

CS 189/289

Today's lecture outline

Clustering (k-means, mixture of Gaussians)

Assigned Reading:

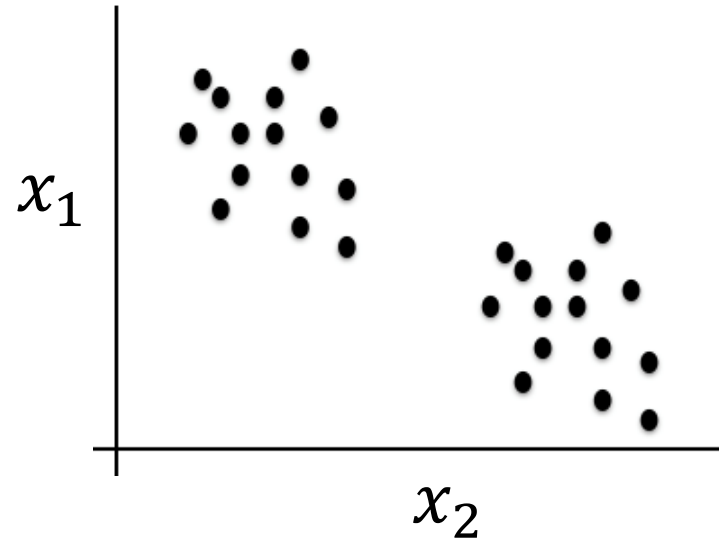
- 15.1 (K-means clustering, up to, not including 15.1.1)
- 15.2 (Mixture of Gaussians)

Recall, Unsupervised learning

- Seen *supervised learning*, $\{(x_i, y_i)\}$ for $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ or $y \in \mathbb{Z}$.
- Much ML is focused on modeling $\{x_i\}$, *unsupervised learning*, which includes:
 - i. Dimensionality reduction, $z \in \mathbb{R}^m = f_\theta(x)$, $m \ll d$.
 - ii. Clustering, $z \in \mathbb{Z} = f_\theta(x)$.
 - iii. Representation learning, $z \in \mathbb{R}^m$, $z = f_\theta(x)$, or $z \sim p_\theta(x)$.
 - iv. Density estimation, evaluate $p_\theta(x)$.
 - v. "Generative" modeling, $x \sim p_\theta(x)$

The main idea of *clustering* $\{x_i\}$

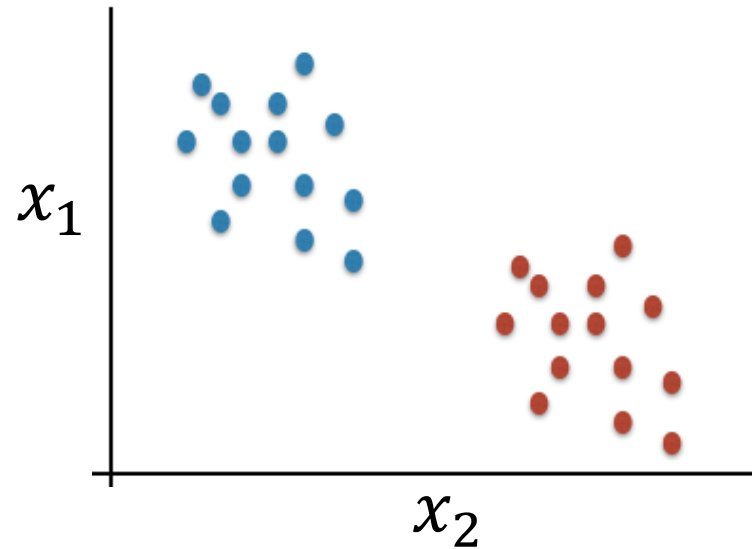
Suppose we had only input features, and no class labels:



We may want to infer/assign discrete “class labels” from the data, based on the structure in the input space.

The main idea of *clustering* $\{x_i\}$

Suppose we had only input features, and no class labels:

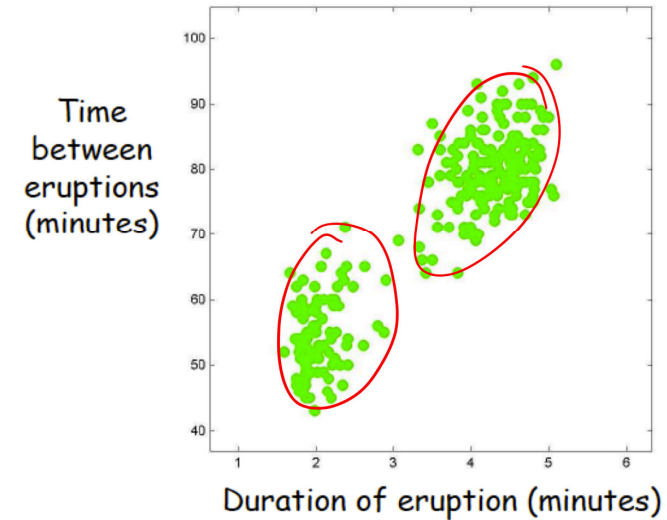


We may want to *infer/assign* discrete “class labels” from the data, based on the structure in the input space.

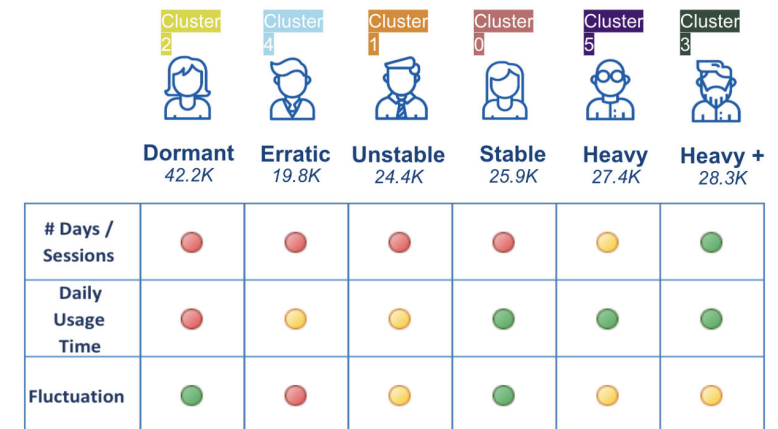
Clustering for data exploration

e.g. find hidden subgroups:

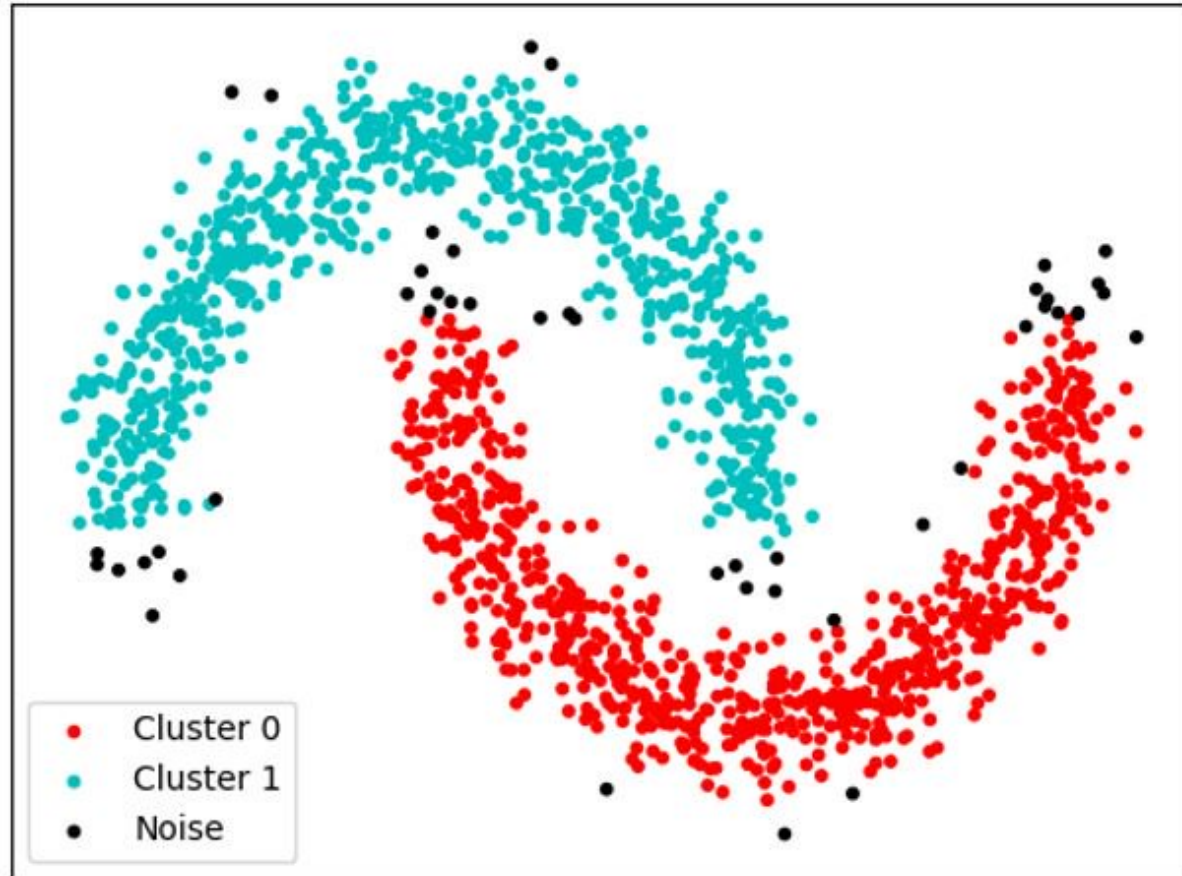
- Types of customers in a database from customer activities.
- Subtypes of disease for therapeutics.
- Types of cells in a tissue from single cell data.
- Ancestry groups from genetic data.
- Finding topics in on-line documents.
- etc.



Cluster Interpretation and Labeling



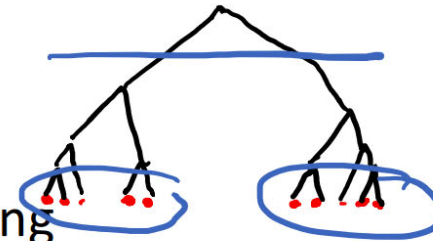
Clustering for outlier detection



Three broad approaches to clustering

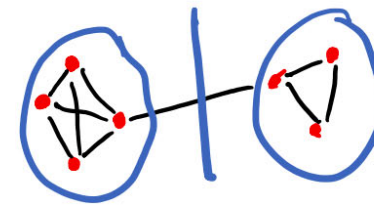
- Hierarchical clustering

- Build a tree (bottom-up or top-down), representing distances among data points
- **Example:** single-, average- linkage clustering



- Partitional approaches

- Define and optimize a notion of “cost” defined over partitions
- **Example:** Spectral clustering, graph-cut based approaches

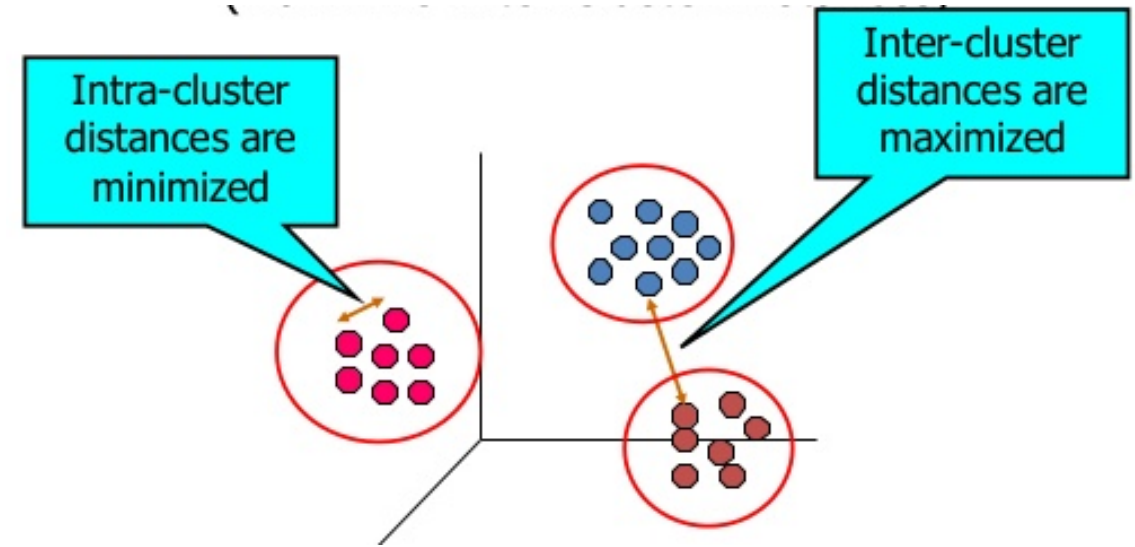


- Model-based approaches

- Maintain cluster “models” and infer cluster membership (e.g., assign each point to closest center)
- **Example:** k-means, Gaussian mixture models, ...

Main desiderata of clustering

1. Want high intra-cluster similarity.
2. Want low inter-cluster similarity.



3. Similarity/distance is in the eye of the beholder!

Aside: distances, metrics and similarities.

- “want points to be similar/dissimilar”
- “want distance to be minimized/maximized”.

Properties of a distance function (metric):

Aside: distances, metrics and similarities.

- “want points to be similar/dissimilar”
- “want distance to be minimized/maximized”.

Properties of a distance function (metric):

1. $j = k$ iff $d(j, k) = 0$.
2. $j \neq k$ iff $d(j, k) > 0$.
3. symmetry, $d(j, k) = d(k, j)$ (why KL-divergence is not a distance)
4. triangle inequality, $d(i, j) + d(i, k) \geq d(j, k)$

Aside: distances, metrics and similarities.

- “want points to be **similar/dissimilar**”
- “want **distance** to be minimized/maximized”.

