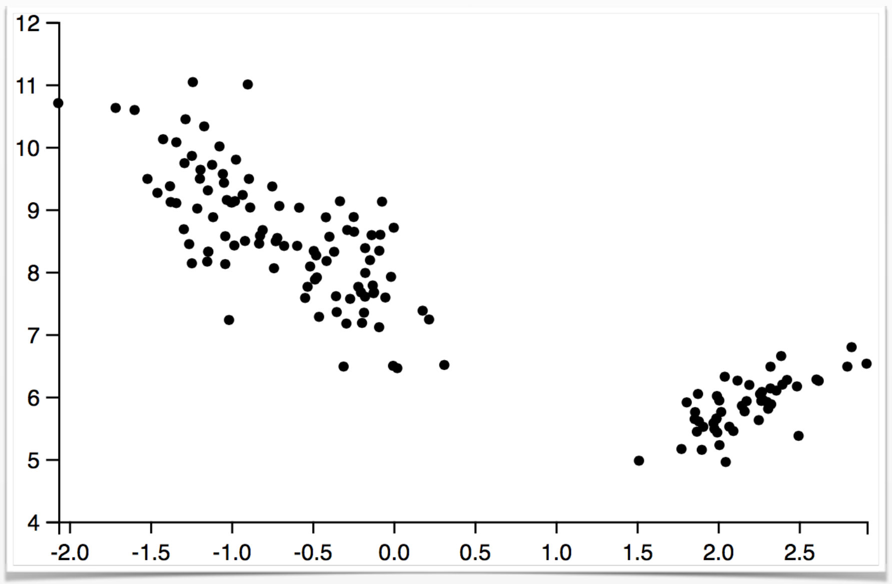
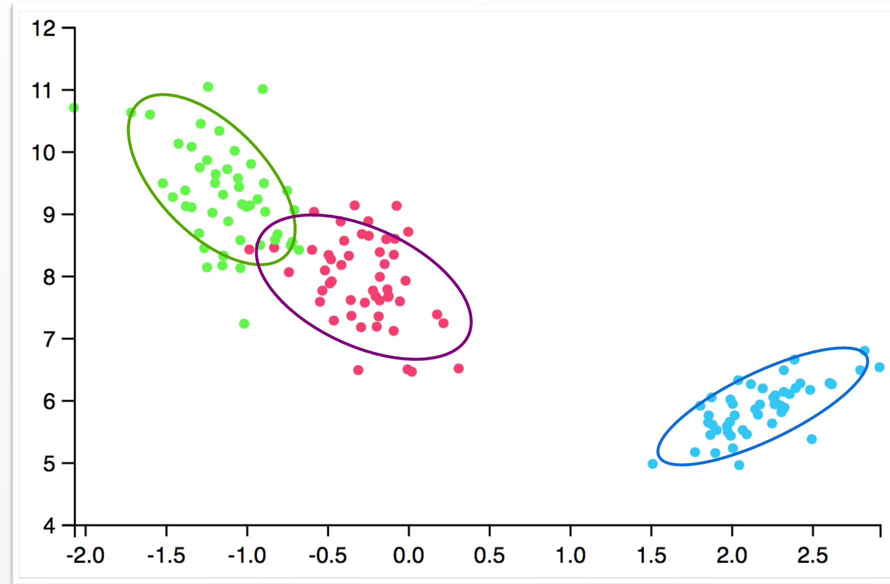
Clustering

Shantanu Jain

Clustering



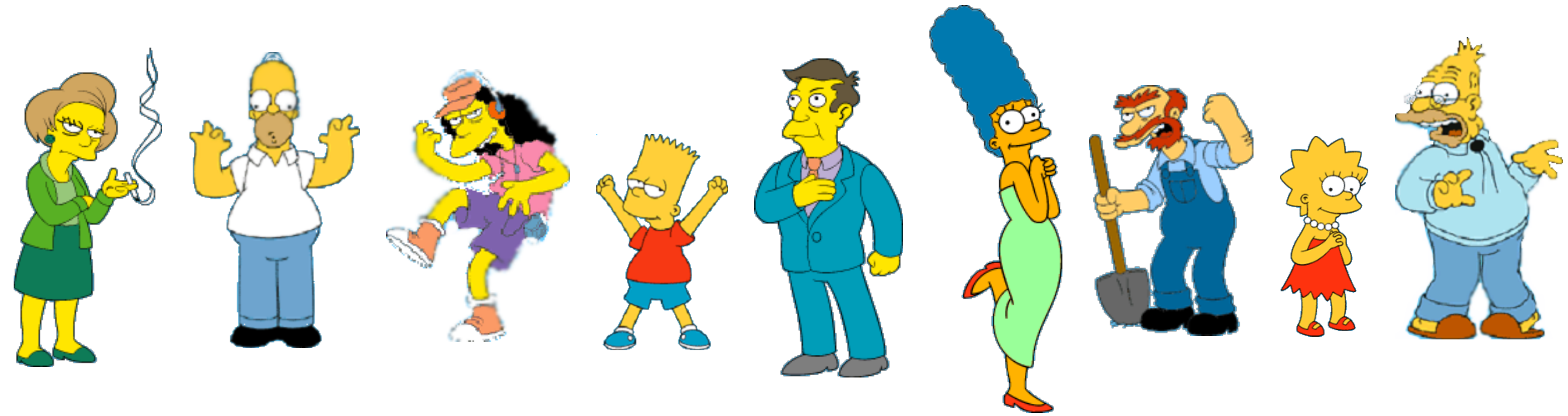


- Unsupervised learning (no labels for training)
- Group data into similar classes that
  - Maximize similarity *within clusters*
  - Minimize similarity *between clusters*

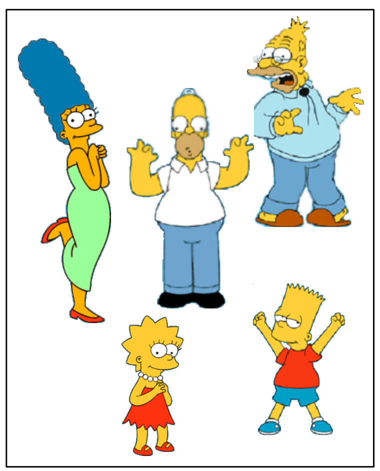
What is Similarity?

Can be hard to define, but we know it when we see it.

What is a natural grouping?

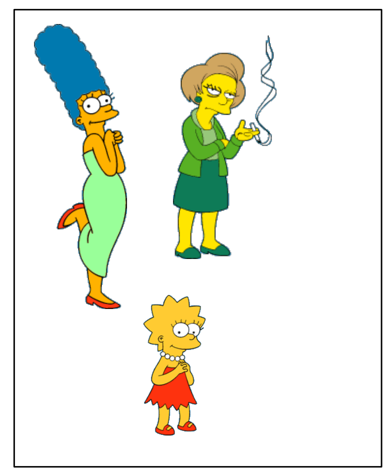


Choice of clustering criterion can be task-dependent

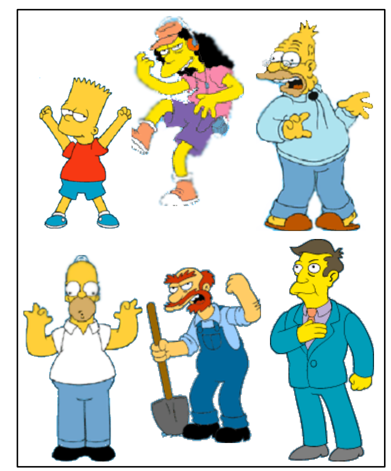


Simpson's Family

School Employees

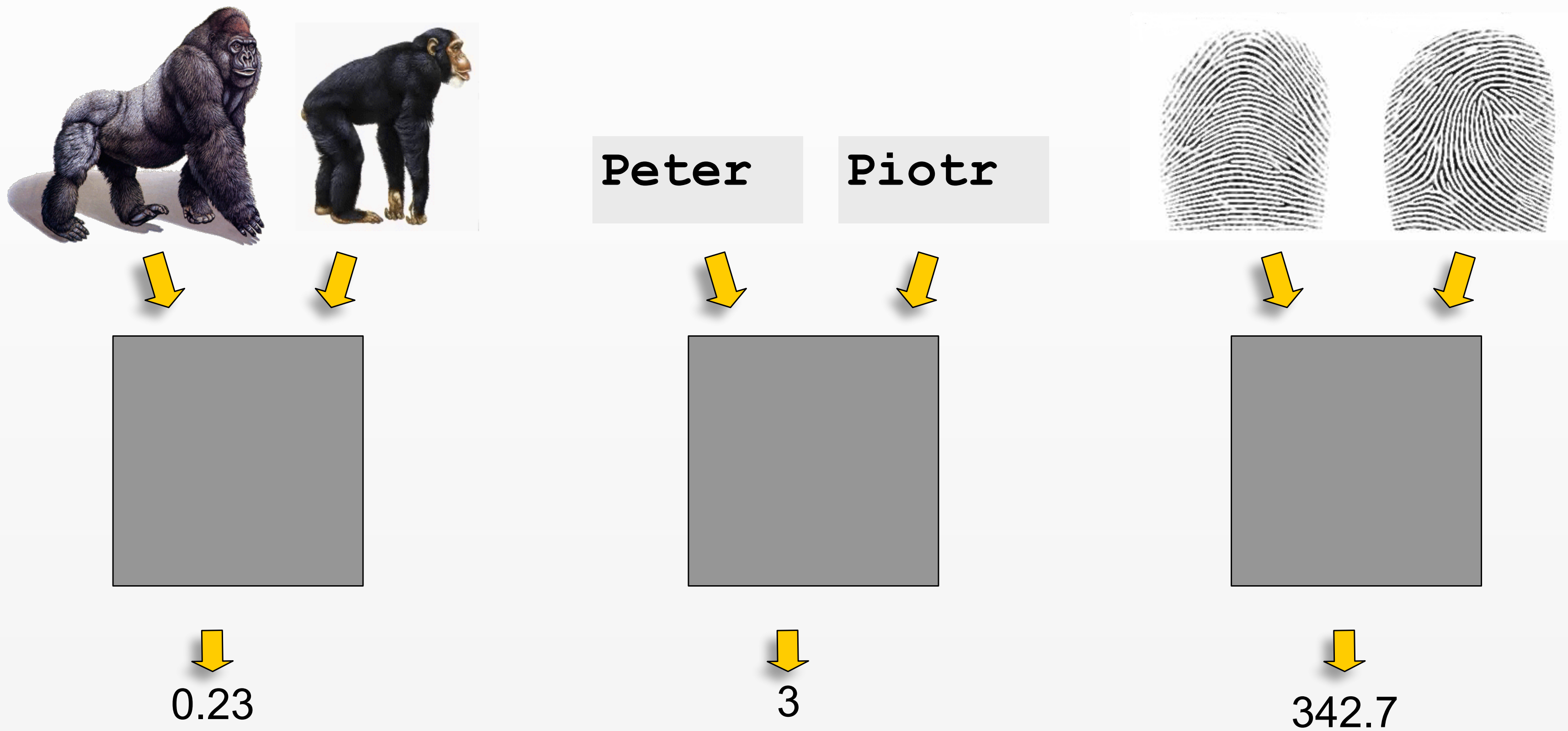


Females



Males

Defining Distance Measures



Dissimilarity/distance: $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$

Similarity: $s(\boldsymbol{x}_1, \boldsymbol{x}_2)$

Proximity: $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$

Common Distance Measures

$x = [x_1, x_2, \ldots x_k]$
$y = [y_1, y_2, \ldots y_k]$

- Euclidean Distance $$\sqrt{\left(\sum_{i=1}^{k}(x_i - y_i)^2\right)}$$
- Mahattan Distance $$\sum_{i=1}^{k} |x_i - y_i|$$
- Minkowski Distance $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}}$$

