sigma^2 I)$$

$$p(x) = \sum_i p(x|y=i) P(y=i)$$

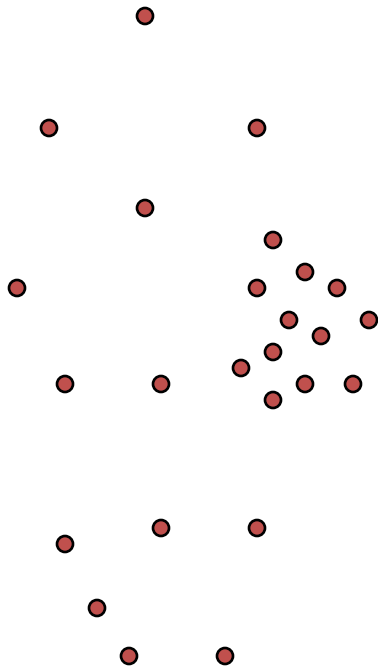
Mixture component

Mixture proportion

$$\begin{aligned} p(x) &= \sum_i p(x, y=i) \\ &= \sum_i p(x|y=i) p(y=i) \end{aligned}$$

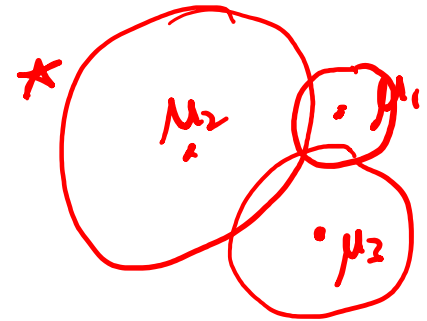


# (One) bad case for K-means



- Clusters may overlap ✓
- Some clusters may be “wider” than others
- Clusters may not be linearly separable

# General GMM



GMM – Gaussian Mixture Model (Multi-modal distribution)