Properties of a **distance** function (metric):

1. $j = k$ iff $d(j, k) = 0$.
2. $j \neq k$ iff $d(j, k) > 0$.
3. *symmetry, $d(j, k) = d(k, j)$ (why KL is not a distance)*
4. *triangle inequality, $d(i, j) + d(i, k) \geq d(j, k)$*

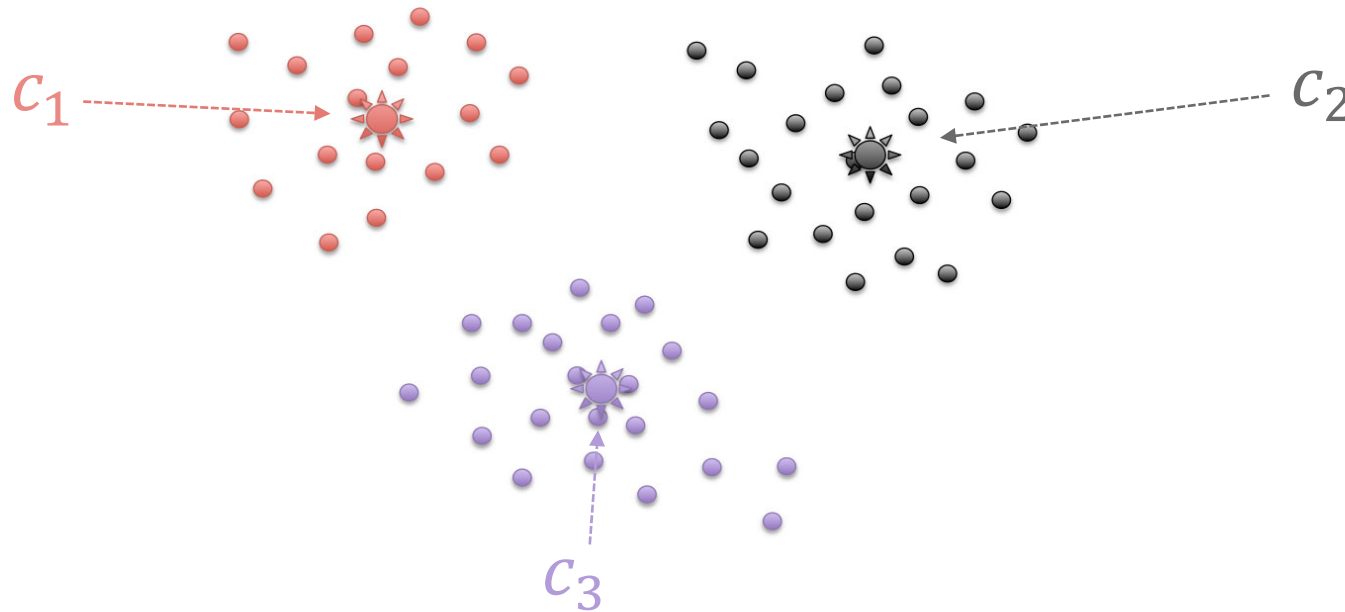
dissimilarity: intuitively related, but may not satisfy metric properties.

similarity: “complement” of dissimilarity, e.g.:

$$\text{similarity}(j, k) = 1 - \text{dissimilarity}(j, k)$$

Centroid-based clustering

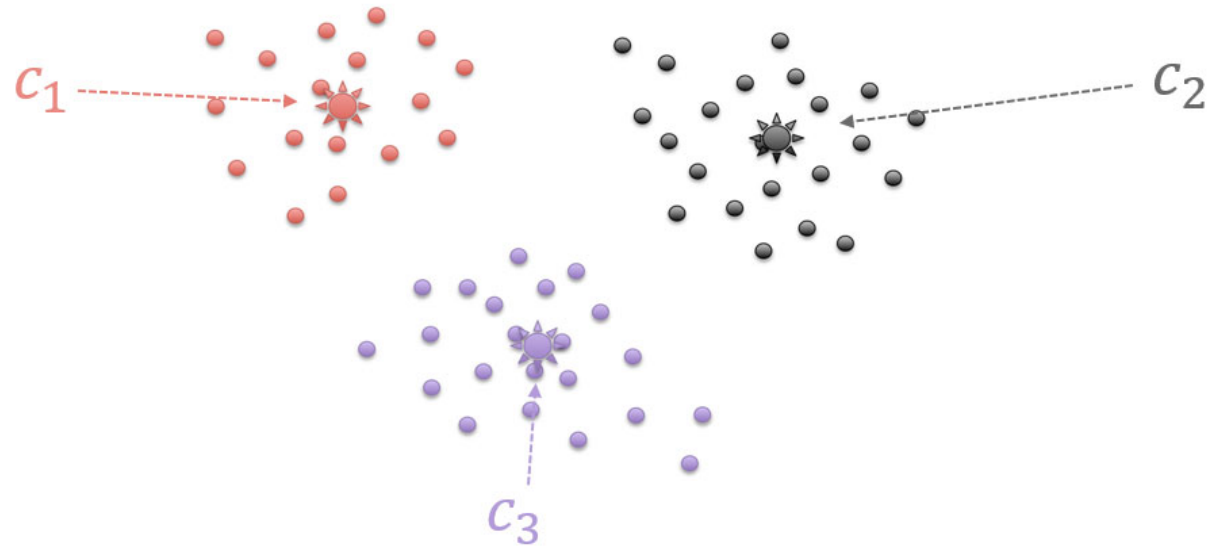
- Each cluster is represented by a point in the input space-- a centroid--though not necessarily in the training data), $c_k \in R^d$ (for $X \in R^d$).



- "K-means" is the most common centroid-based approach.

K-means clustering

- Parameters are $\{c_k\}$.
- Chosen such that:
 - the distance of each point, x_i , to its assigned centroid, is minimized.

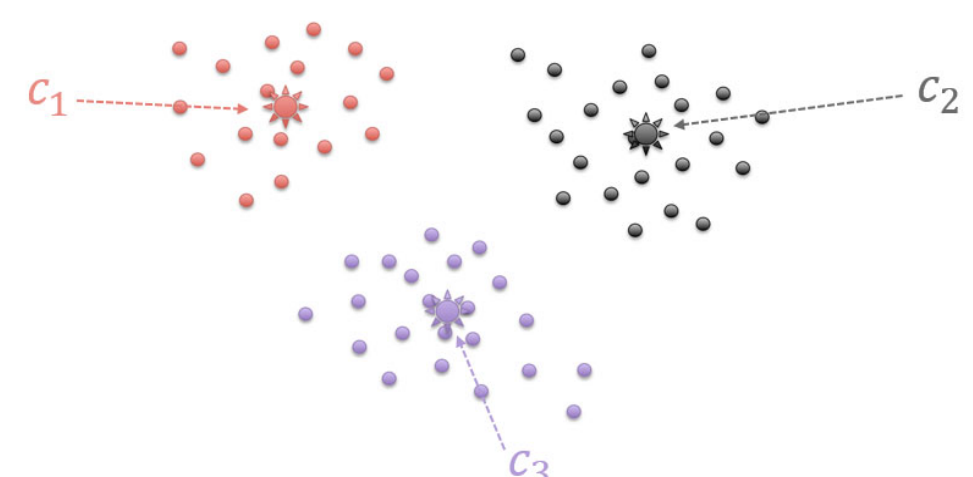


Formally: K-means clustering

- Training data, $X = \{x_i\}_{i=1}^n, x_i \in R^d$.
- Parameters are $\{c_k \in R^d\}$.
- A cluster partition, $C_1 \cup C_2 \cup \dots \cup C_K$, wherein every x_i is assigned to one (and only one) of the K clusters.
- Optimization problem:

$$\underset{S=C_1 \cup \dots \cup C_K, \{c_1, \dots, c_K\}}{\operatorname{argmin}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$

cluster partition cluster centroids



The diagram shows three clusters of data points in a 2D space. The top-left cluster consists of red points with a red star-shaped centroid labeled c_1 . The top-right cluster consists of black points with a black star-shaped centroid labeled c_2 . The bottom cluster consists of purple points with a purple star-shaped centroid labeled c_3 . Dashed lines connect the labels c_1 , c_2 , and c_3 to their respective centroids.

Parameter learning in K-means

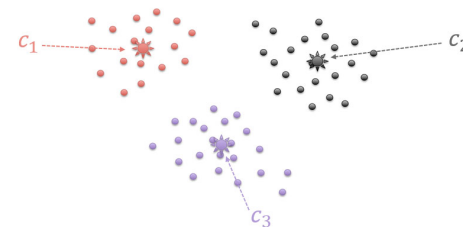
- Suppose we knew $C_1 \cup C_2 \cup \dots \cup C_K$ how could we find $\{c_k\}$?
- The optimization problem would reduce to:

$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

$$\left[\begin{aligned} & \sum \frac{\partial}{\partial c_k} (x - c_k)^T (x - c_k) \\ &= -2 \sum (x - c_k) \\ &= 0 \Rightarrow \sum x = \sum c_k \end{aligned} \right]$$

- For which the answer is $\hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$

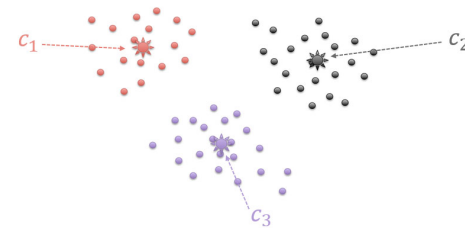
$$\operatorname{argmin}_{S=C_1 \cup \dots \cup C_K, \{c_1, \dots, c_K\}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$



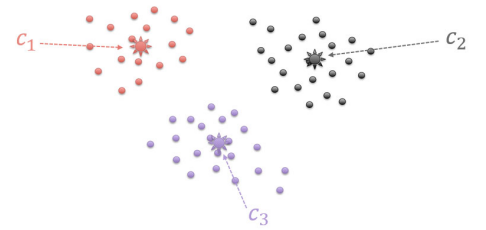
Other way around (parameter learning)

- Suppose we knew $\{c_k\}$, how could we find $C_1 \cup C_2 \cup \dots \cup C_K$?
- Answer: choose the cluster which is closest to each point, $z_i \equiv \operatorname{argmin}_k \|x_i - c_k\|^2$, and then $\hat{C}_k = \{x_i | z_i = k\}$.

$$\operatorname{argmin}_{S=C_1 \cup \dots \cup C_K, \{c_1, \dots, c_K\}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$



The K-Means algorithm ("Lloyd's Algorithm")



1. Initialize the cluster centers, $\{c_k\}$

(e.g., pick k points at random from your training data).

$$\operatorname{argmin}_{S=C_1 \cup \dots \cup C_K, \{c_1, \dots, c_K\}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$

2. Repeat until convergence:

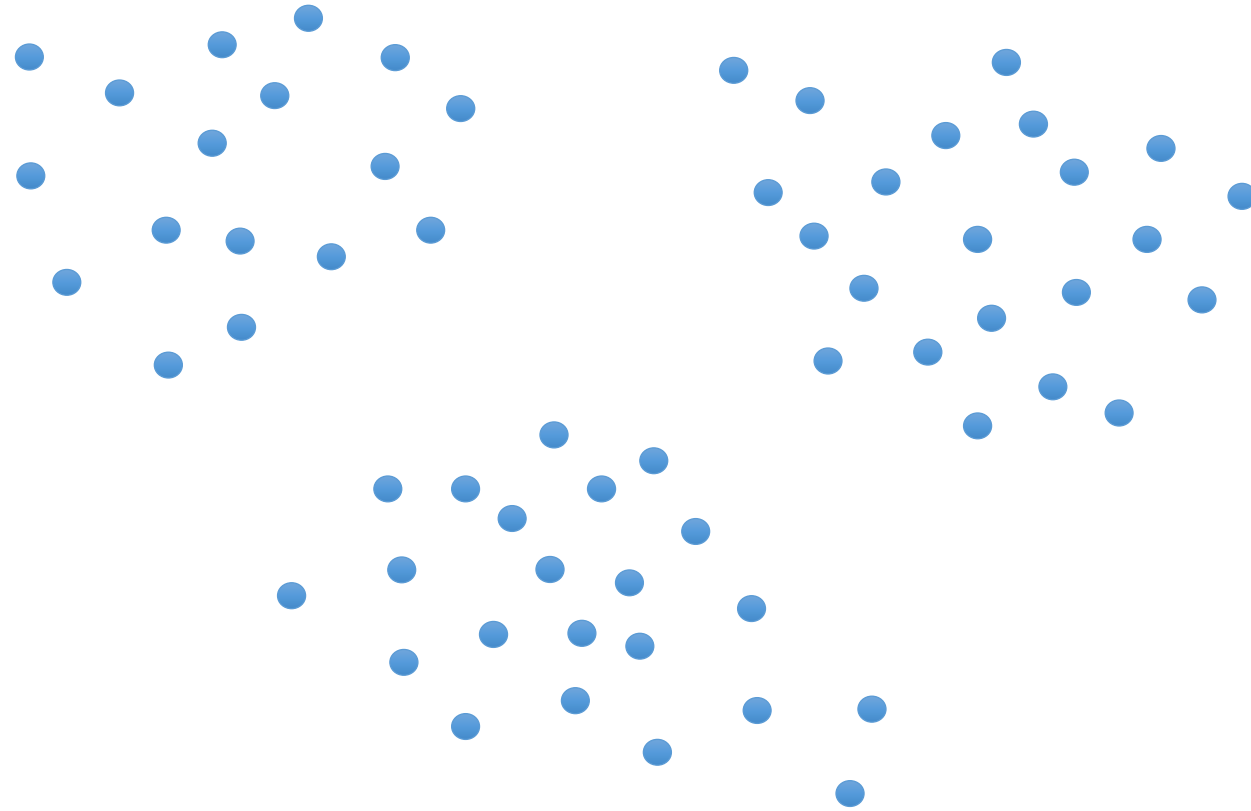
- Compute partition $C_1 \cup C_2 \cup \dots \cup C_K$, given the $\{c_k\}$.
- Compute centers $\{c_k\}$, given $C_1 \cup C_2 \cup \dots \cup C_K$.

Does this converge?