Common Similarity Measures

Inner Product

$$\langle x, y \rangle = x_1 y_1 + x_2 y_2 + \ldots x_k y_k$$

Cosine Similarity

$$cosine(x, y) = \frac{\langle x, y \rangle}{||x||\ ||y||}$$

Jaccard Similarity

$$J(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

If x and y are sets

Similarity: Kernel Functions

Formal Definition: Inner Product (in Hilbert space)

$$\boxed{k(x, x') = \langle \phi(x), \phi(x') \rangle} \longleftarrow \text{Feature map}$$

ϕ: $\mathbb{R}^D \longrightarrow \mathbb{R}^E$

In Practice: Can compute directly from x and x'

Radial Basis Function (RBF) $$k(\boldsymbol{x}, \boldsymbol{x}') = \exp^{-\frac{1}{2}\gamma^{-2}||\boldsymbol{x}-\boldsymbol{x}'||^2}$$

Squared Exponential (SE) $$k(\boldsymbol{x}, \boldsymbol{x}') = \exp^{-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{x}'}$$

Automatic Relevance Determination (ARD) $$k(\boldsymbol{x}, \boldsymbol{x}') = \exp^{-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\sigma_i^2}}$$

Inner Product vs Distance Measure

Inner Product

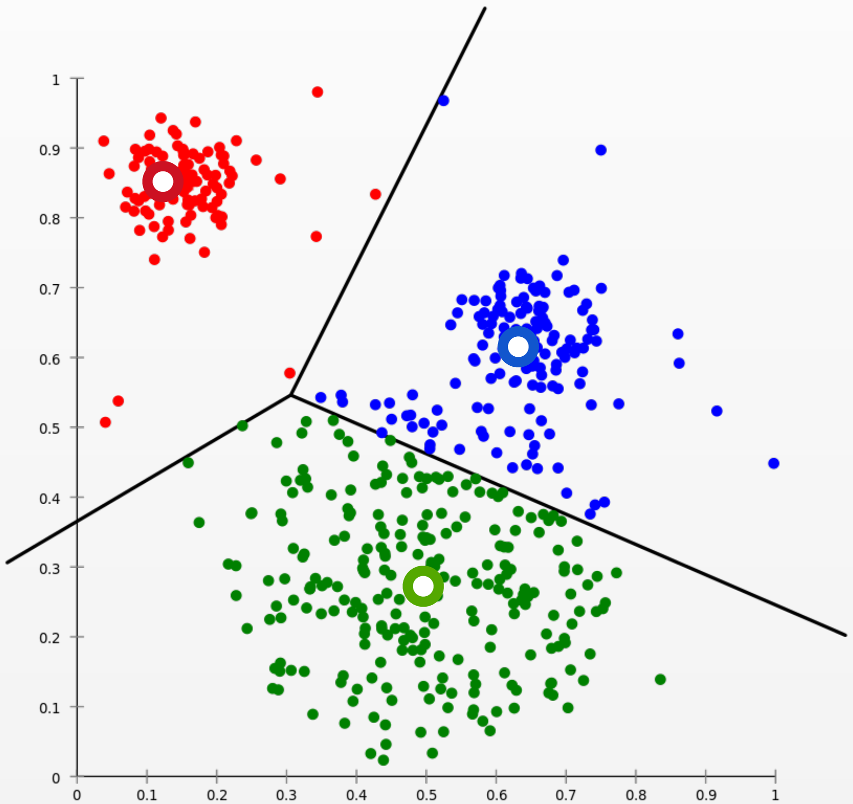
• $\langle A, B \rangle = \langle B, A \rangle$	*Symmetry*
• $\langle \alpha A, B \rangle = \alpha \langle A, B \rangle$	*Linearity*
• $\langle A, A \rangle \geq 0$, $\langle A, A \rangle = 0$ iff $A = 0$	*Postive-definiteness*

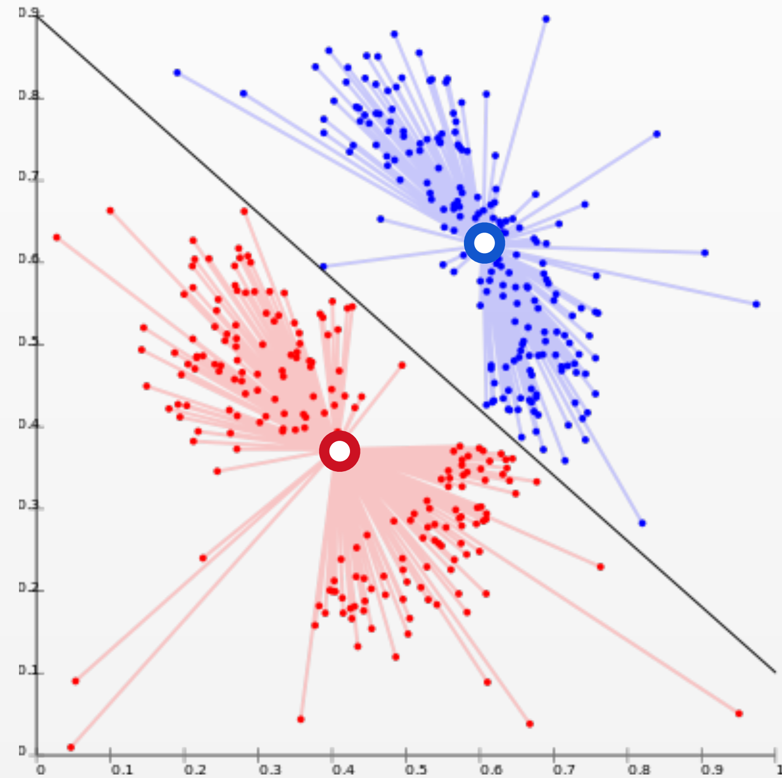
Distance Measure

• $D(A, B) = D(B, A)$	*Symmetry*
• $D(A, A) = 0$	*Constancy of Self-Similarity*
• $D(A, B) = 0$ iff $A = B$	*Positivity (Separation)*
• $D(A, B) \leq D(A, C) + D(B, C)$	*Triangular Inequality*

Types of Clustering

Centroid-based (K-means, K-medoids)

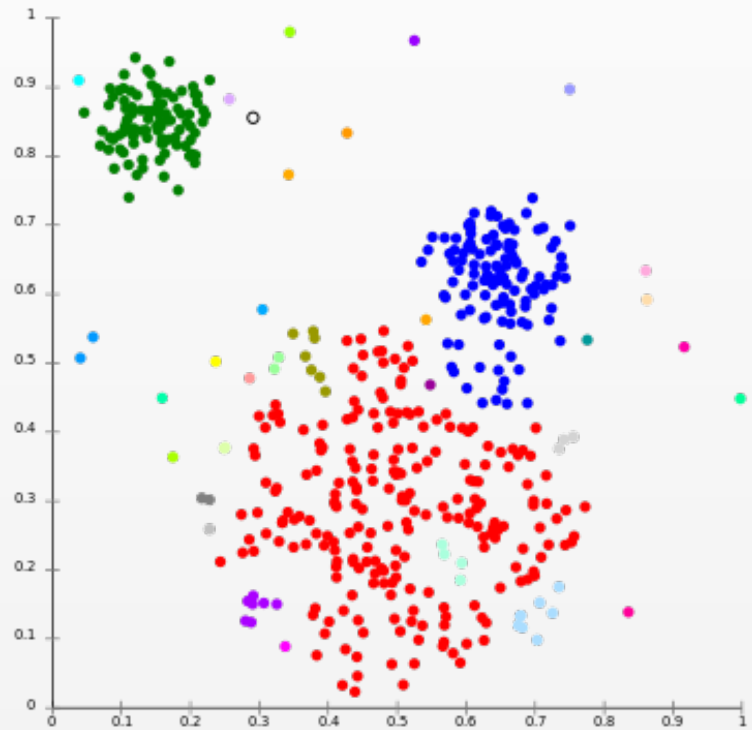


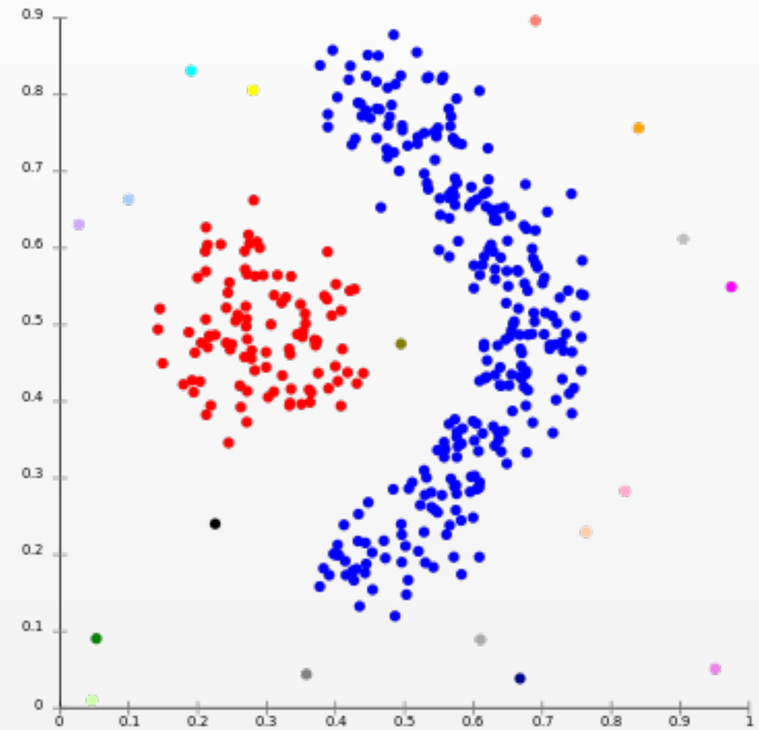


Notion of Clusters: Voronoi tesselation

Types of Clustering

Connectivity-based (Hierarchical)