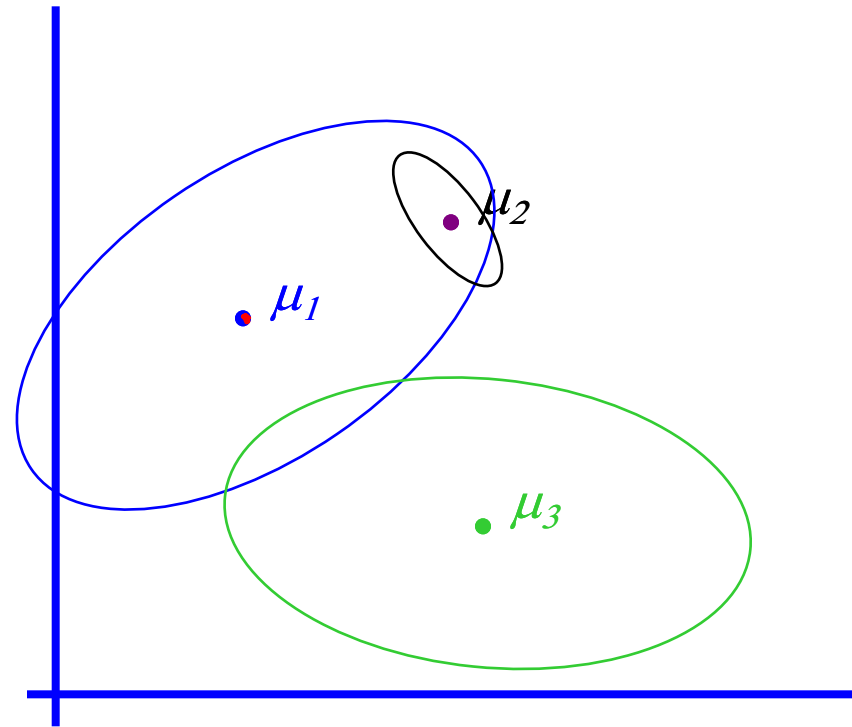
$$p(x|y=i) \sim N(\mu_i, \Sigma_i)$$

$\sigma_i^2 I \rightarrow *$

$$p(x) = \sum_i p(x|y=i) P(y=i)$$

Mixture component

Mixture proportion



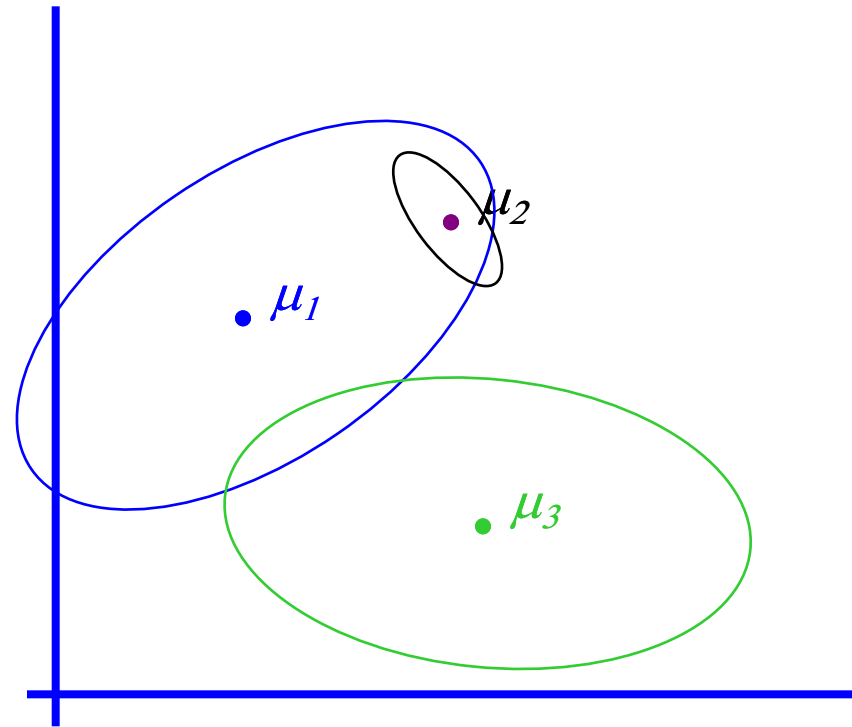
# General GMM

GMM – Gaussian Mixture Model (Multi-modal distribution)

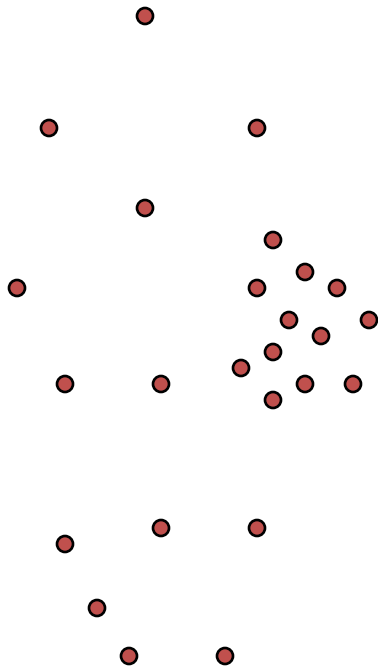
- There are  $k$  components
- Component  $i$  has an associated mean vector  $\mu_i$
- Each component generates data from a Gaussian with mean  $\mu_i$  and covariance matrix  $\Sigma_i$

Each data point is generated according to the following recipe:

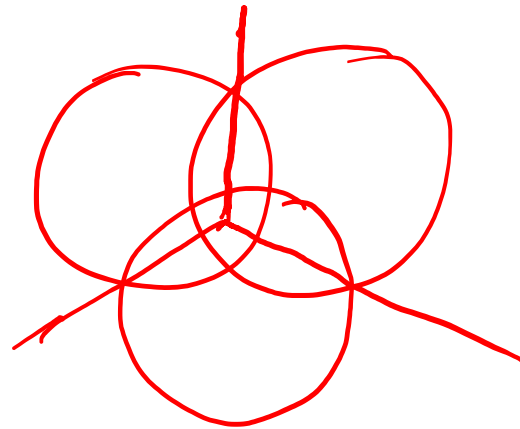
- 1) Pick a component at random:  
Choose component  $i$  with probability  $P(y=i)$
- 2) Datapoint  $x \sim N(\mu_i, \Sigma_i)$



# (One) bad case for K-means



- Clusters may overlap ✓
- Some clusters may be “wider” than others ✓
- Clusters may not be linearly separable