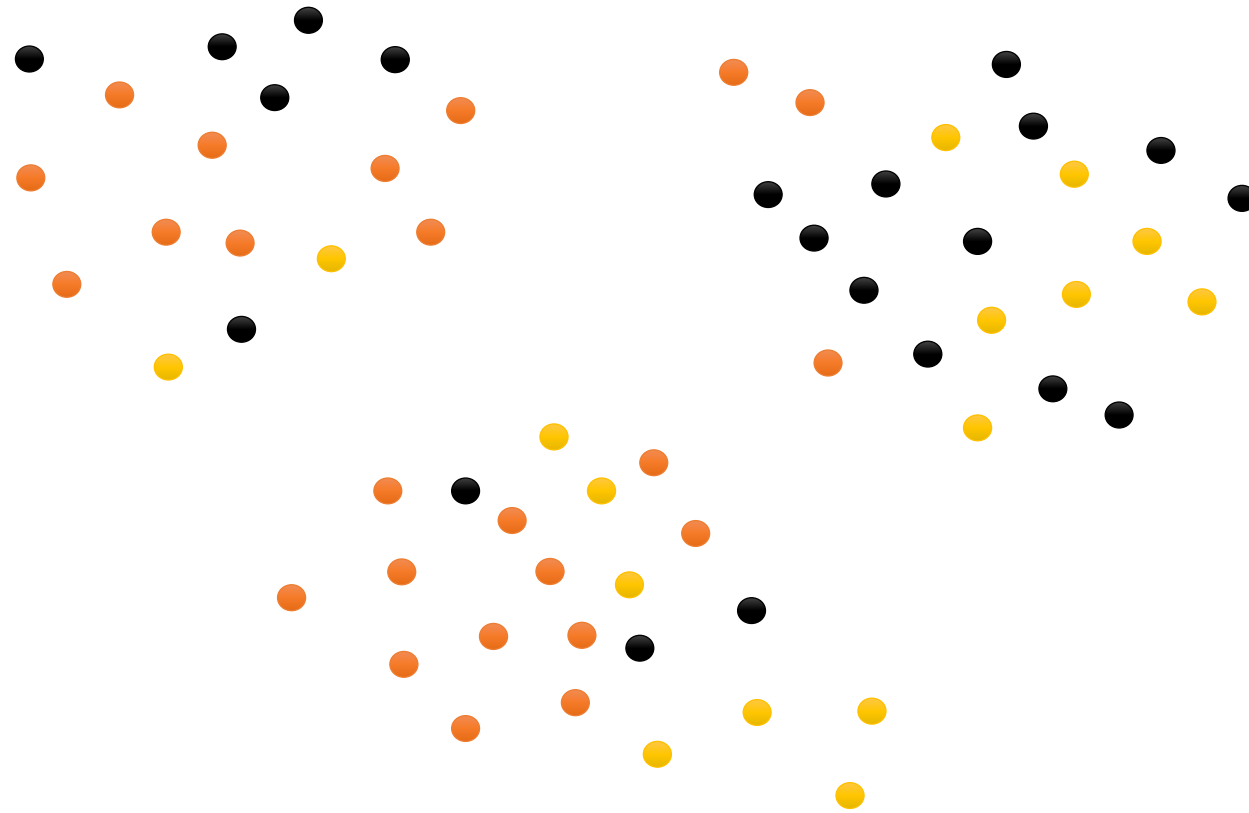
- Yes: at each step, we are reducing the objective function or have converged.
- If assignments do not change, we have a local min.

Can you think of a slight variation to this algorithm that's also valid?

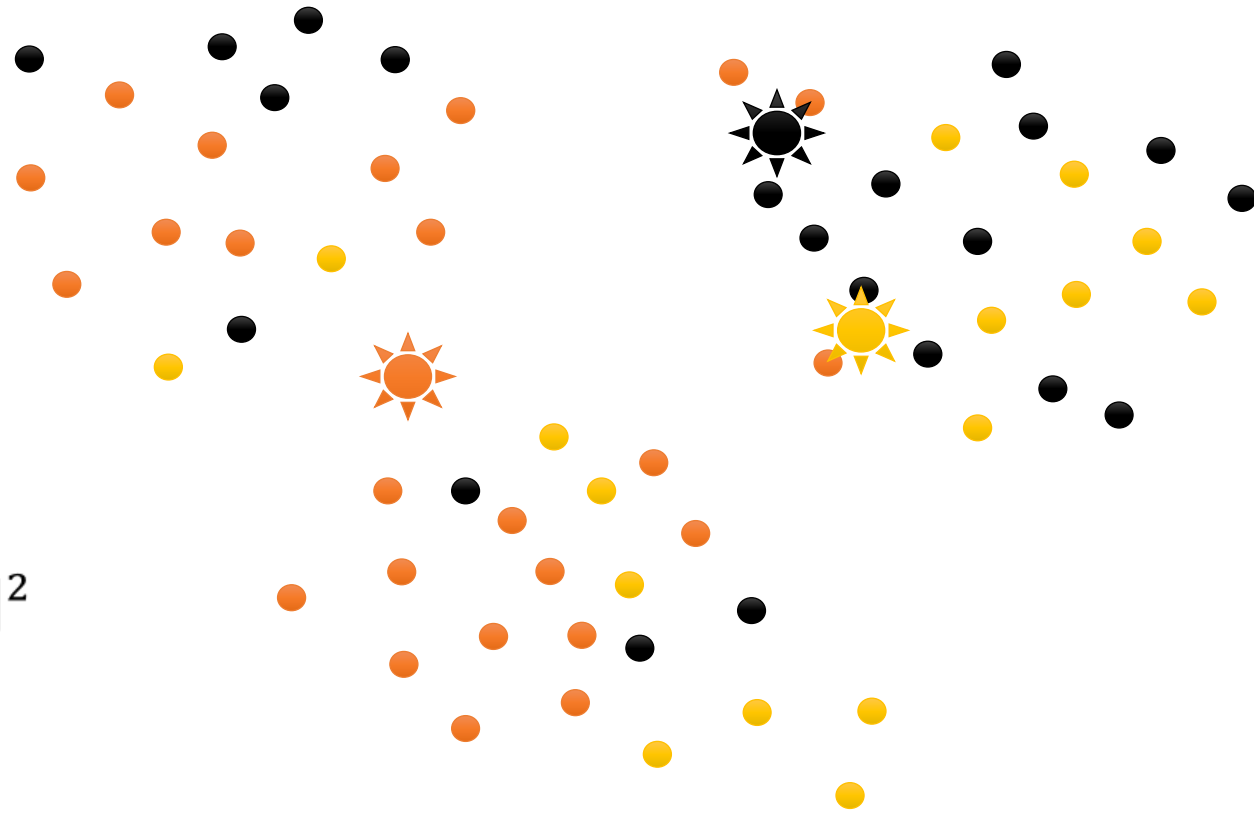
The K-Means algorithm



The K-Means algorithm



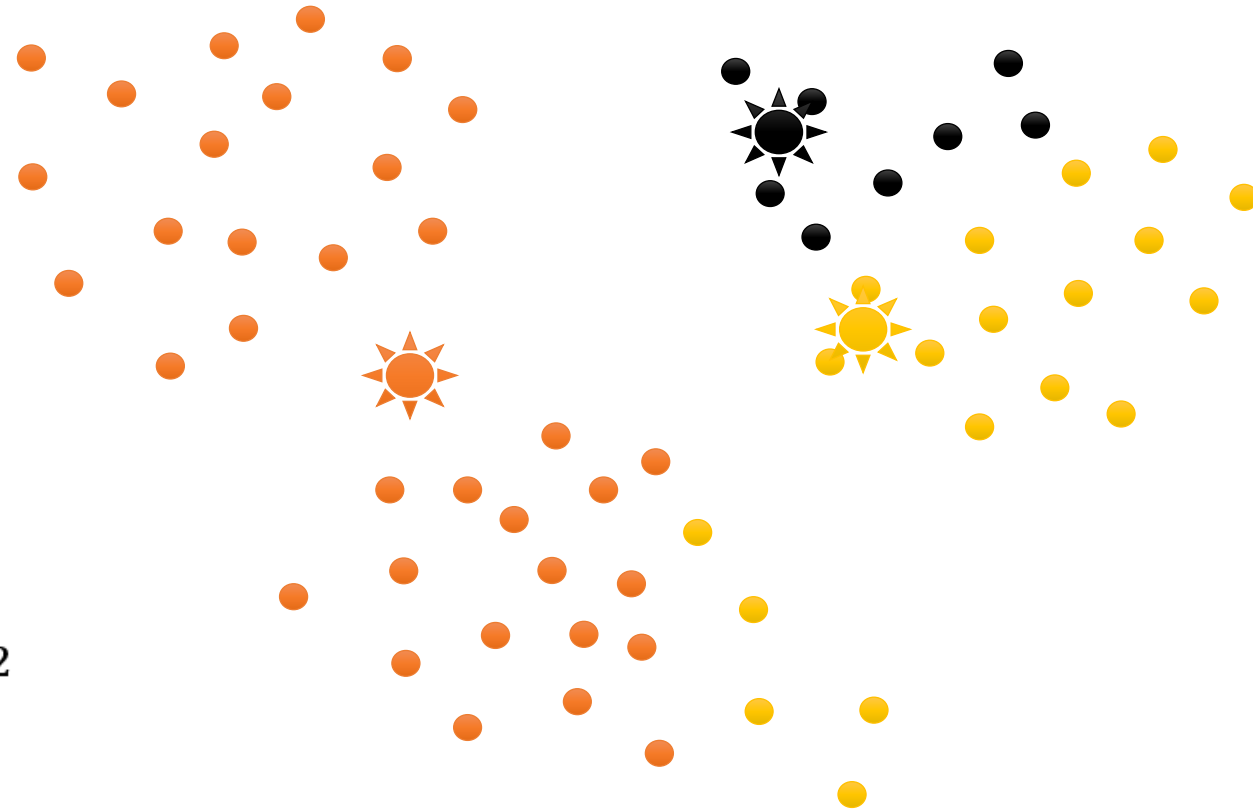
The K-Means algorithm



$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

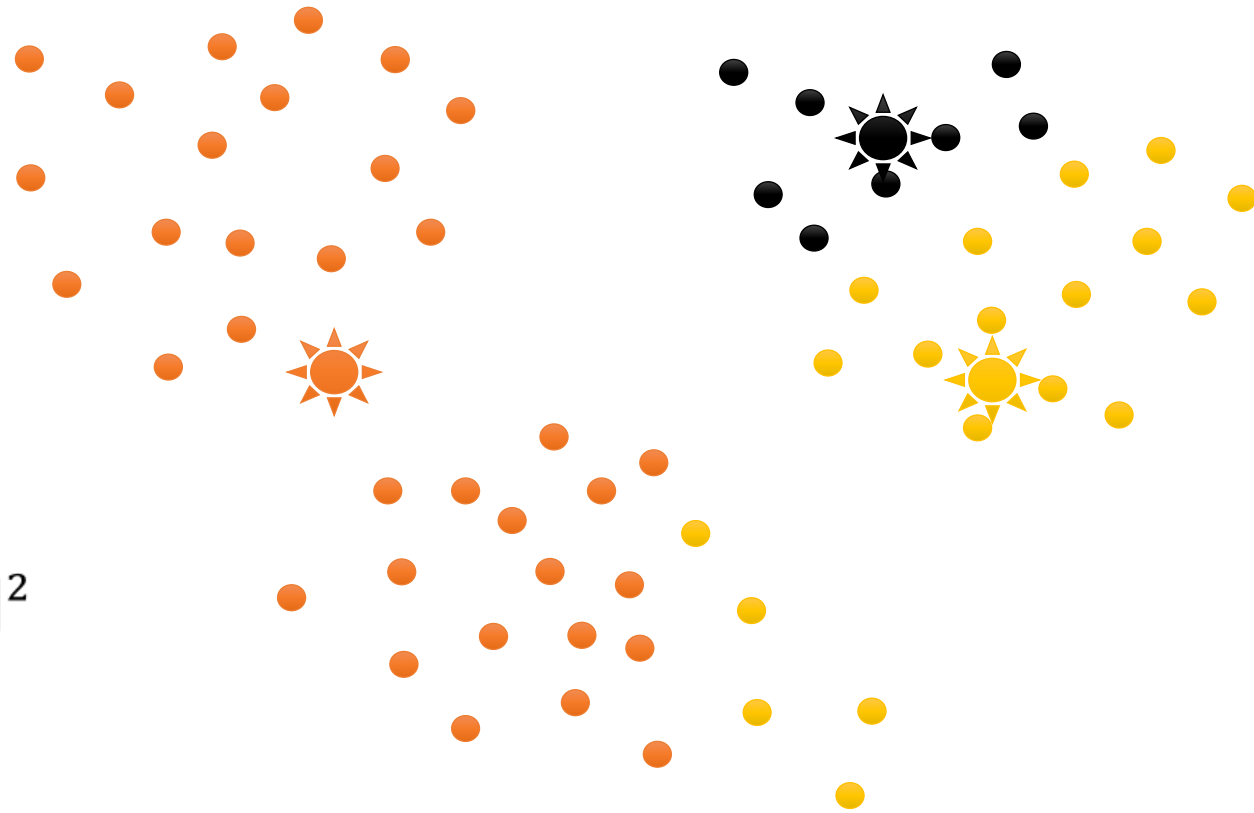
$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$

The K-Means algorithm



$$z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$$
$$\Rightarrow \hat{C}_k = \{x_i | z_i = k\}.$$