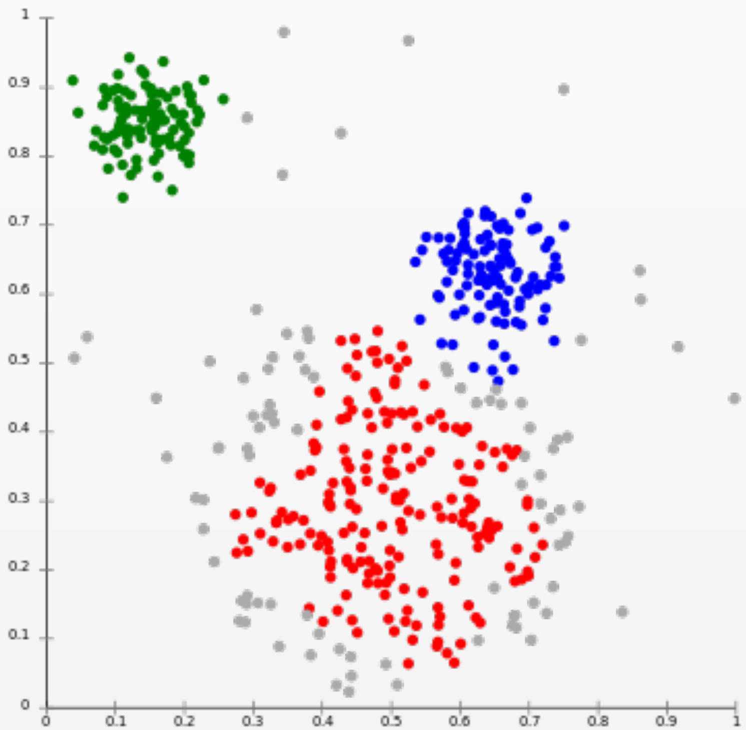


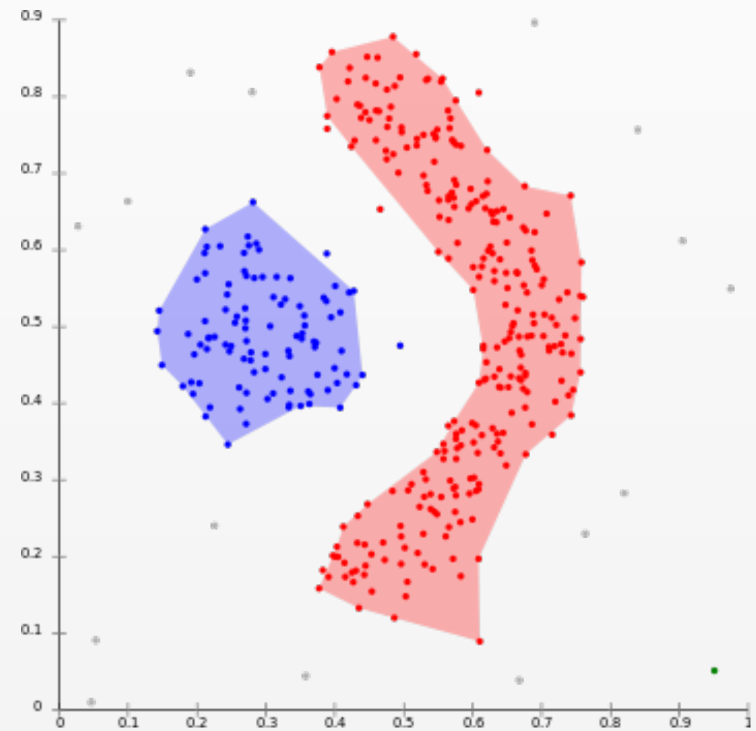


Notion of Clusters: Cut off dendrogram at some depth

Types of Clustering

Density-based (DBSCAN, OPTICS)

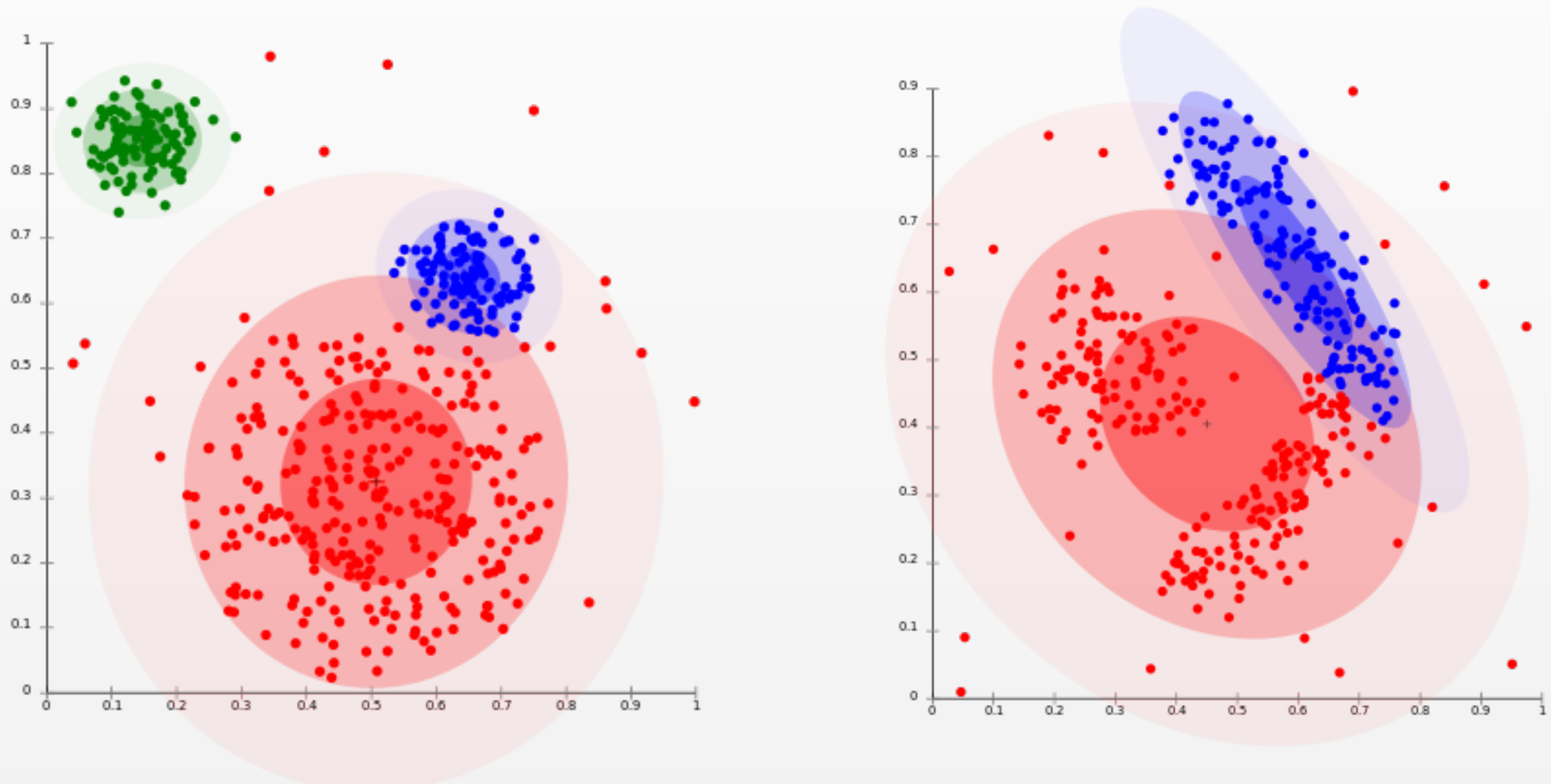




Notion of Clusters: Connected regions of high density

Types of Clustering

Distribution-based (Mixture Models)