# General GMM

GMM – Gaussian Mixture Model (Multi-modal distribution)

$$p(x|y=i) \sim N(\mu_i, \Sigma_i)$$

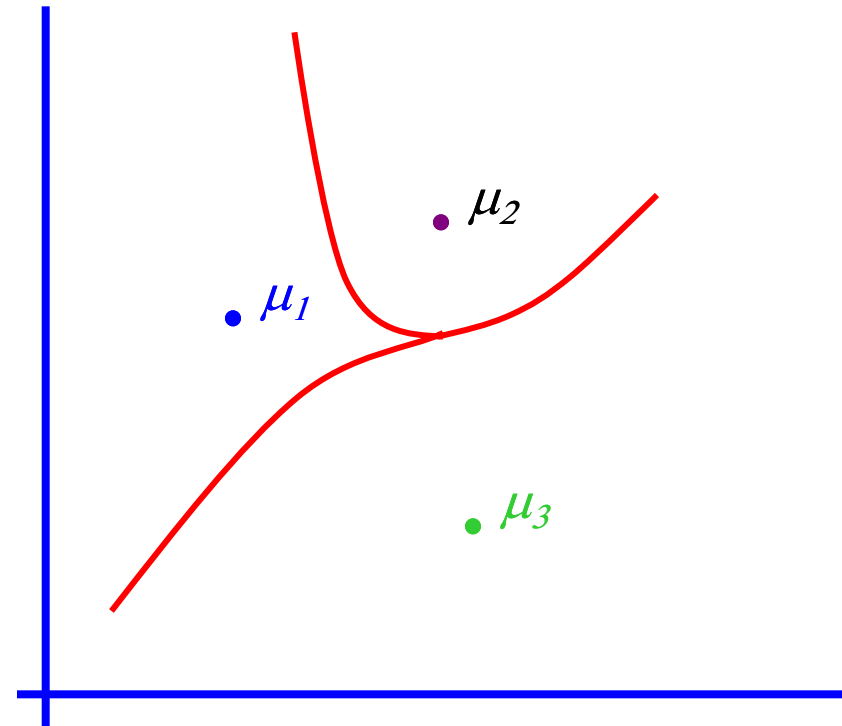
Gaussian Bayes Classifier:

$$\log \frac{P(y = i | \mathbf{x})}{P(y = j | \mathbf{x})}$$

$$= \log \frac{p(\mathbf{x} | y = i)P(y = i)}{p(\mathbf{x} | y = j)P(y = j)}$$

$$= \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{w}^T \mathbf{x}$$

$\Sigma_i \neq \Sigma_j$



Depend on  $\mu_1, \mu_2, \dots, \mu_K, \Sigma_1, \Sigma_2, \dots, \Sigma_K, P(y=1), \dots, P(Y=k)$

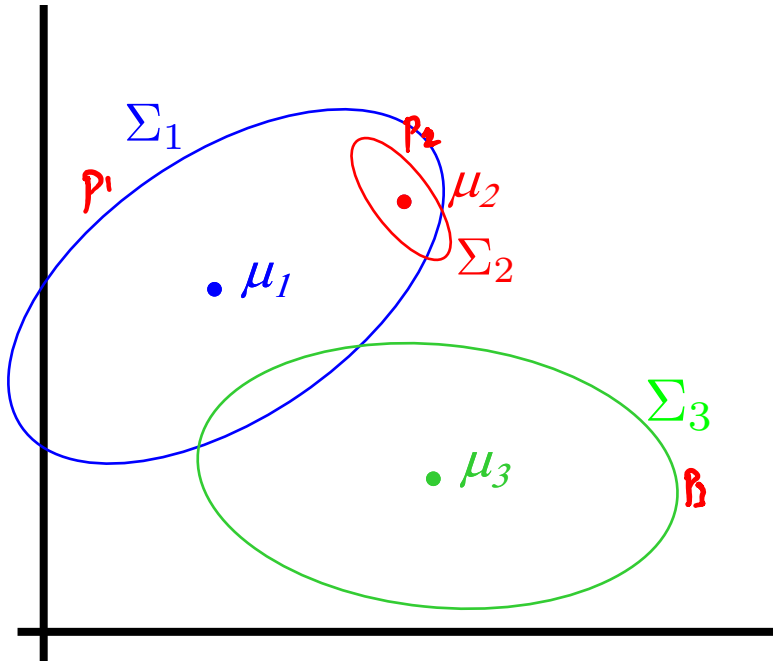
**“Quadratic Decision boundary”** – second-order terms don’t cancel out



# Learning General GMM

$$x_1, \dots, x_m \sim \underline{\underline{p(x)}} = \sum_{i=1}^k p(x|Y=i) \underbrace{P(Y=i)}_{\substack{\text{Mixture} \\ \text{proportion, } p_i}} \quad \leftarrow$$