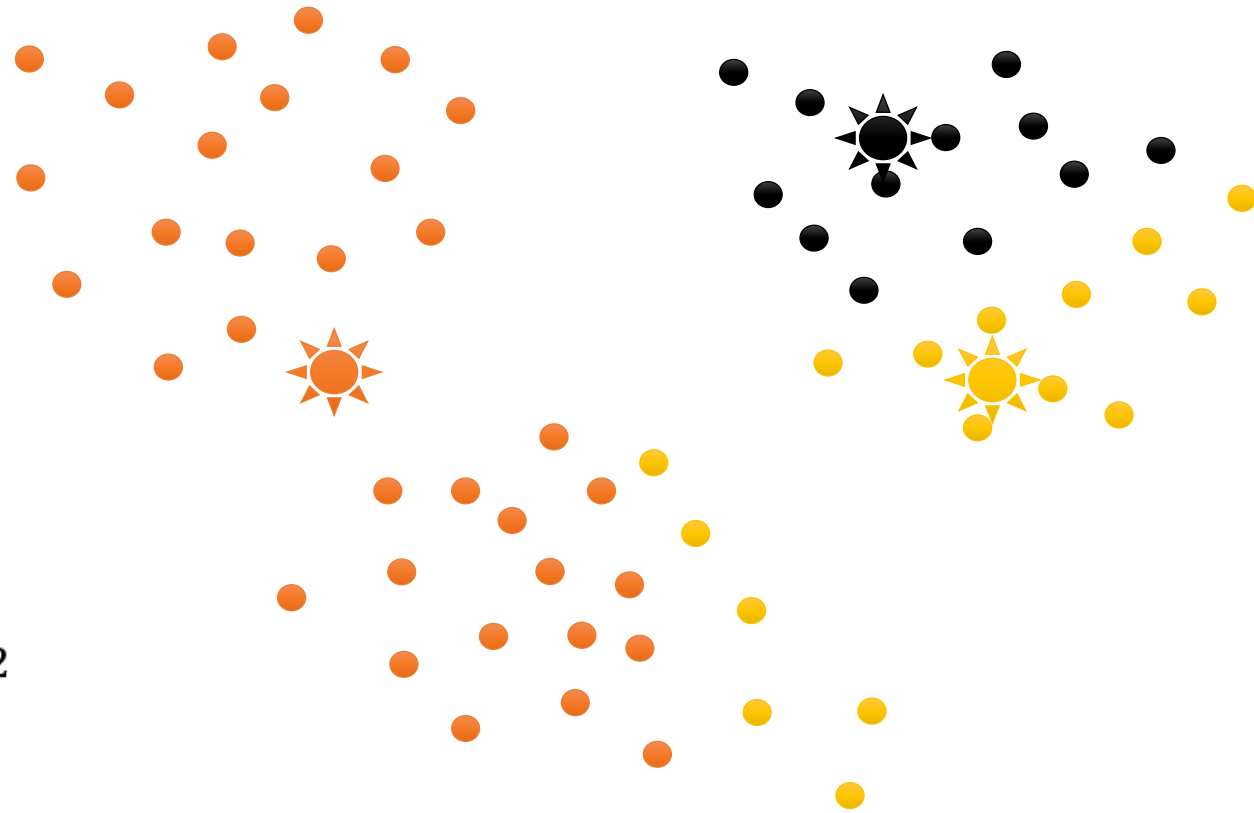
The K-Means algorithm



$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$

The K-Means algorithm

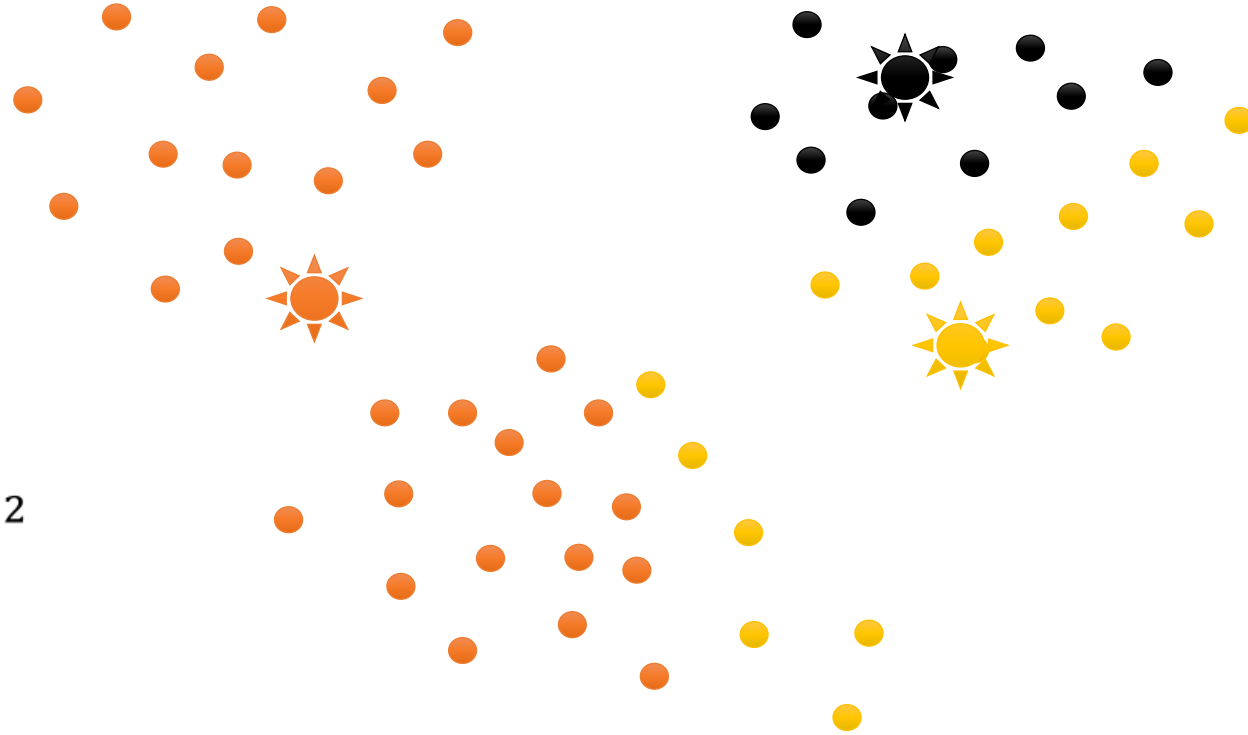


$$z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$$
$$\Rightarrow \hat{C}_k = \{x_i | z_i = k\}.$$

The K-Means algorithm

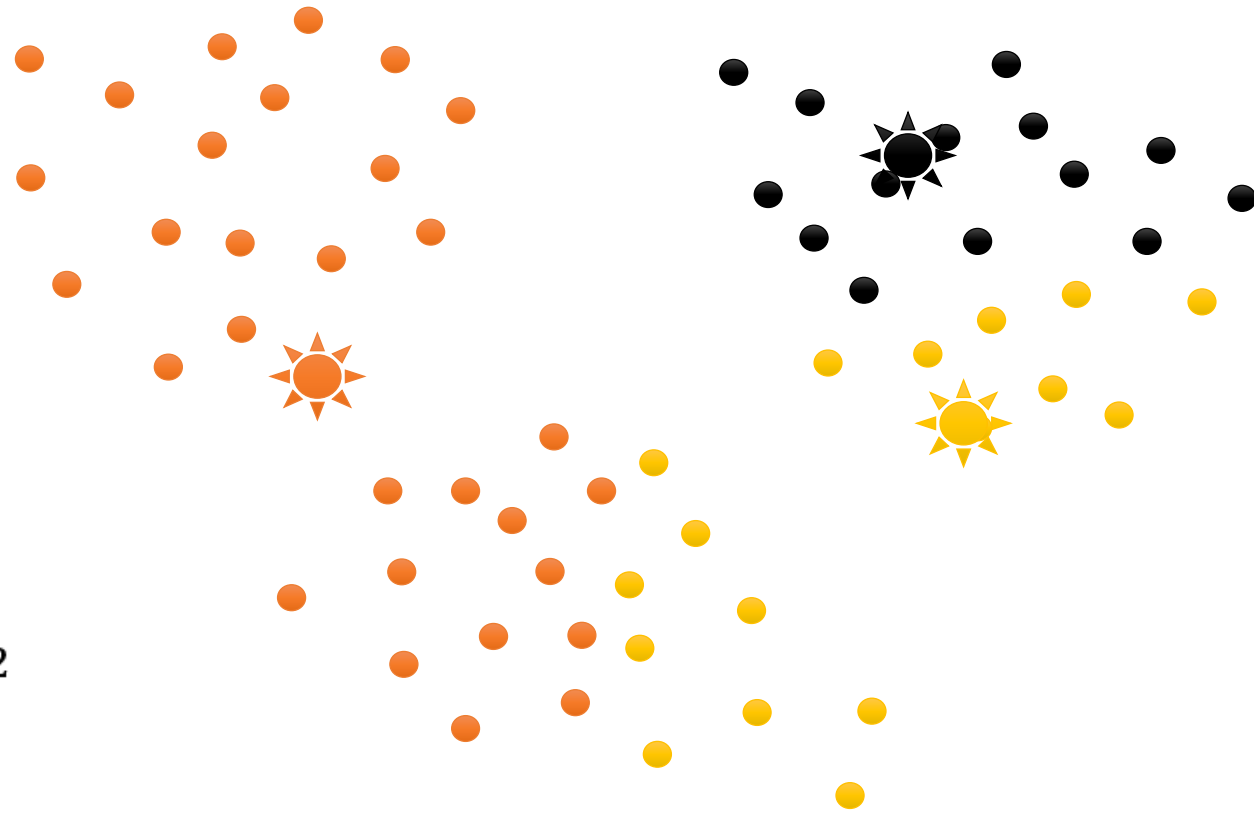
$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$



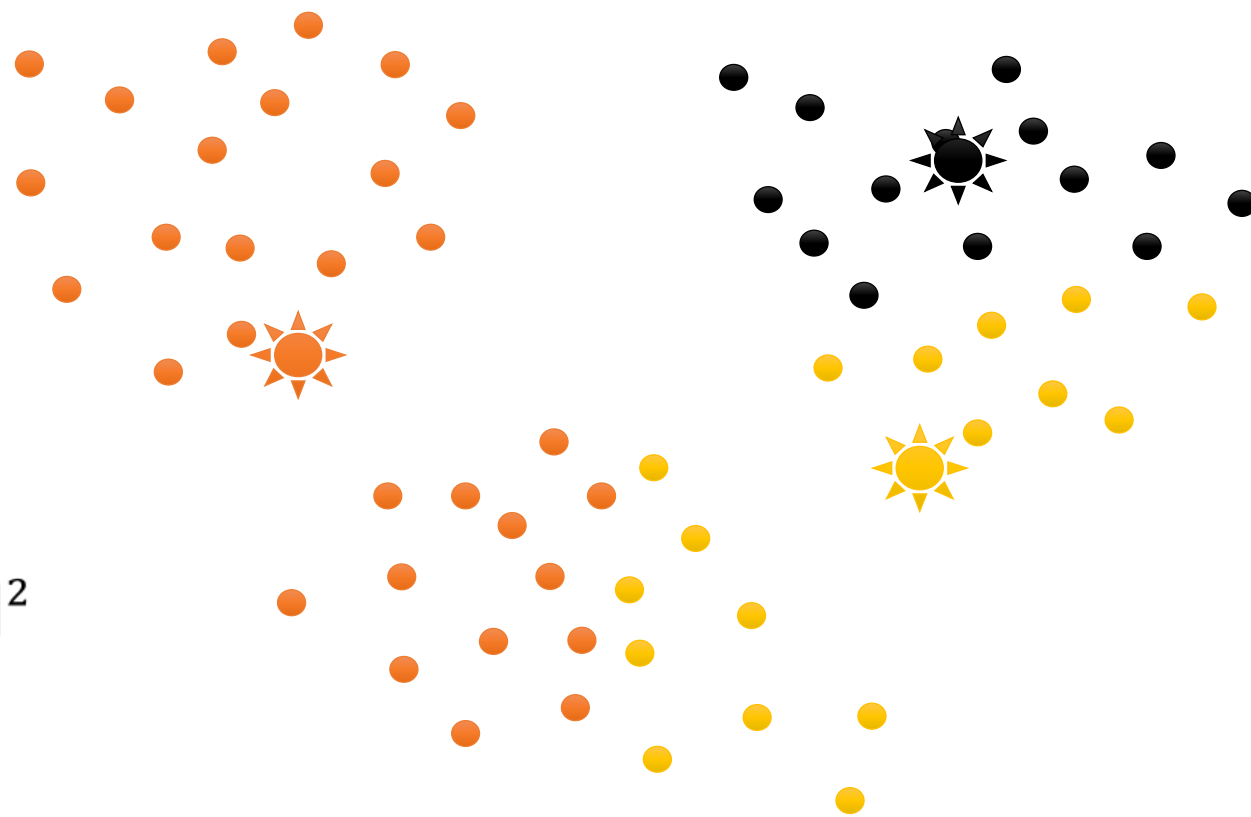
[slide courtesy Yisong Yue]

The K-Means algorithm



$$z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$$
$$\Rightarrow \hat{C}_k = \{x_i | z_i = k\}.$$