Notion of Clusters: Distributions over features

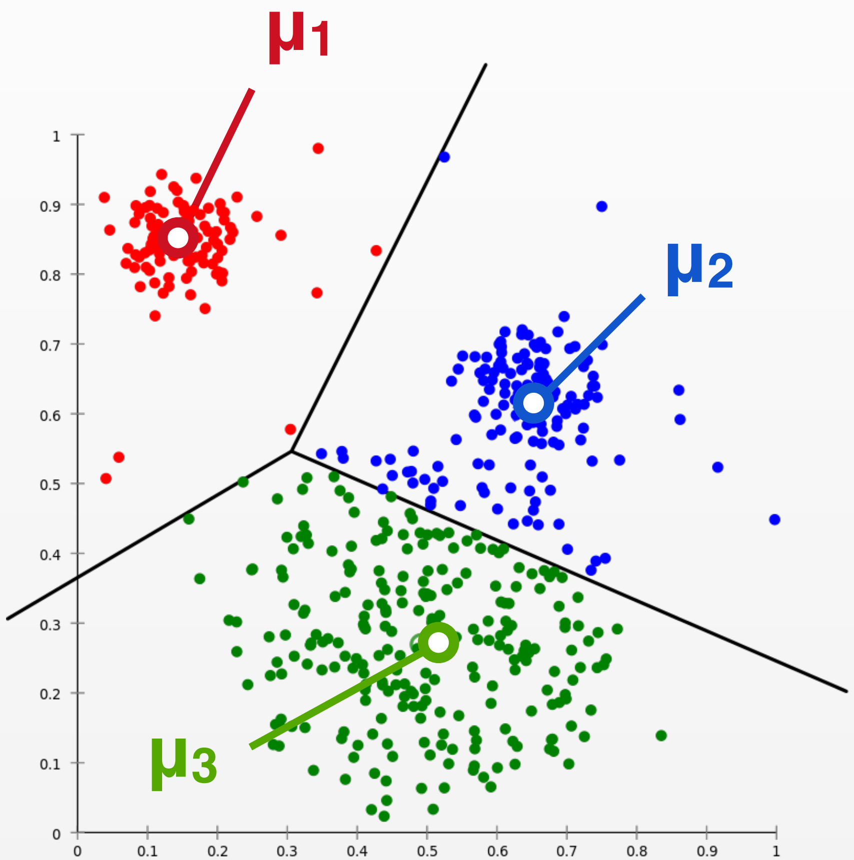
Clustering 1

Shantanu Jain

K-means Clustering

Algorithm and Objective

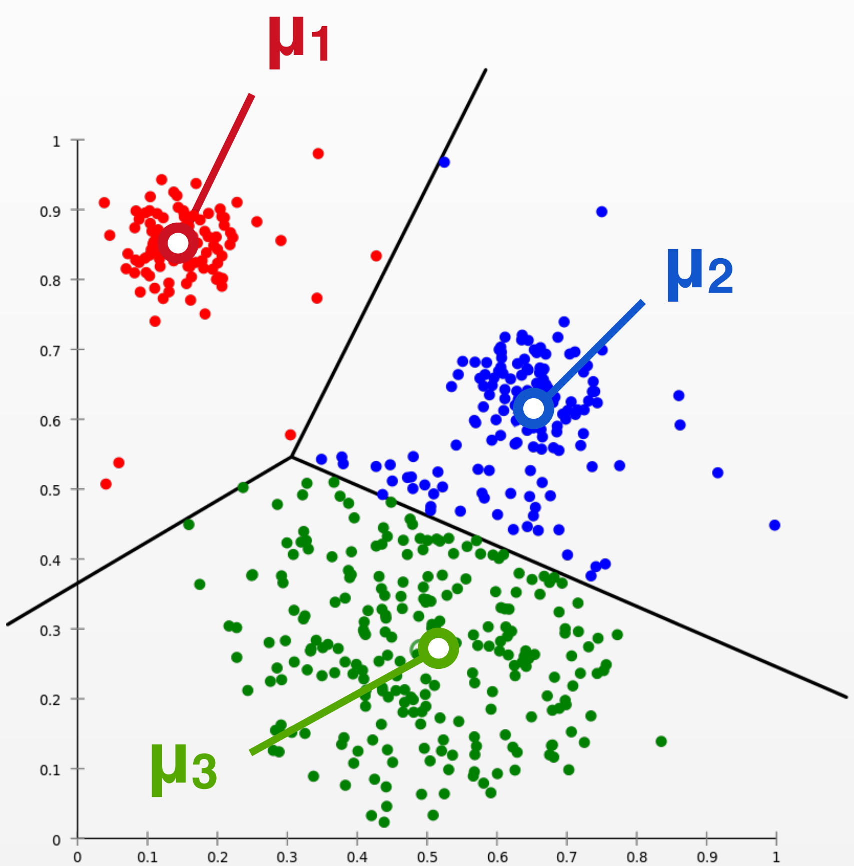
K-means Clustering



Idea: Find Clusters with Smallest Variance

- *Points:* $[\mathbf{x}_1, \ldots, \mathbf{x}_N]$, where each $\mathbf{x}_n \in \mathbb{R}^D$
- *Cluster assignments:* $[\mathbf{z}_1, \ldots, \mathbf{z}_N]$, where each $\mathbf{z}_n \in \{1, \ldots, K\}$
- *Cluster means:* $[\mu_1, \ldots, \mu_K]$, where each $\mu_k \in \mathbb{R}^D$
- ***Goal: find clusters with small variance*** **(all points near their means)**

K-means Clustering



K-means Algorithm

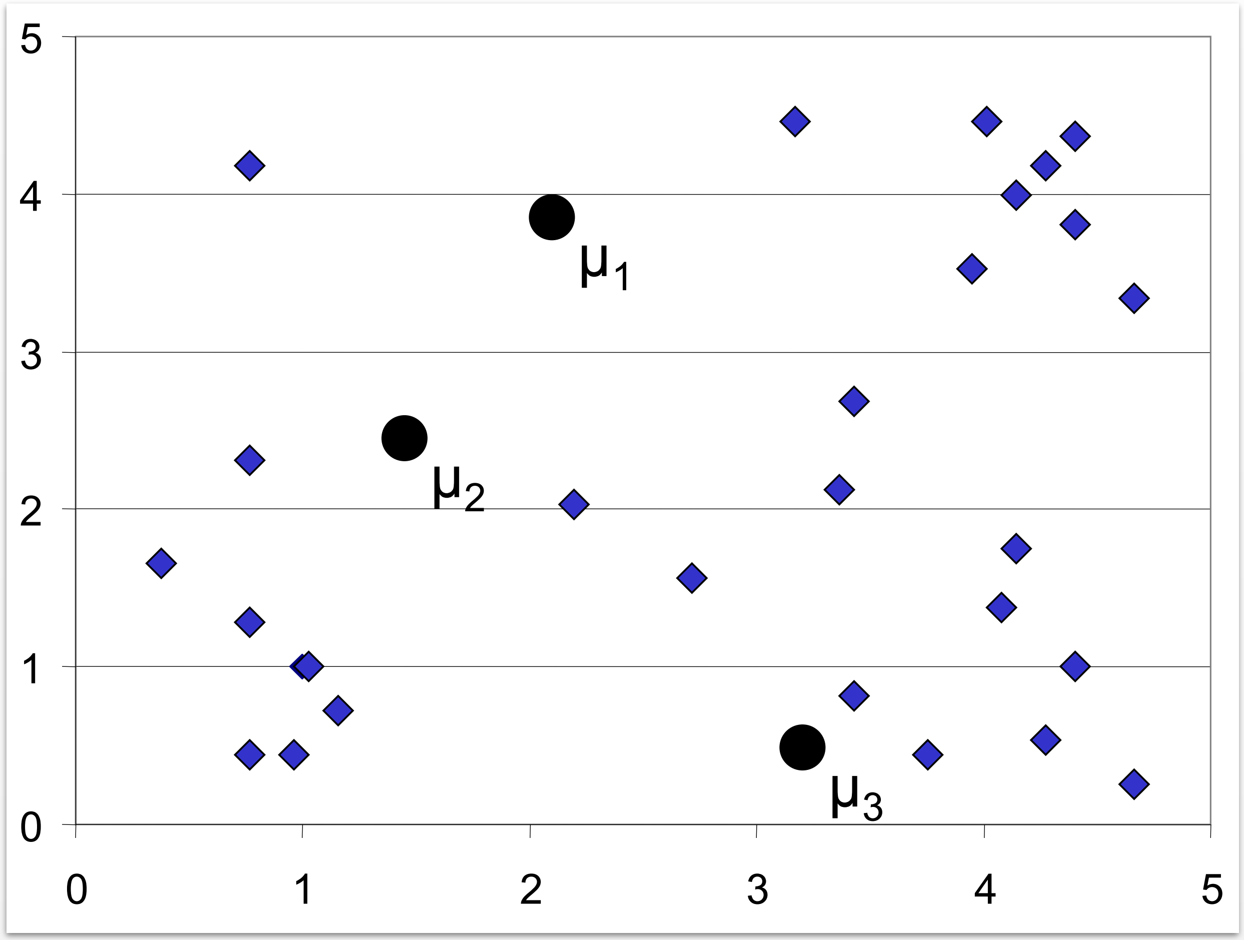
- *Randomly initialize means* $[\mu_1, \ldots, \mu_K]$
- Repeat until $[\mu_1, \ldots, \mu_K]$ unchanged
  - *Assign all points to* ***nearest*** *cluster*

    $$z_n = \operatorname*{argmin}_k \, ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2$$

  - *Update cluster means*

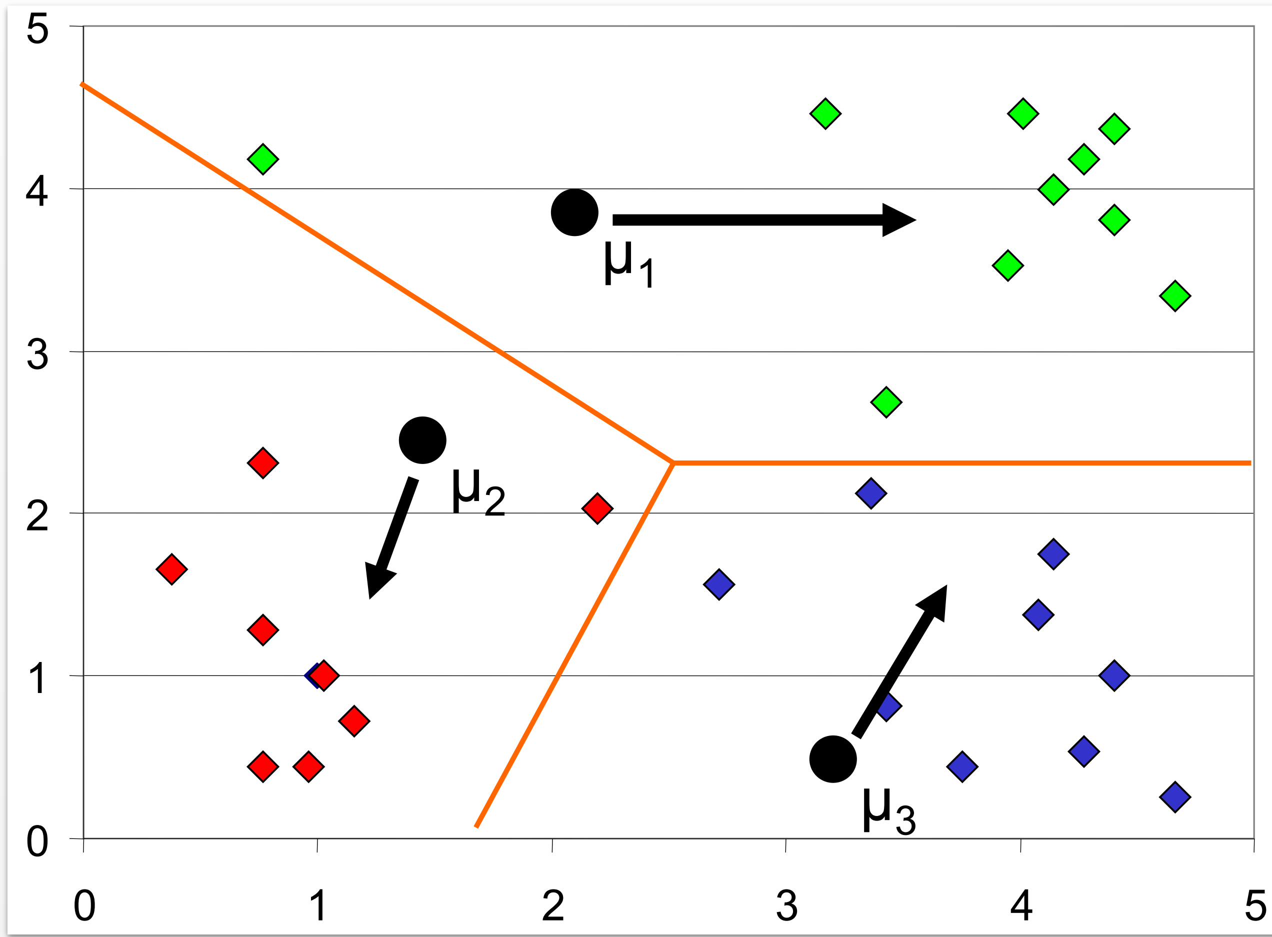
    $$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n:\, z_n = k} \boldsymbol{x}_n$$