Mixture component                      Mixture proportion,  $p_i$



Gaussian mixture model

$$\underline{p(x|Y=i)} \sim \underline{\mathcal{N}(\mu_i, \Sigma_i)}$$

**Parameters:**  $\{ \underline{p_i}, \underline{\mu_i}, \underline{\Sigma_i} \}_{i=1}^K$

- How to estimate parameters? Max Likelihood  
But don't know labels Y (recall Gaussian Bayes classifier)

# Learning General GMM

Maximize marginal likelihood:

$$\begin{aligned} \operatorname{argmax} \prod_j P(x_j) &= \operatorname{argmax} \prod_j \sum_{i=1}^K P(y_j=i, x_j) \\ \underbrace{\{\mu_i, \Sigma_i, P_i\}}_{i=1}^K &= \operatorname{argmax} \prod_j \sum_{i=1}^K \underbrace{P(y_j=i)}_{P_i} \underbrace{p(x_j | y_j=i)}_{\mu_i, \Sigma_i} \end{aligned}$$

$P(y_j=i) = P(y=i)$  Mixture component  $i$  is chosen with prob  $P(y = i)$

$$= \operatorname{arg max} \prod_{j=1}^m \sum_{i=1}^k P(y = i) \frac{1}{\sqrt{\det(\Sigma_i)}} \exp\left[-\frac{1}{2} (x_j - \mu_i)^T \Sigma_i (x_j - \mu_i)\right]$$

How do we find the  $\mu_i, \Sigma_i$ s and  $P(y=i)$ s which give max. marginal likelihood?

\* Set  $\frac{\partial}{\partial \mu_i} \log \text{Prob} (\dots) = 0$  and solve for  $\mu_i$ 's. Non-linear not-analytically solvable

\* Use gradient descent: Doable, but often slow

# Expectation-Maximization (EM)

A general algorithm to deal with hidden data, but we will study it in the context of unsupervised learning (hidden labels)

- No need to choose step size as in Gradient methods. —
- EM is an Iterative algorithm with two linked steps:
  - E-step: fill-in hidden data ( $Y$ ) using inference
  - M-step: apply standard MLE/MAP method to estimate parameters  $\{\mu_i, \Sigma_i\}_{i=1}^k$
- This procedure monotonically improves the marginal likelihood (or leaves it unchanged). Thus it always converges to a local optimum of the likelihood.



# EM for spherical, same variance GMMs

Initialize:  $\{p_i, \mu_i, \Sigma_i\}_{i=1}^K$

$$p(x) = \sum_i p(x|y=i) p(y=i)$$

## E-step

Compute "expected" classes of all datapoints for each class

$$P(y = i | x_j, \mu_1, \dots, \mu_k) \propto \exp\left(-\frac{1}{2\sigma^2} \|x_j - \mu_i\|^2\right) P(y = i)$$

In K-means "E-step"  
we do hard assignment

EM does soft assignment

## M-step

Compute Max. like  $\mu$  given our data's class membership distributions (weights)

$$\mu_i = \frac{\sum_{j=1}^m P(y = i | x_j) x_j}{\sum_{j=1}^m P(y = i | x_j)}$$

Exactly same as MLE with  
weighted data

Iterate.

# EM for general GMMs

Iterate. On iteration  $t$  let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \dots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \dots \Sigma_k^{(t)}, p_1^{(t)}, p_2^{(t)} \dots p_k^{(t)} \}$$

$p_i^{(t)}$  is shorthand for estimate of  $P(y=i)$  on  $t$ 'th iteration

## E-step

Compute “expected” classes of all datapoints for each class

$$P(y = i | x_j, \lambda_t) \propto p_i^{(t)} p(x_j | \mu_i^{(t)}, \Sigma_i^{(t)})$$

Just evaluate a Gaussian at  $x_j$

## M-step

Compute MLEs given our data's class membership distributions (weights)

$$\mu_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t) x_j}{\sum_j P(y = i | x_j, \lambda_t)} \quad \Sigma_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t) (x_j - \mu_i^{(t+1)})(x_j - \mu_i^{(t+1)})^T}{\sum_j P(y = i | x_j, \lambda_t)}$$

$$p_i^{(t+1)} = \frac{\sum_j P(y = i | x_j, \lambda_t)}{m}$$

$m = \#$ data points