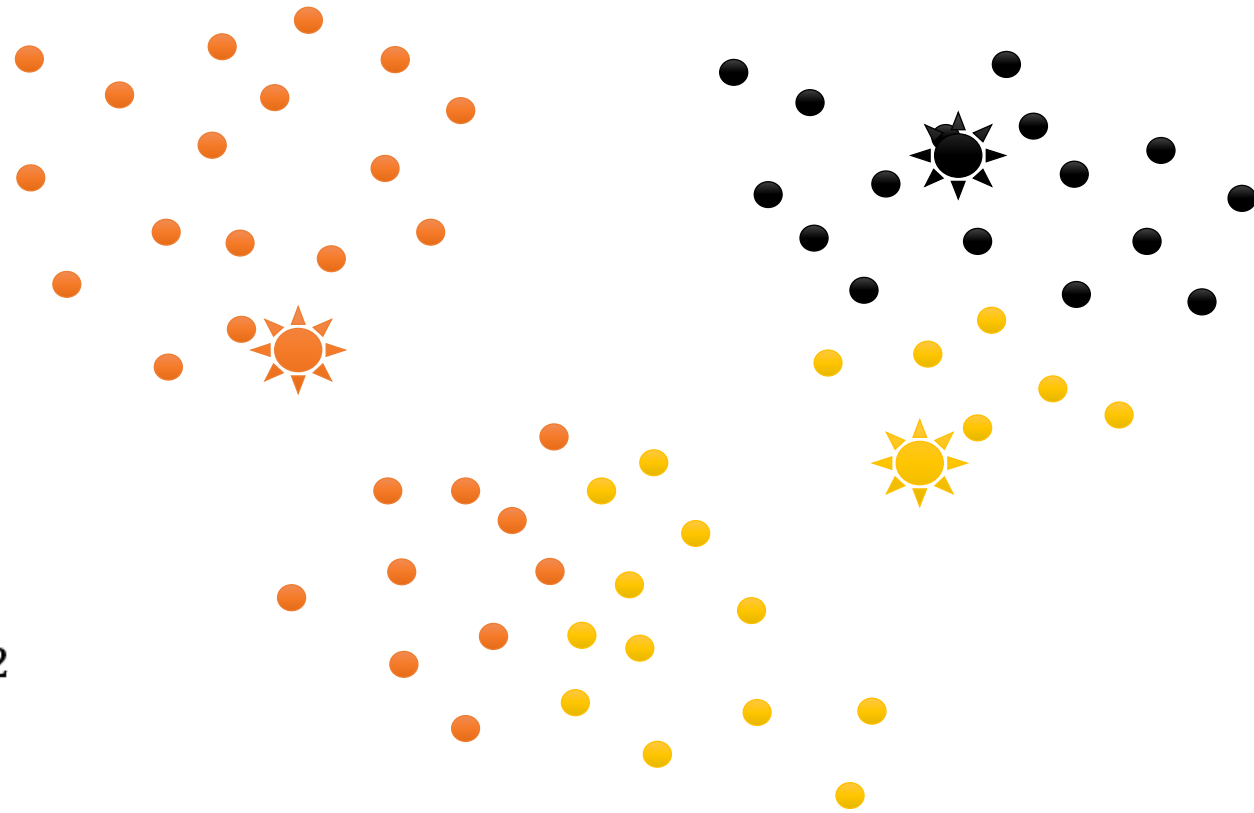
The K-Means algorithm



$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$

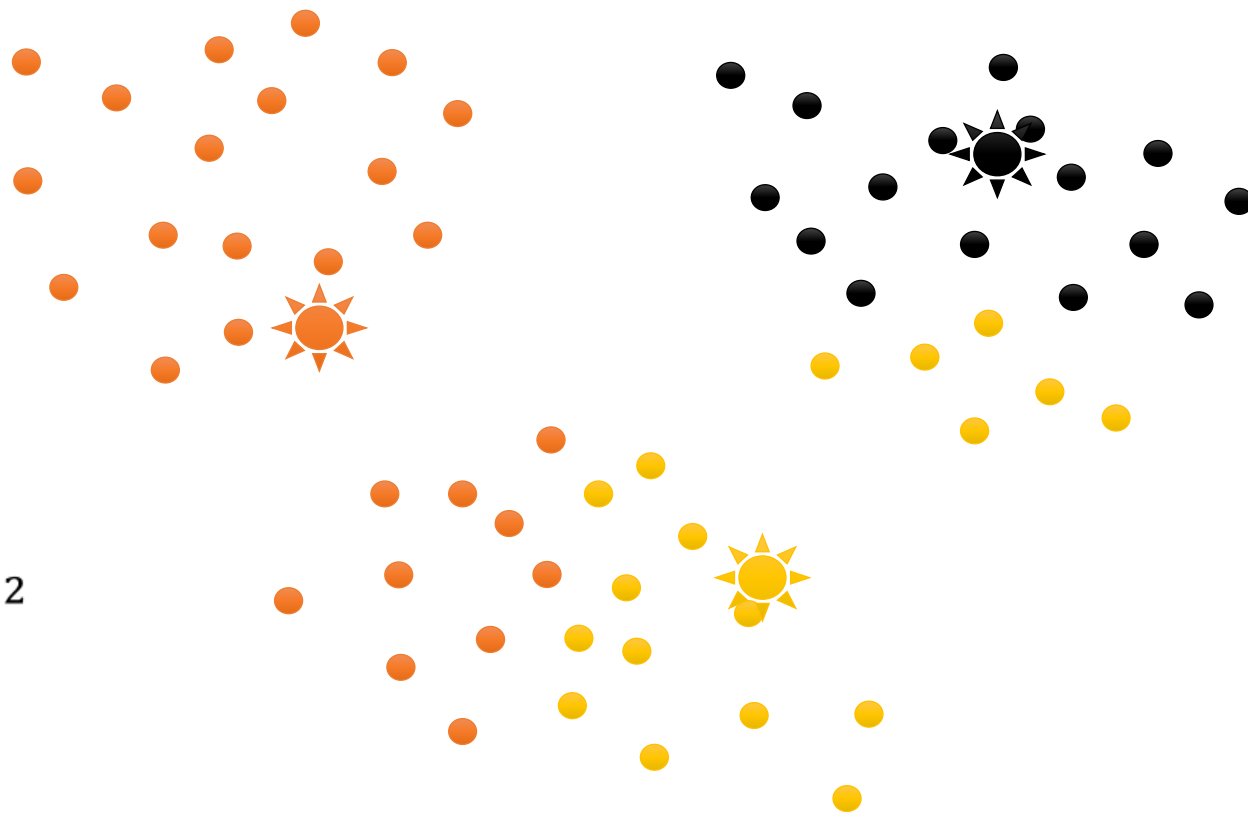
The K-Means algorithm



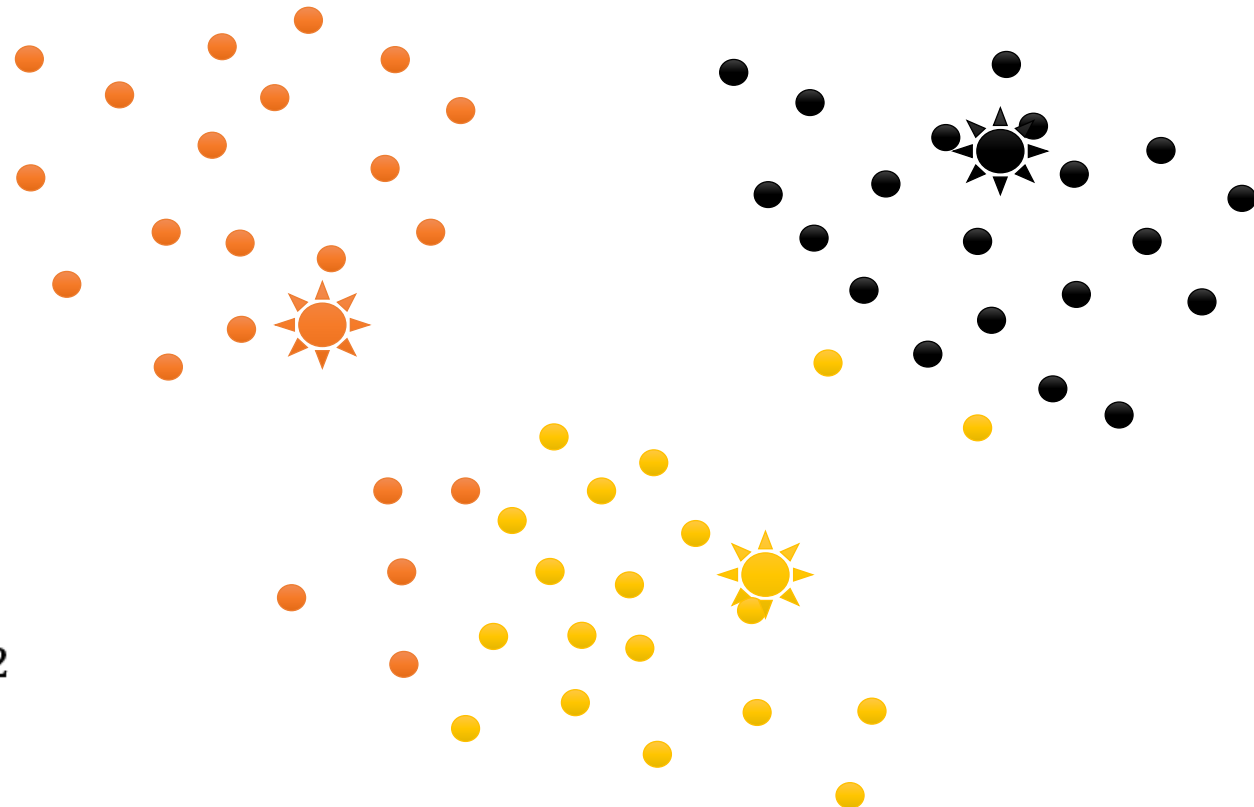
$$z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$$
$$\Rightarrow \hat{C}_k = \{x_i | z_i = k\}.$$

$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$

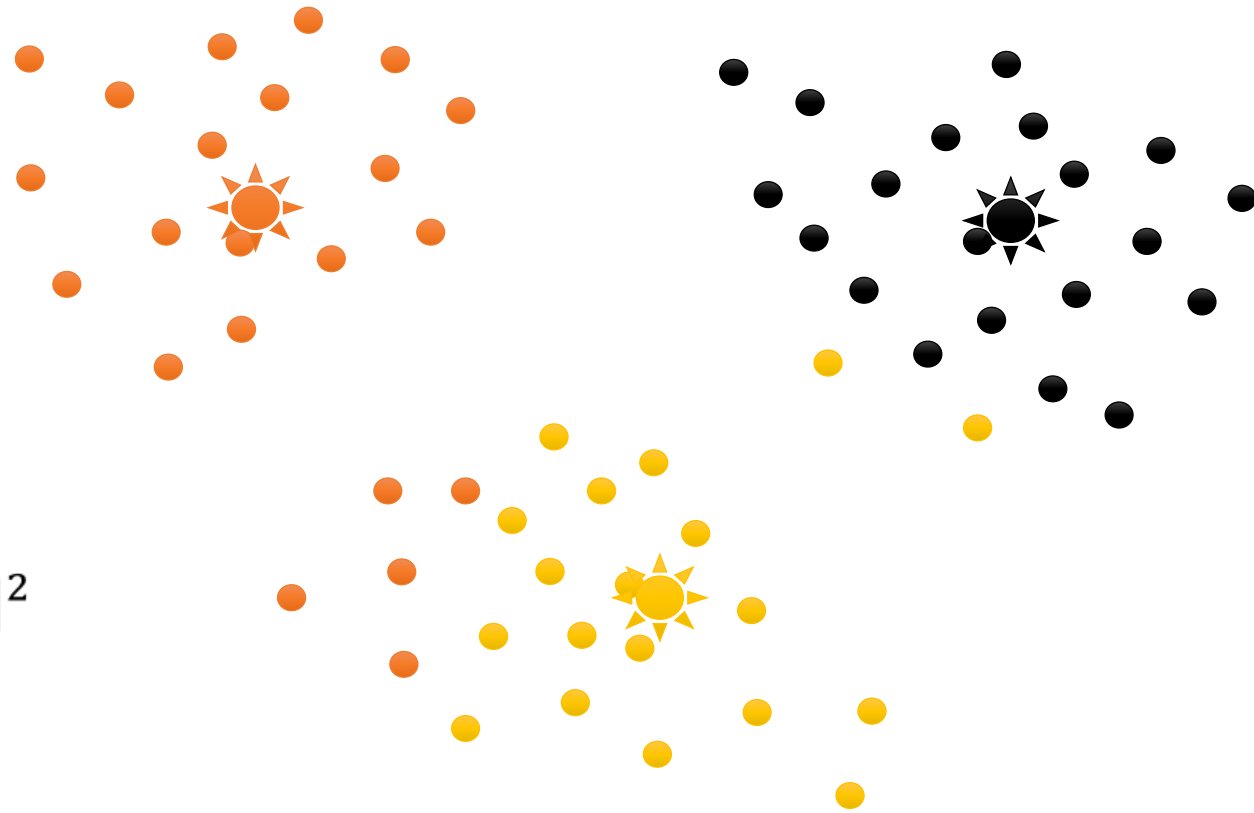


[slide courtesy Yisong Yue]



$$z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$$
$$\Rightarrow \hat{C}_k = \{x_i | z_i = k\}.$$