K-means Clustering



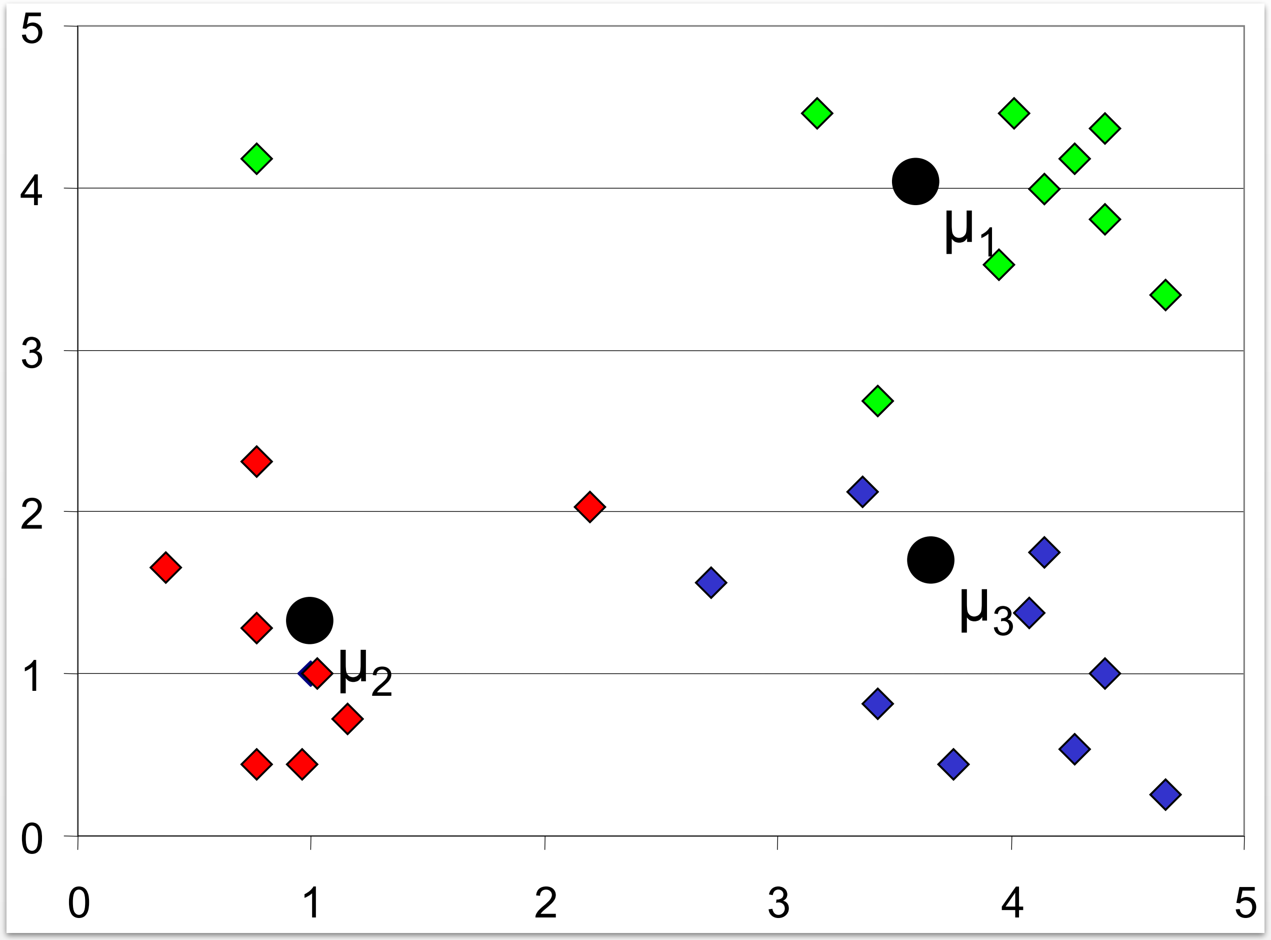
Randomly initialize *K* means μ_k

K-means Clustering



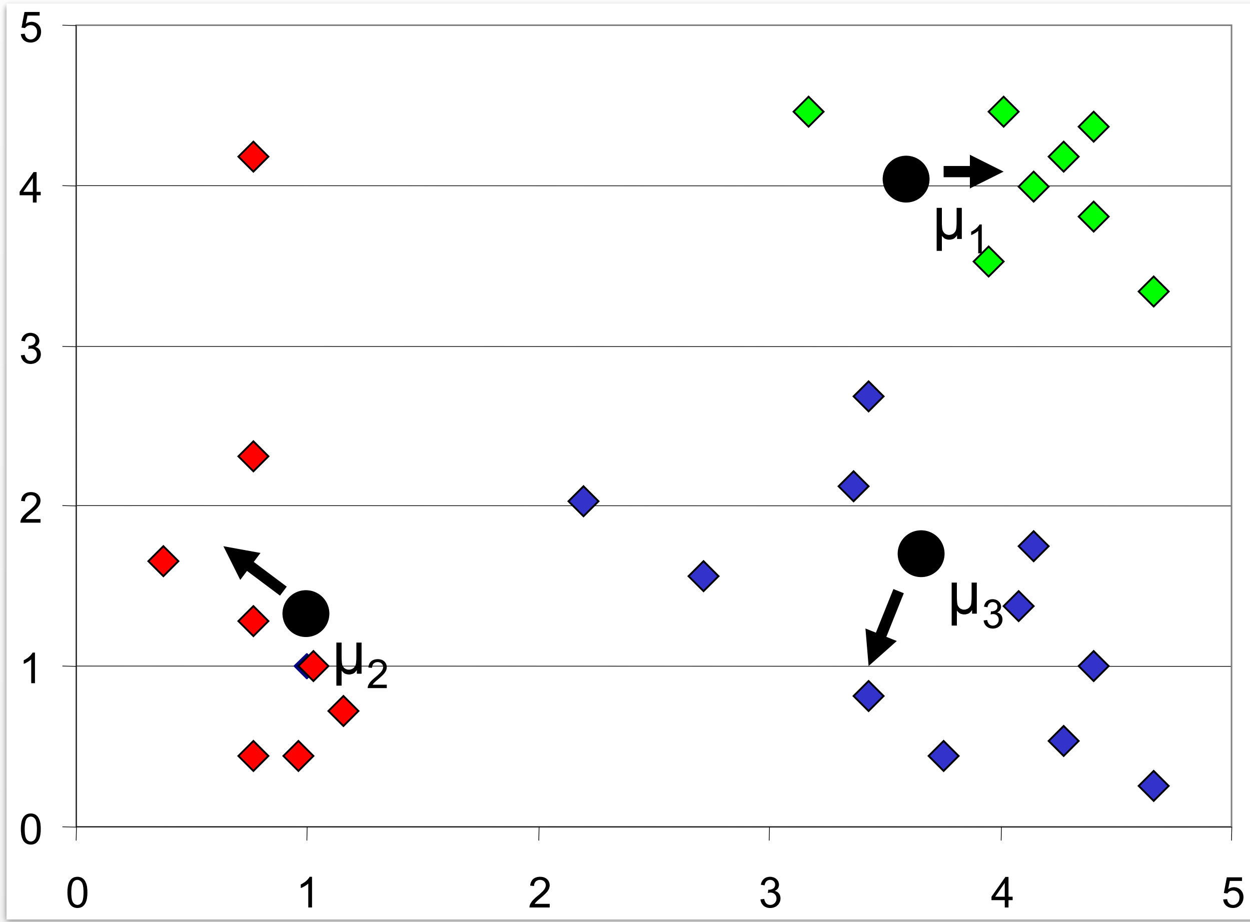
Assign each point to closest cluster,
then update means to average of points

K-means Clustering



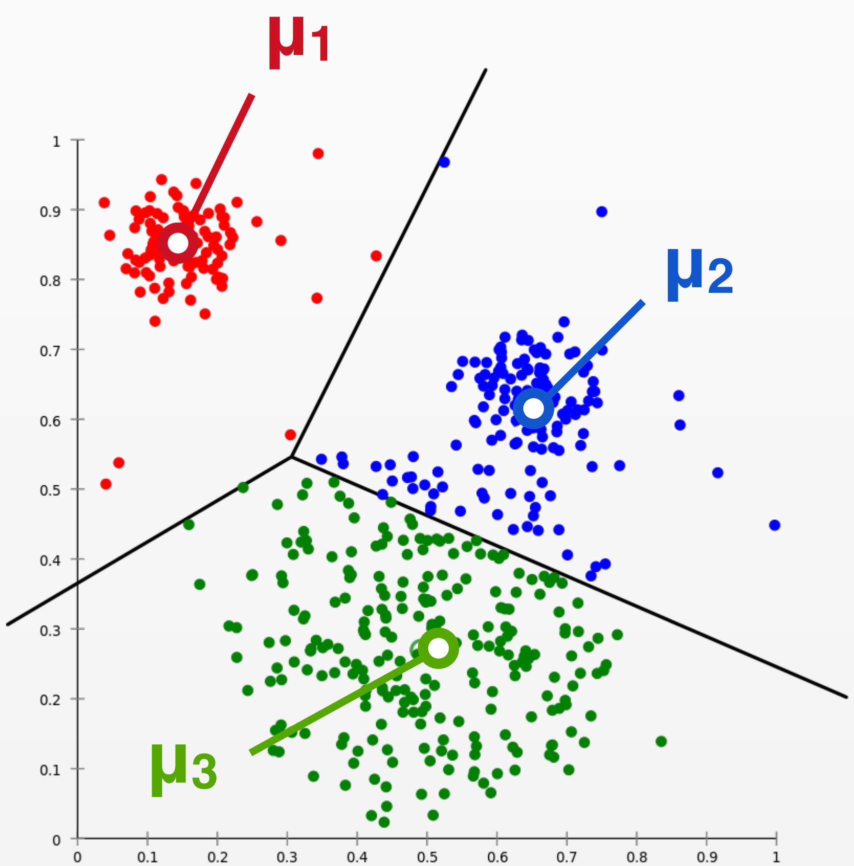
Assign each point to closest cluster,
then update means to average of points

K-means Clustering



Repeat until convergence
(no points reassigned, means unchanged)