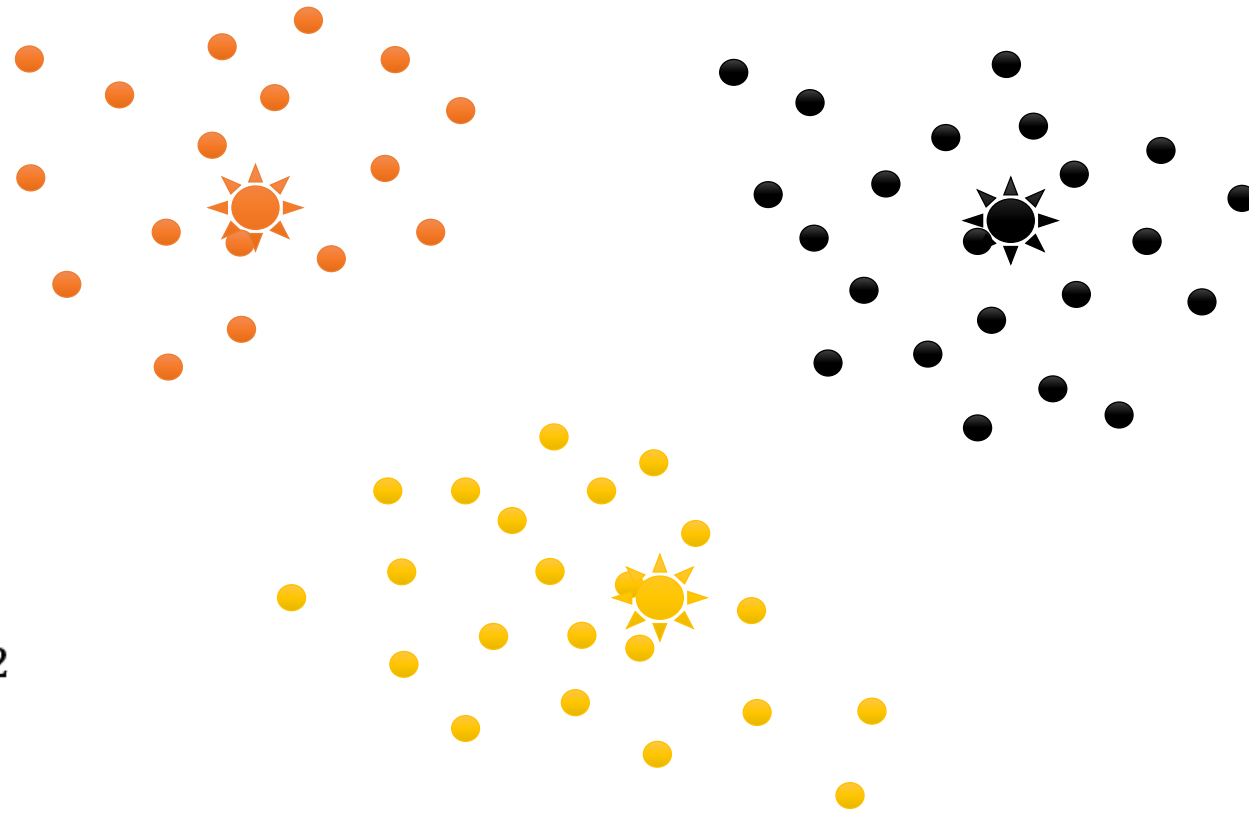
The K-Means algorithm



$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

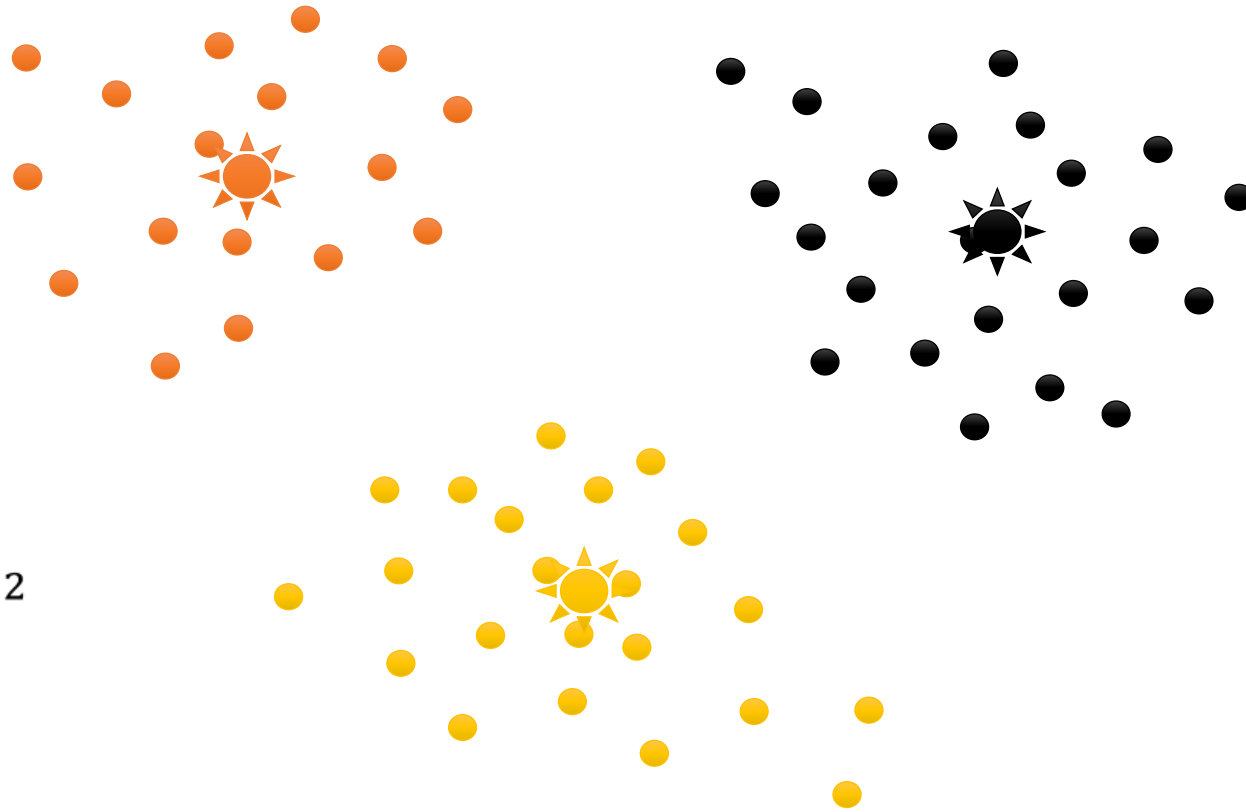
$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$

The K-Means algorithm



$$z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$$
$$\Rightarrow \hat{C}_k = \{x_i | z_i = k\}.$$

The K-Means algorithm



$$\hat{c}_k = \operatorname{argmin}_{c_k} \sum_{x \in C_k} \|x - c_k\|^2$$

$$\Rightarrow \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x$$

Could we instead use gradient descent?

$$loss = L(\{c_k, C_k\}) = \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$

$$\Rightarrow L(\{c_k\}) = \sum_i \min_k \|x_i - c_k\|^2$$

Let z_i be the closest centroid to x_i , then:

$$L(\{c_k\}) = \sum_i \sum_k \|x_i - c_k\|^2 [z_i = k]$$

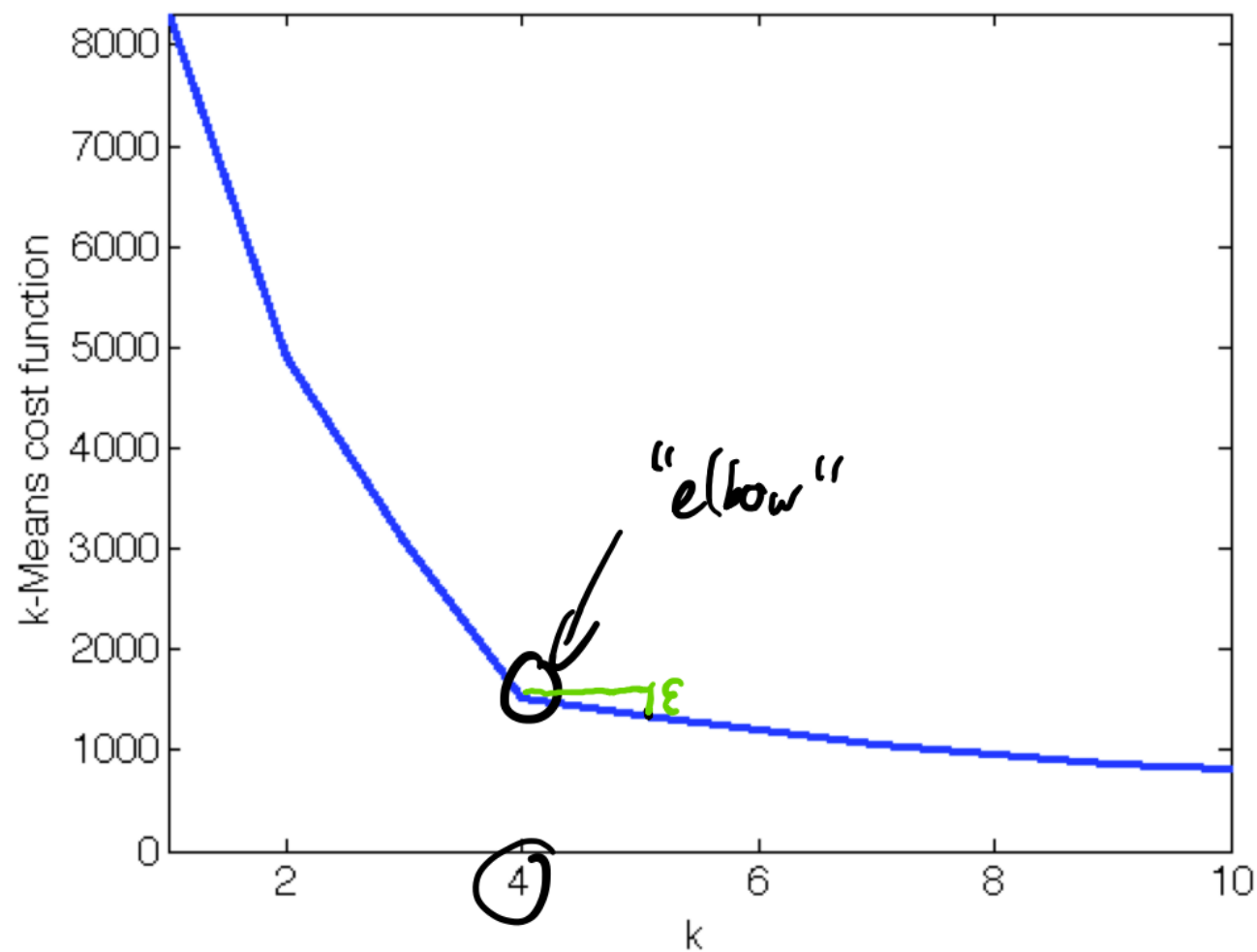
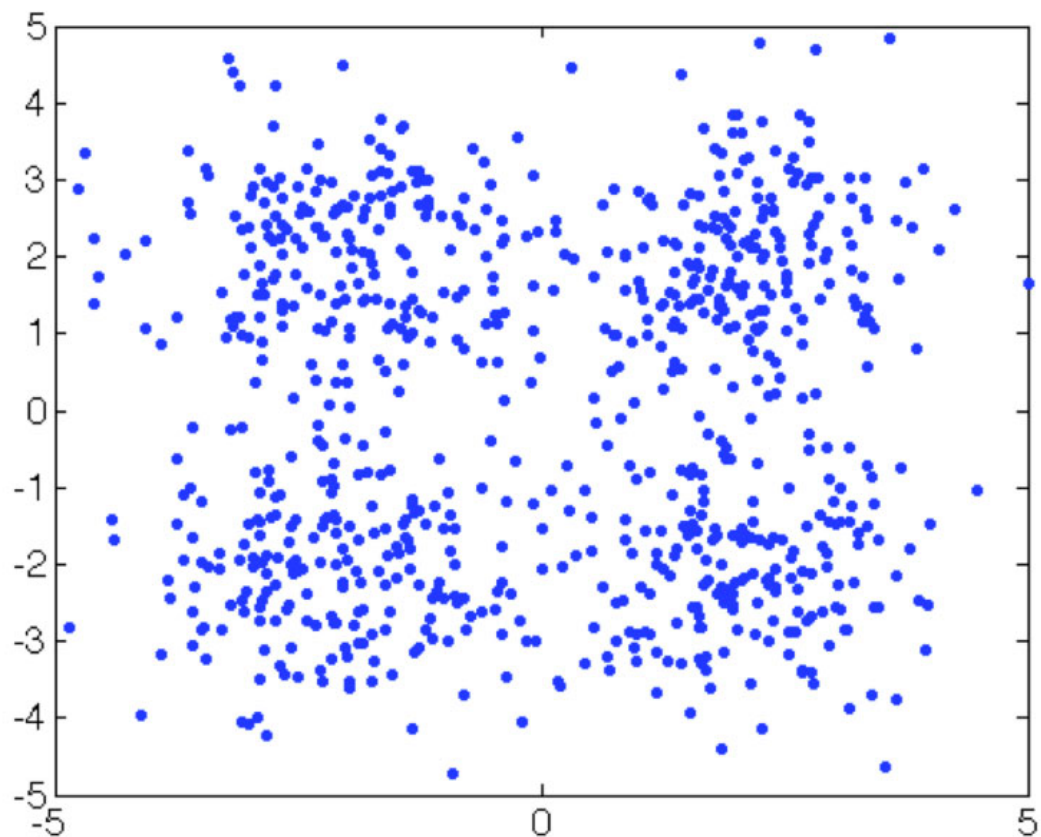
$$\Rightarrow \nabla_{c_k} L(\{c_k\}) = -2 \sum_i (x_i - c_k) [z_i = k]$$

$$\Rightarrow c_{k(new)} = c_{k(old)} - \eta N_k c_{k(old)} + \eta \sum_{x_i | z_i = k} x_i$$

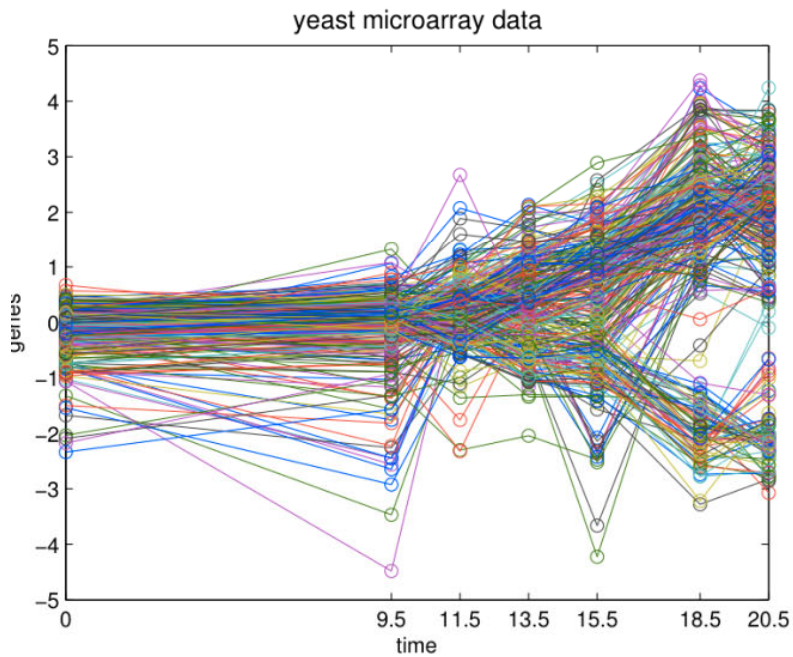
Are we still guaranteed to converge?

Why might this be slower?

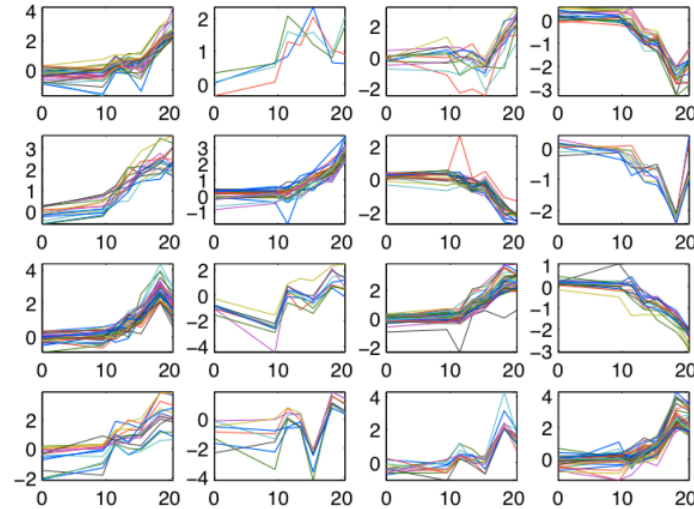
How to find good # of clusters?