K-means Objective



Loss: Variance of All Clusters Combined

$$L(z_1, \ldots, z_N) = \sum_{k=1}^{K} N_k \sigma_k^2$$

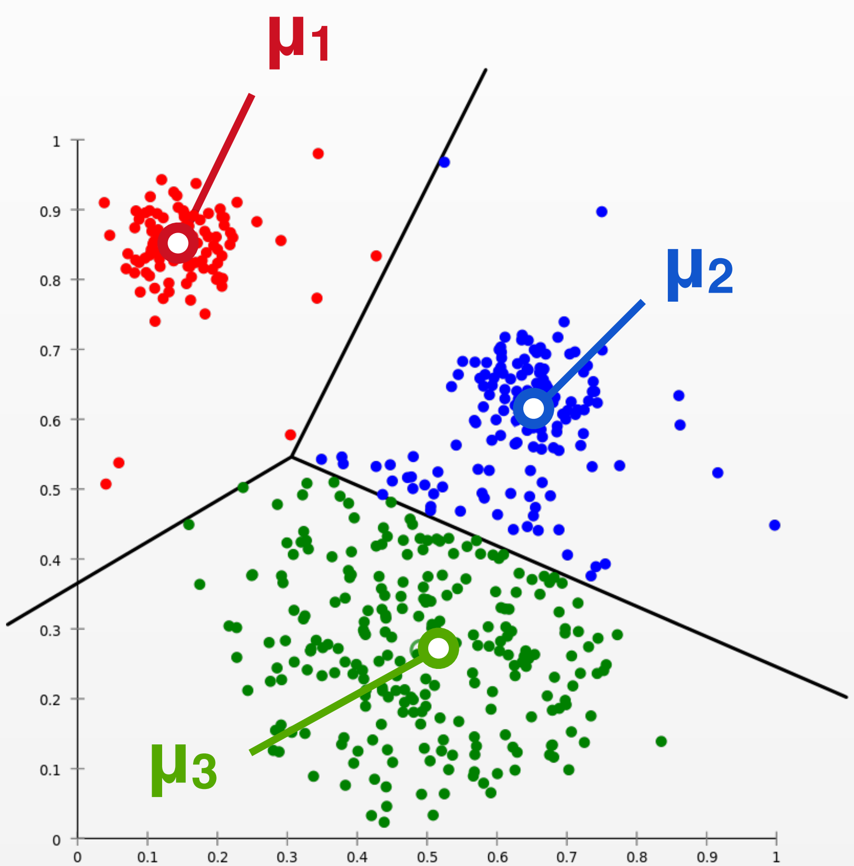
*Variance of cluster **k***

*Number of points in cluster **k***
(clusters with more points
contribute more to the loss)

Goal: Minimize Loss with Respect to Assignments

$$\min_{z_1,\ldots,z_N} L(z_1, \ldots, z_N)$$

Mean and Variance of a Cluster



Number of Points in a Cluster

$$N_k = \sum_{n=1}^{N} I[z_n = k] \qquad I[z_n = k] = \begin{cases} 1 & z_n = k, \\ 0 & z_n \neq k. \end{cases}$$

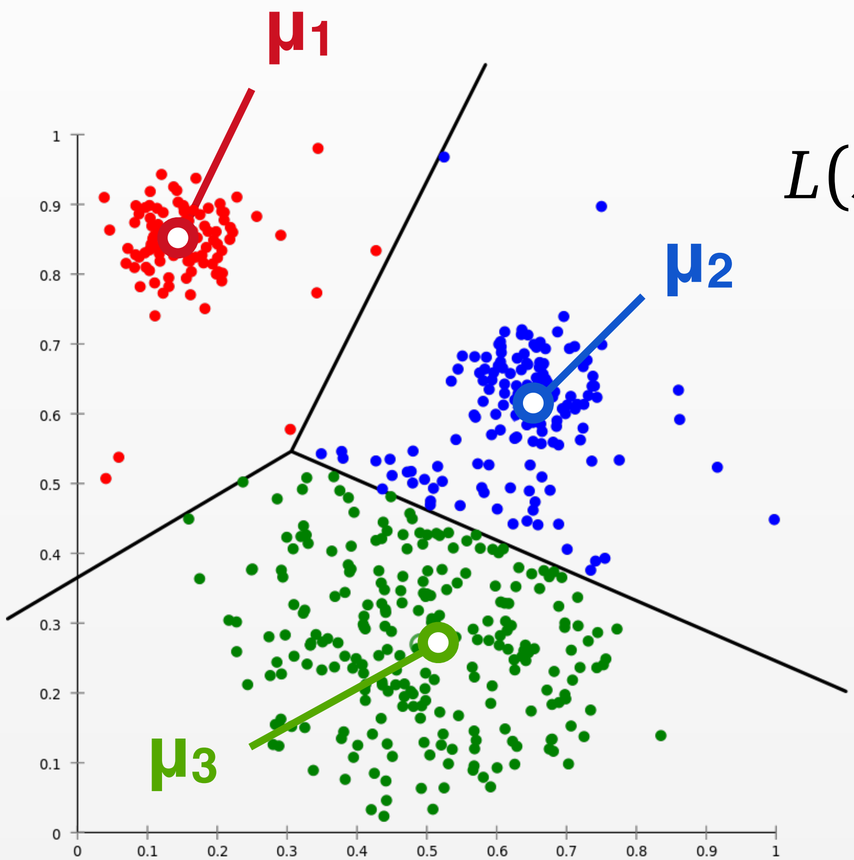
Mean of a Cluster

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} I[z_n = k]\, \boldsymbol{x}_n$$

Variance of a cluster

$$\sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} I[z_n = k]\ ||x_n - \mu_k||^2$$

K-means Objective



Loss: Variance of Clusters (given assignments)

$$L(z) = \sum_{k=1}^{K} N_k \sigma_k^2 \qquad \sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} I[z_n = k] \; ||x_n - \mu_k|$$

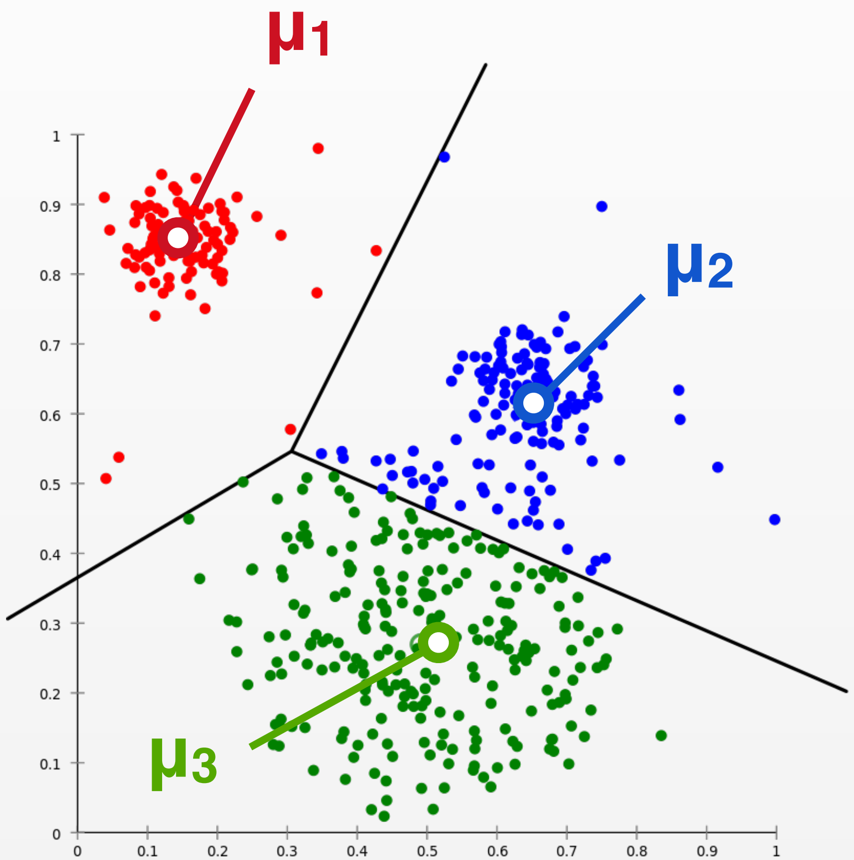
$$= \sum_{k=1}^{K} \sum_{n=1}^{N} I[z_n = k] \; ||x_n - \mu_k||^2 \qquad \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} I[z_n = k] \, \boldsymbol{x}$$

Goal: Minimize Loss with Respect to Assignments

$$\min_{z_1,\ldots,z_N} \; L(z_1, \ldots, z_N)$$

K^N possible combinations; can't solve via brute force

K-means Iteration



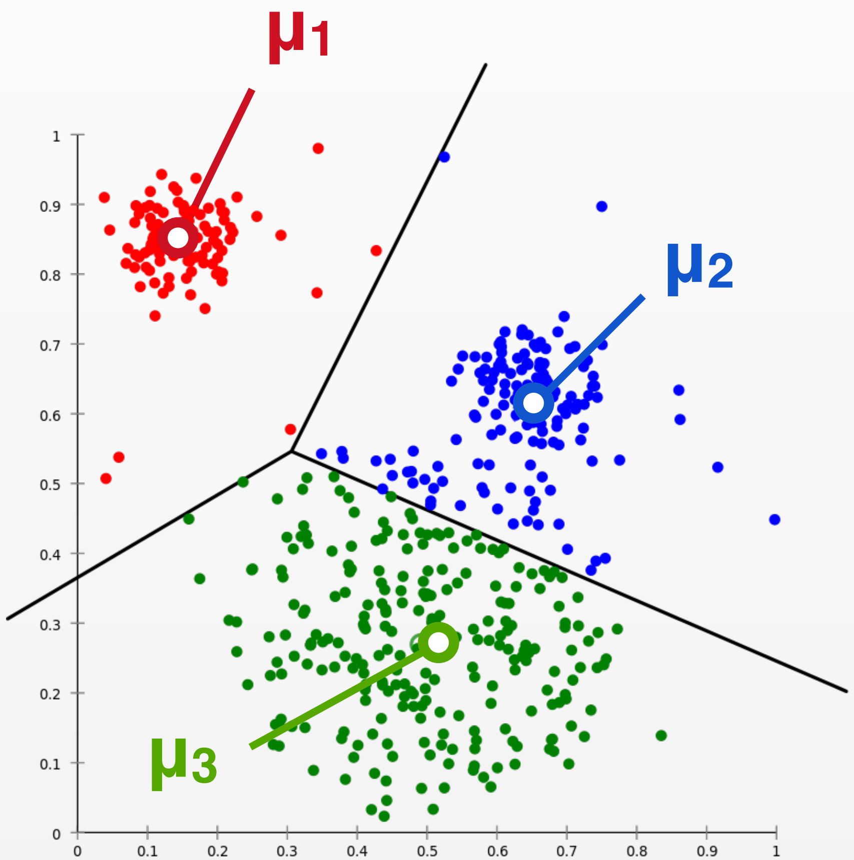
Solution: Define Loss in terms of μ and z

$$L(\mu, z) = \sum_{k=1}^{K} \sum_{n=1}^{N} I[z_n = k] \; ||x_n - \mu_k||^2$$

K-means Algorithm

- *Randomly initialize μ*
- Repeat until L(μ, z) does not improve
  1. Minimize **L(μ, z)** with respect to *z*
     (assign points to closest cluster)
  2. Minimize **L(μ, z)** with respect to *μ*
     (place clusters close to points)