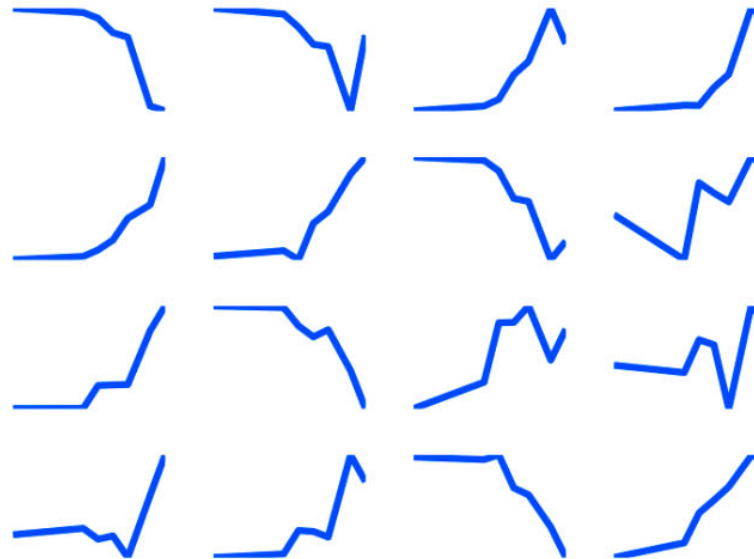
Application of K-Means Clustering



K-Means Clustering of Profiles

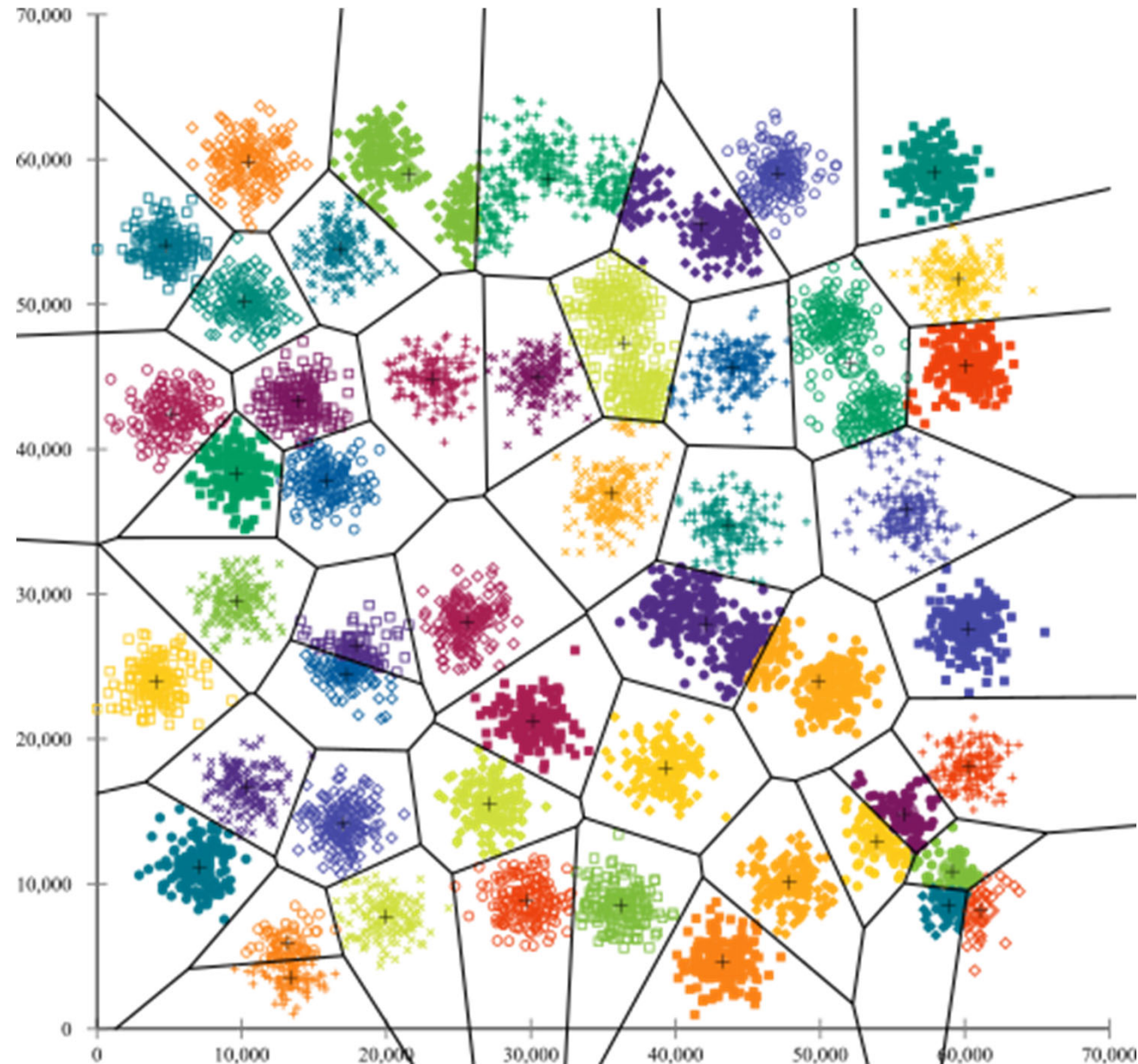


K-Means centroids



clustering yeast
genes by their
“gene expression”
measurements
over time

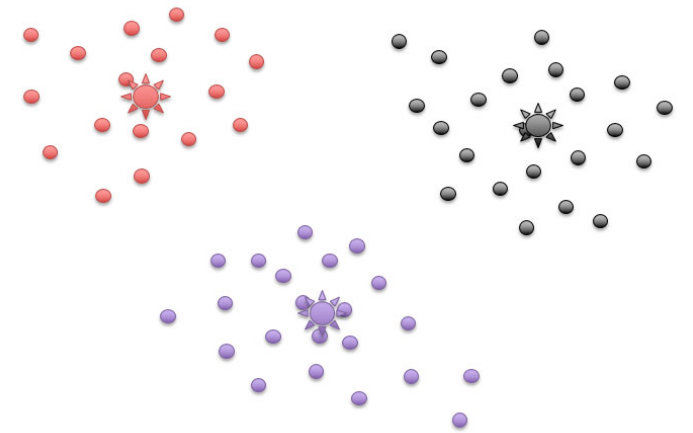
Example of bad local minimum in K-means



More on clustering desiderata

So far we have mentioned:

1. Want high intra-cluster similarity.
2. Want low inter-cluster similarity.



Can you think of any others?

- May want invariances to rotation and or scaling of $\{x_i\}$.
- If clustering depends only on distance/similarity, then whatever invariances these have, the clustering will also have.

Fun fact: Kleinberg's Impossibility Theorem for Clustering

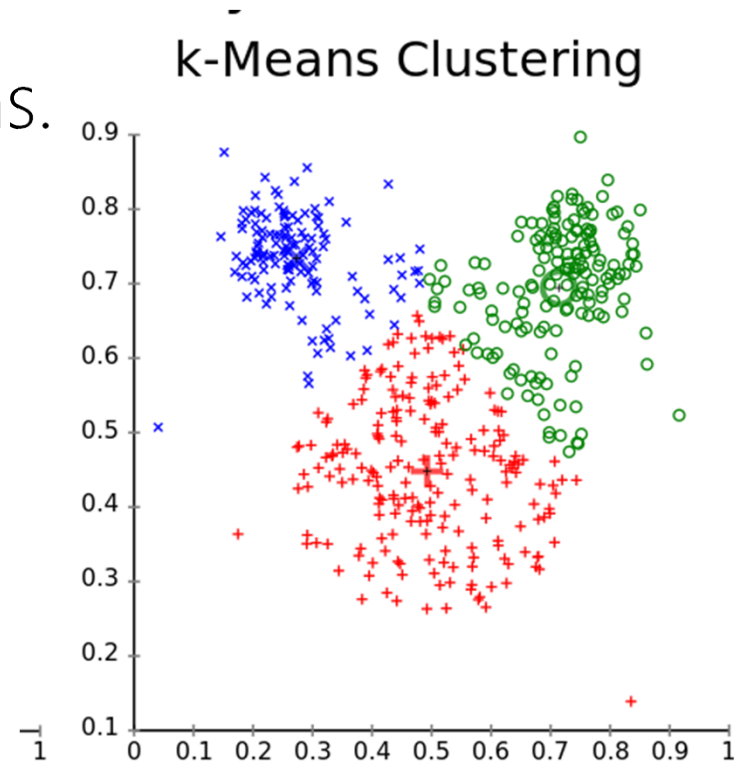
Three (more) clustering desiderata of which provably, one **can achieve only two at a time for a given clustering algorithm**:

1. **Scale-Invariance** (if stretch the data out ($\tilde{d}(j, k) = c \times d(j, k)$), then clustering should stay the same).
2. **Consistency** (if stretch data such that distance within cluster only gets smaller, and between clusters only gets bigger, then clustering should stay the same).
3. **Richness** (clustering function should be able to produce any arbitrary partition/clustering of data points).

Lets revisit K-means—any weaknesses?

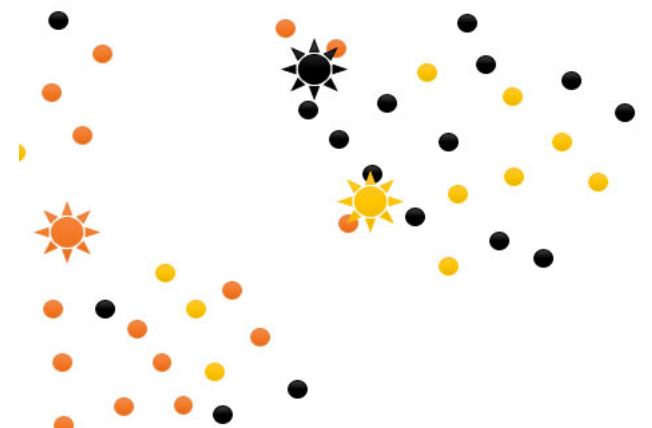
$$\operatorname{argmin}_{S=C_1 \cup \dots \cup C_K, \{c_1, \dots, c_K\}} \sum_k \sum_{x \in C_k} \|x - c_k\|^2$$

1. No likelihood, so hard to understand assumptions.
2. e.g., implicitly corresponds to clusters with “spherical” shape because each feature is treated equally.
3. Each step in the optimization has a “hard” assignment which means that can’t have any uncertainty as to which point belongs to which cluster, which feels “brittle”



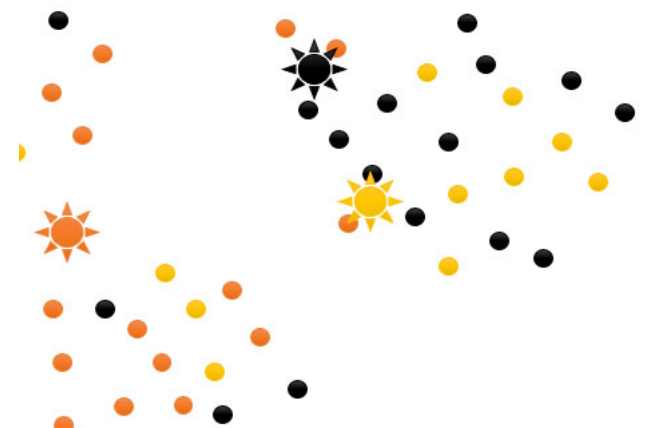
Lets develop a "soft" K-means algorithm

- Previously: $z_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$, and then $\hat{C}_k = \{x_i | z_i = k\}$.
- Convert to max and exp it, $z_i = \underset{k}{\operatorname{argmax}} \exp(-\|x_i - c_k\|^2)$.
- Let $v_{ik} \equiv \exp(-\|x_i - c_k\|^2)$ so that $z_i = \underset{k}{\operatorname{argmax}} \{v_{ik}\}$.
- Now normalize the $\{v_{ik}\}$ so that $r_{ik} \equiv \frac{v_{ik}}{\sum_k v_{ik}}$
- Use r_{ik} as the soft/probabilistic cluster assignments.
- This is just the familiar softmax:
- $r_{ik} = \frac{\exp[v_{ik}]}{\sum_j \exp[v_{ij}]} = \operatorname{softmax}(\{v_{ij}\})[k]$
- $r_{ik} \equiv \frac{\exp[\beta v_{ik}]}{\sum_j \exp[\beta v_{ij}]} = \operatorname{softmax}(\{\beta v_{ij}\})[k]$



Consider a “soft” K-means algorithm

- In [52]: `data=np.array([-5,-3,-7])` $k\}$.
- In [53]: `softmax(data)`
Out[53]: `array([0.11731043, 0.86681333, 0.01587624])`
- In [54]: `softmax(0.00001*data)`
Out[54]: `array([0.33333333, 0.33334 , 0.33332667])`
- In [55]: `softmax(100*data)`
Out[55]: `array([1.38389653e-087, 1.00000000e+000, 1.91516960e-174])`
- Use r_{ik} as the soft/probabilistic cluster assignments.
- This is just the familiar softmax:
- $$r_{ik} = \frac{\exp[v_{ik}]}{\sum_j \exp[v_{ij}]} = \text{softmax}(\{v_{ij}\})[k]$$
- $$r_{ik} = \frac{\exp[\beta v_{ik}]}{\sum_j \exp[\beta v_{ij}]} = \text{softmax}(\{\beta v_{ij}\})[k]$$



Consider a "soft" K-means algorithm

```

In [52]: data=np.array([-5
In [53]: softmax(data)
Out[53]: array([ 0.1173104
In [54]: softmax(0.00001*d
Out[54]: array([ 0.3333333
In [55]: softmax(100*data)
Out[55]: array([ 1.383896
    
```

Use r_{ik} as the soft/probability

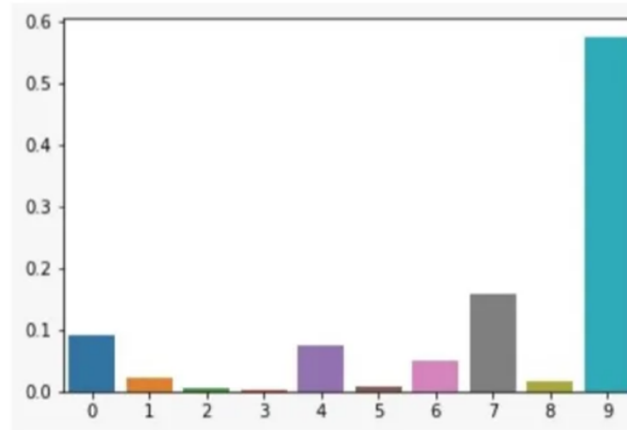
This is just the familiar

$$r_{ik} = \frac{\exp[v_{ik}]}{\sum_j \exp[v_{ij}]} = \text{softmax}$$

$$r_{ik} = \frac{\exp[\beta v_{ik}]}{\sum_j \exp[\beta v_{ij}]} = \text{softmax}(\{\beta v_{ij}\})[k]$$