K-means Clustering



K-means Algorithm

- *Randomly initialize means* [$\mu_1, \ldots, \mu_K$]
- Repeat until $L(\mu, z)$ unchanged
  - *Assign all points to* ***nearest*** *cluster*

    $$z_n = \underset{k}{\operatorname{argmin}} \, ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2 = \underset{z_n}{\operatorname{argmin}} \, L(z, \mu)$$

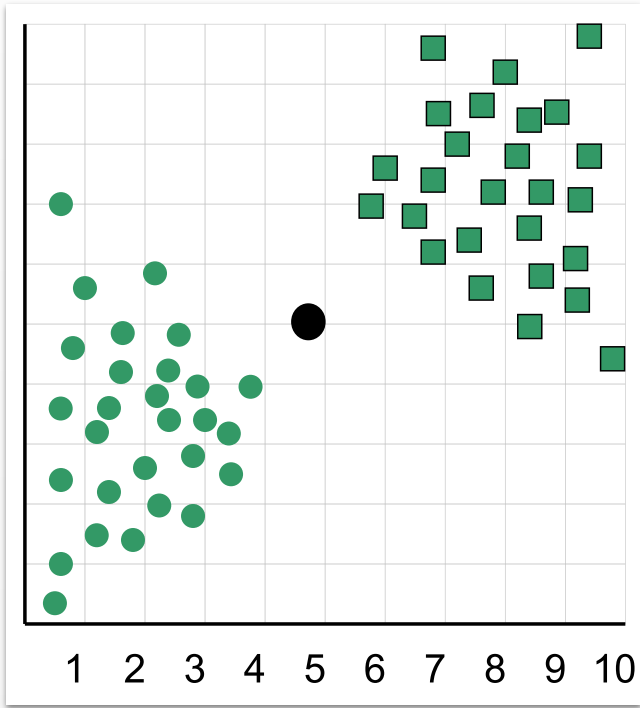
  - *Update cluster means*

    $$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n: z_n = k} \boldsymbol{x}_n = \underset{\mu_k}{\operatorname{argmin}} \, L(z, \mu)$$

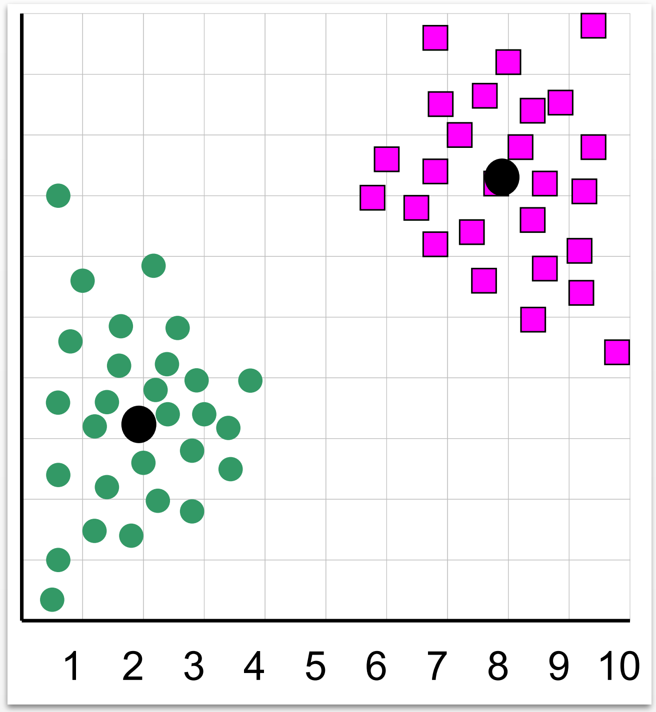
Each iteration reduces loss until (local) optimum is found

Choosing K

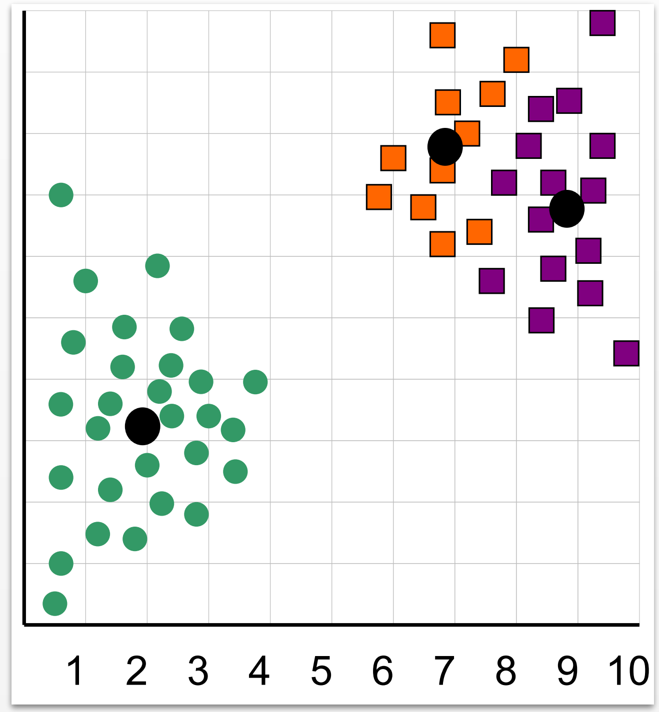
K=1, L=873



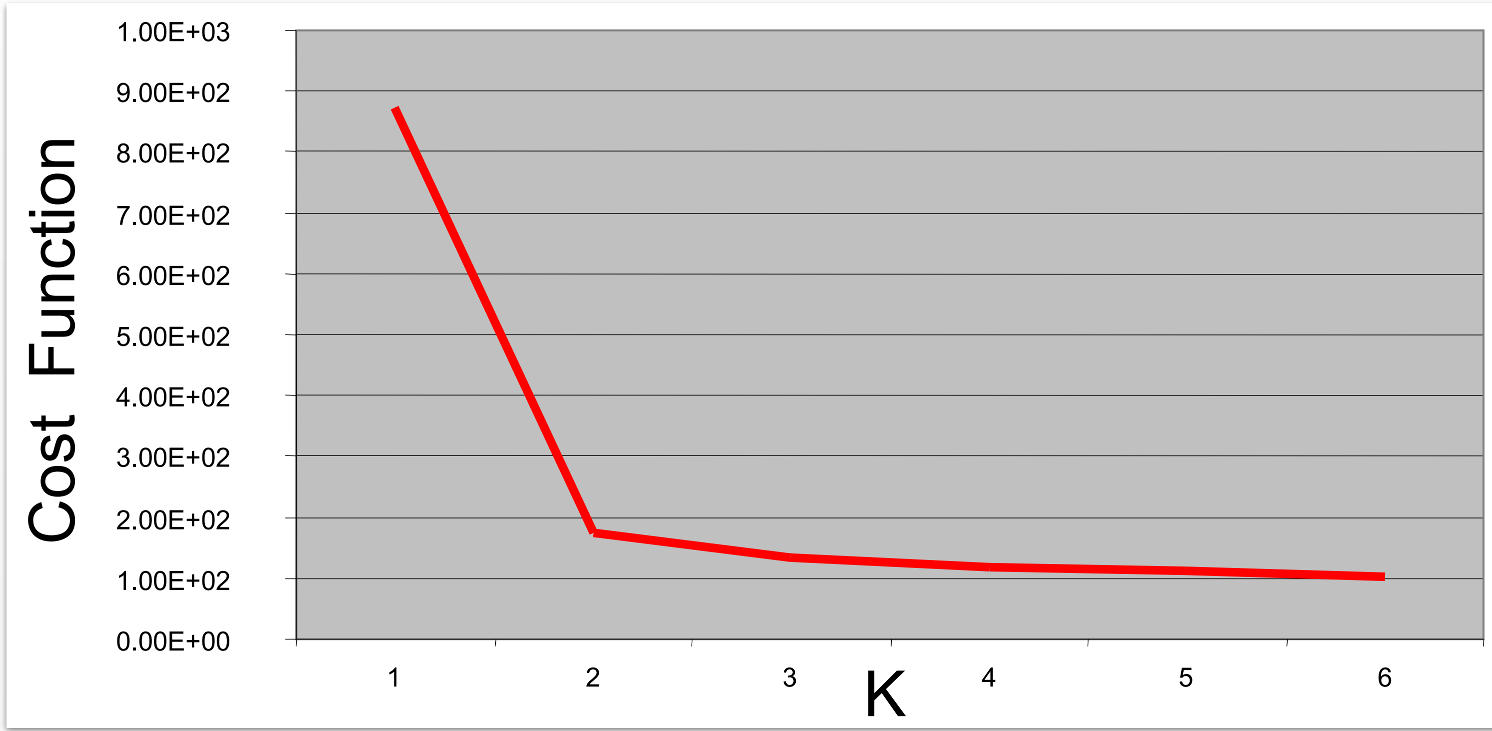
K=2, L=173



K=3, L=134



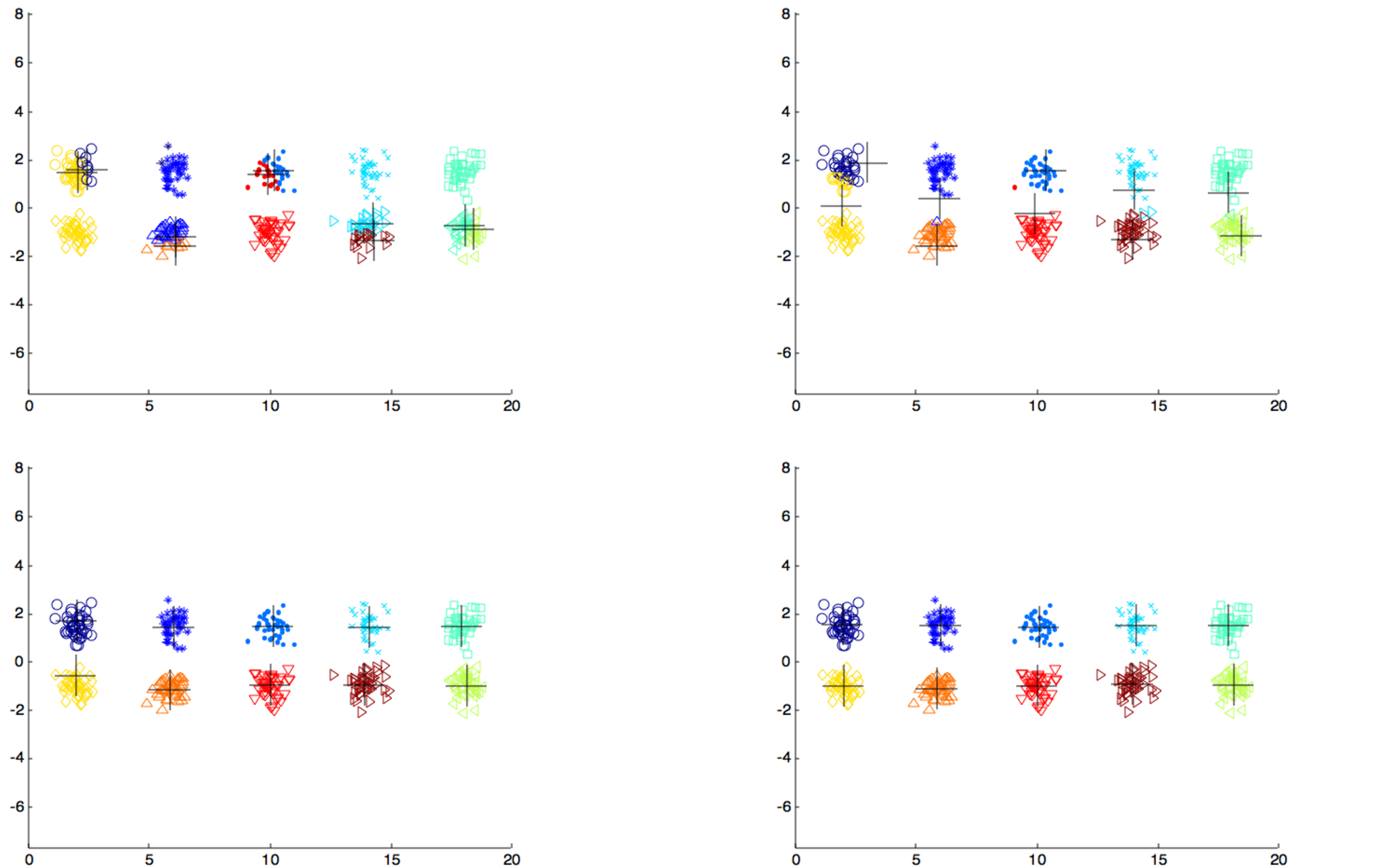
Choosing K



“Elbow finding” (a.k.a. “knee finding”)

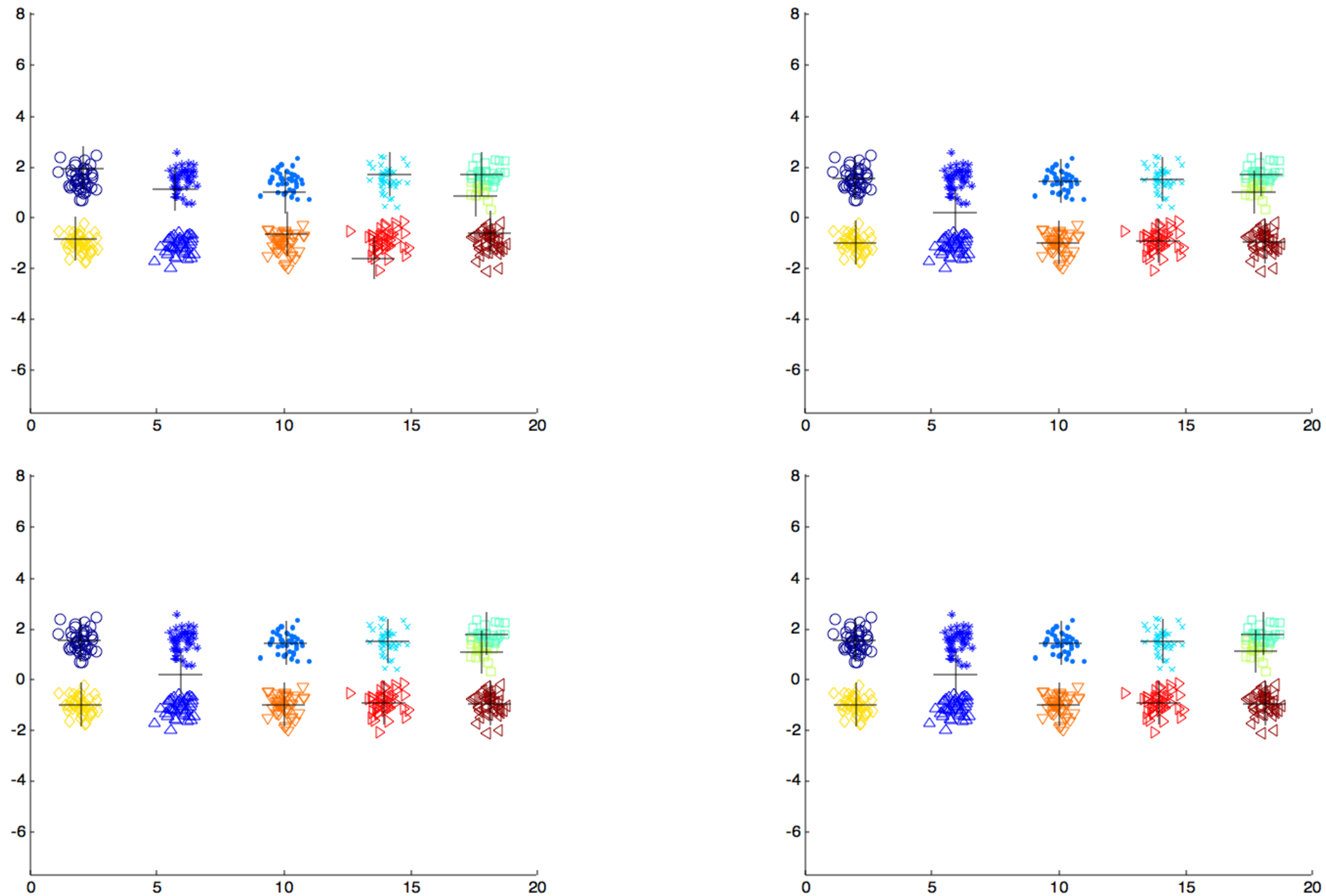
Set K to value just above “abrupt” increase

Sensitivity to initial centroids



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

Sensitivity to initial centroids



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

Picking the initialization cluster centers: a significant issue

cluster assignments
turned by K-means,
local minimizer of
e loss

pt: the global
inimizer of the loss

- It is the speed and simplicity of the k-means method that make it appealing, not its accuracy. Indeed, there are many natural examples for which the algorithm generates arbitrarily bad clustering (i.e., $L(\hat{z})/L(z_{opt})$ is unbounded even when N and K are fixed). This does not rely on an adversarial placement of the starting centers, and in particular, it can hold with high probability if the centers are chosen uniformly at random from the data points.

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Postprocessing
- Bisecting K-means
  - Not as susceptible to initialization issues

Furthest first

- Pick first center to be the mean of the data

$$M_1 \leftarrow \{\mu_1\}$$

- For the subsequent centers iteratively pick the point whose distance to the closest center is largest.

$$\mu_{j+1} \leftarrow \text{argmax}_{x \in X}[D_{min}(x, M_j)]$$

$$M_{j+1} \rightarrow M_j \cup \{\mu_{j+1}\}$$

$D_{min}(x, M_j)$ distance of x to the closest center in M_j.

Problem: Outliers get chosen as centers.

M_j is the set of centroids at j^{th} step.

K-Means ++

1. Pick first center uniformly at random

$$M_1 \leftarrow \{\mu_1\}$$

2. For the subsequent centers iteratively pick a point $x \in X$ randomly with probability proportional to $D_{min}(x, M_j)$

$$\mu_j \leftarrow x \sim p(x) = \frac{D_{min}(x, M_j)}{\sum_{x \in X} D_{min}(x, M_j)}$$

$$M_{j+1} \rightarrow M_j \cup \{\mu_{j+1}\}$$

$D_{min}(x, M_j)$ distance of x to the closest center in M_j.

M_j is the set of centroids at j^{th} step.

Here the outliers still have a high probability of being selected compared to other points individually. However, the cumulative probability of points having moderately large distances lying in a dense region dominate the probability as a group.

Theoretical guarantees when using K-Means++

$$\mathbf{E}[L(\hat{z})] \leq (8 \log K + 2) L(z_{opt})$$

Arthur, David, and Sergei Vassilvitskii. *k-means++: The advantages of careful seeding*. Stanford, 2006.

k	Average ϕ		Minimum ϕ		Average T	
	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	135512	126433	119201	111611	0.14	0.13
25	48050.5	15.8313	25734.6	15.8313	1.69	0.26
50	5466.02	14.76	14.79	14.73	3.79	4.21

Table 2: Experimental results on the *Norm-25* dataset (n = 10000, d = 15)

k	Average ϕ		Minimum ϕ		Average T	
	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	7553.5	6151.2	6139.45	5631.99	0.12	0.05
25	3626.1	2064.9	2568.2	1988.76	0.19	0.09
50	2004.2	1133.7	1344	1088	0.27	0.17

Table 3: Experimental results on the *Cloud* dataset (n = 1024, d = 10)

k	Average ϕ		Minimum ϕ		Average T	
	k-means	k-means++	k-means	k-means++	k-means	k-means++
10	$3.45 \cdot 10^8$	$2.31 \cdot 10^7$	$3.25 \cdot 10^8$	$1.79 \cdot 10^7$	107.5	64.04
25	$3.15 \cdot 10^8$	$2.53 \cdot 10^6$	$3.1 \cdot 10^8$	$2.06 \cdot 10^6$	421.5	313.65
50	$3.08 \cdot 10^8$	$4.67 \cdot 10^5$	$3.08 \cdot 10^8$	$3.98 \cdot 10^5$	766.2	282.9

Table 4: Experimental results on the *Intrusion* dataset (n = 494019, d = 35)

K-means Complexity

Loss function

$$L = \sum_{k=1}^{K} \sum_{n=1}^{N} I[z_n = k] (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^2$$

O(KND) computational complexity (per iteration) for *K* clusters, *N* points, and *D* features.