SOFTMAX WITHOUT TEMPERATURE (T=1)

$$\frac{e^{z_i}}{\sum_j e^{z_j}}$$

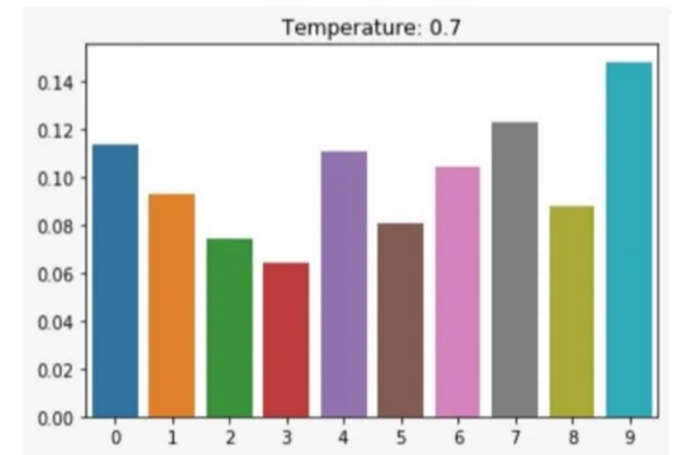


LESS ENTROPY

INCREASE IN ENTROPY
WITH INCREASE IN T

SOFTMAX WITH TEMPERATURE

$$\frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$



MORE ENTROPY



Generalize hard to soft k-means:

Repeat until convergence:

1. Replace $r_i \equiv \underset{k}{\operatorname{argmin}} \|x_i - c_k\|^2$ with

$$r_{ik} = \operatorname{softmax}(\{-\beta \|x_i - c_k\|^2\}) \text{ (yields a "soft partition")}$$

2. Replace $\hat{c}_k = \underset{c_k}{\operatorname{argmin}} \sum_{x \in C_k} \|x - c_k\|^2$ with

$$\hat{c}_k = \underset{c_k}{\operatorname{argmin}} \sum_{i=1}^N r_{ik} \|x_i - c_k\|^2$$

$$\text{Had, } \hat{c}_k = \frac{1}{N} \sum_{x \in C_k} x, \text{ now have, } \hat{c}_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}.$$

(Reduces to hard assignment if β is high, which causes $r_{ik} \in \{0,1\}$)

Un-answered issues with “soft” K-means

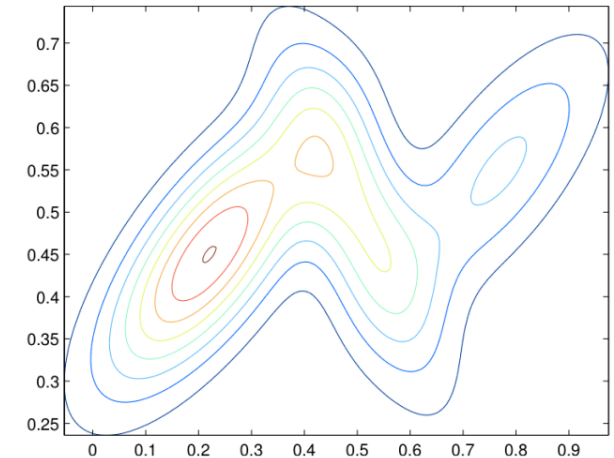
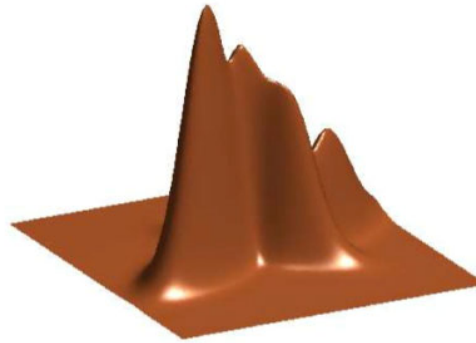
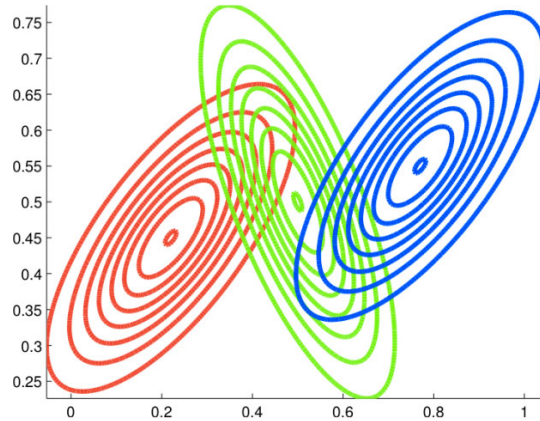
1. How should we set β ? (not clear)
2. We are still treating all the features in \mathbf{x}_i equally. Does this make sense? It implies a spherical cluster. But what if cluster would be “better” elongated (non-spherical)? But how?

Going “soft” has gotten us some flexibility, but we can do better.

Lets go to a fully probabilistic model! (Mixture of Gaussians)

Mixture of Gaussians (MoG)

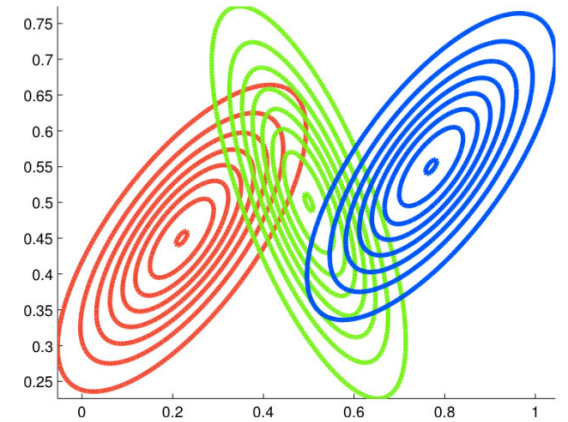
- Each cluster is now represented by a Gaussian $N(\mathbf{x}_i | \mathbf{u}_k, \Sigma_k)$, with two free parameters
- Now we can write down a likelihood and perform MLE!



Mixture of Gaussians (MoG) likelihood for one x_i

- Let $z_i \in \{1, \dots, K\}$ be the hidden/unobserved assigned cluster to x_i . We don't know its value, so have to marginalize it (sum it out):

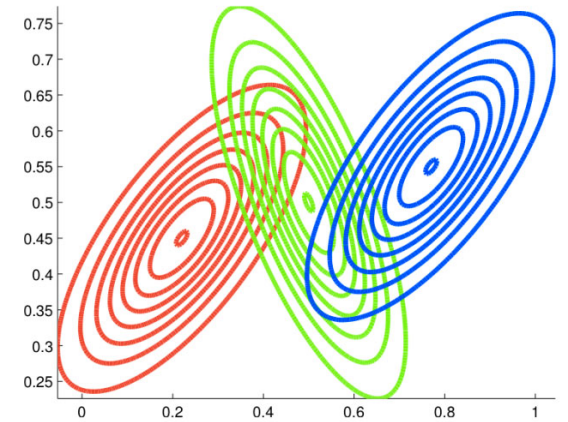
$$L_i = p(x_i) = \sum_{k=1}^K p(x_i, z_i = k)$$



Mixture of Gaussians (MoG) likelihood for one x_i

- Let $z_i \in \{1, \dots, K\}$ be the hidden/unobserved assigned cluster to x_i . We don't know its value, so have to marginalize it (sum it out):

$$\begin{aligned} L_i &= p(x_i) = \sum_{k=1}^K p(x_i, z_i = k) \\ &= \sum_{k=1}^K p(x_i | z_i = k) p(z_i = k) \\ &= \sum_{k=1}^K N(x_i | \mu_k, \Sigma_k) p(z_i = k) \end{aligned}$$

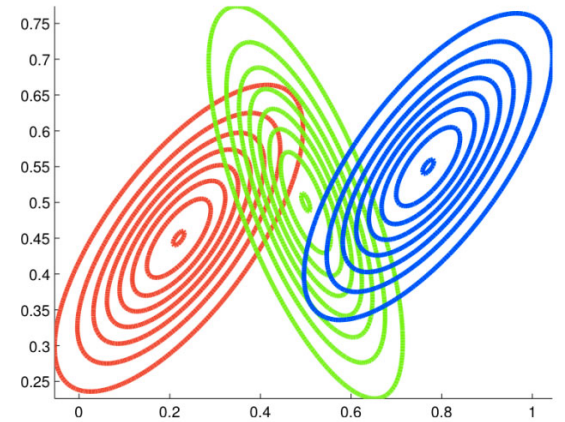


Mixture of Gaussians (MoG) likelihood for one x_i

- Let $z_i \in \{1, \dots, K\}$ be the hidden/unobserved assigned cluster to x_i . We don't know its value, so have to marginalize it (sum it out):

$$\begin{aligned} L_i &= p(x_i) = \sum_{k=1}^K p(x_i, z_i = k) \\ &= \sum_{k=1}^K p(x_i | z_i = k) p(z_i = k) \\ &= \sum_{k=1}^K N(x_i | \mu_k, \Sigma_k) p(z_i = k) \\ &= \sum_{k=1}^K N(x_i | \mu_k, \Sigma_k) \alpha_k \end{aligned}$$

where $\alpha_k \equiv p(z_i = k)$ and $\sum_k \alpha_k = 1$



- The parameters we want to learn are $\theta \equiv \{\mu_k, \Sigma_k, \alpha_k\}$.
- α_k are called the "mixing weights".
- Now we can use MLE on $LL = \log \prod_i L_i = \sum_i \log L_i$.

Alternative uses of MoG beyond clustering

Once we have estimated the values of $\theta = \{\mu_k, \Sigma_k, \alpha_k\}$ from the training data, we can make calls to $p(x|\theta)$, for any data point in the training data or otherwise.

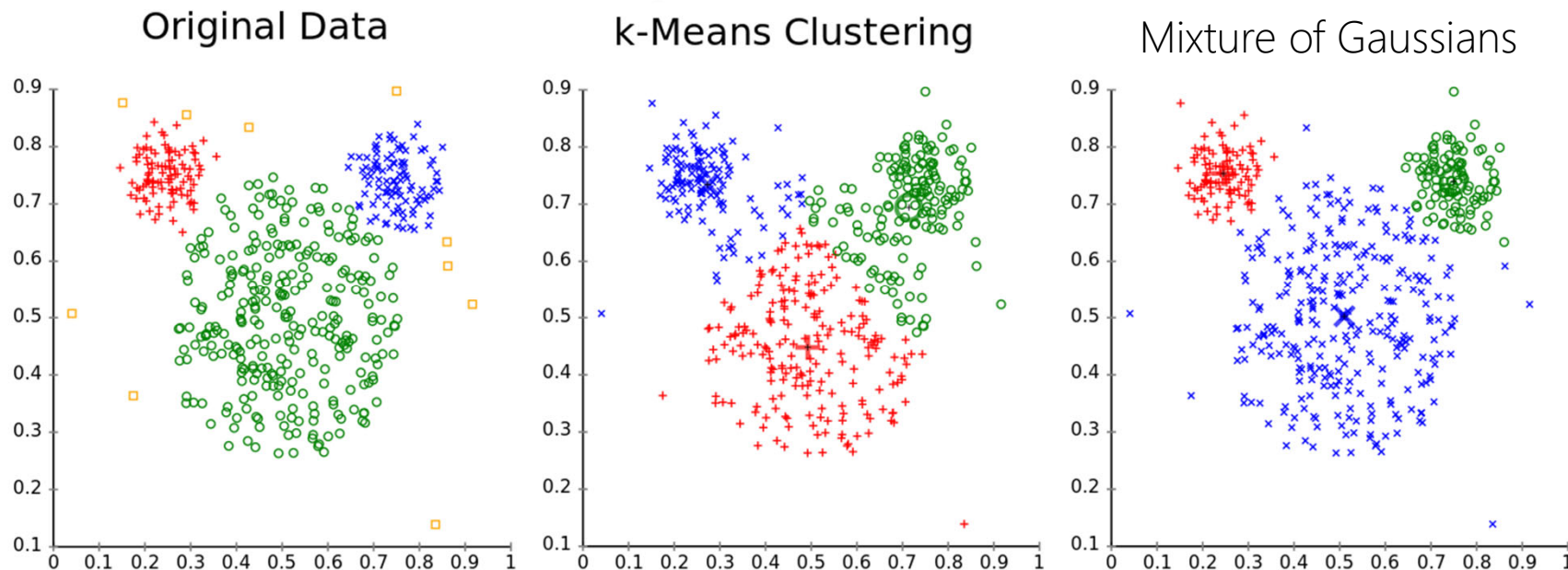
- So we have also performed *density estimation*.

We can also use it to generate data, $x \sim p(x|\theta)$.

- So we have a *generative model*:
 1. For each point, j , sample cluster $c_j \sim \text{multinomial}(\{\alpha_k\})$.
 2. Then sample from the corresponding Gaussian.

K-means vs. Mixture of Gaussians

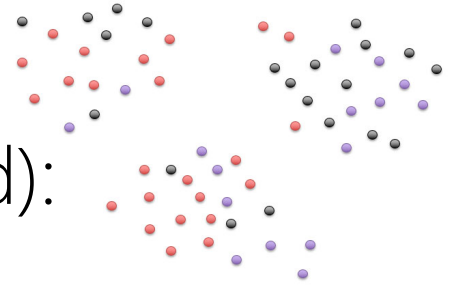
- If we take the zero noise limit in Mixture of Gaussians (zero variance in the Gaussians), we get K-means.
- MoG allows non-spherical clusters (via the covariance matrix).
- And different covariance per cluster, which is helpful here:



K-means vs. Mixture of Gaussians

- MoG: explicit assumptions in the form of statistical distributions.
- Thus easier to generalize, while understanding assumptions.
- Can derive principled objective in the form of a likelihood, which involves marginalizing over the hidden/latent variable (cluster assignment).
- There is a special form of MLE for these latent variables called Expectation-Maximization.

EM for Mixture of Gaussians $\theta \equiv \{\mu_k, \Sigma_k, \alpha_k\}$.



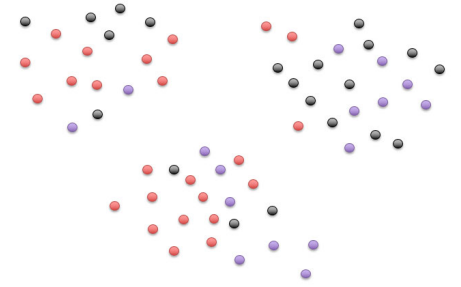
Intuitive Description of EM (EM is exact maximum likelihood):

Initialize with random cluster assignments

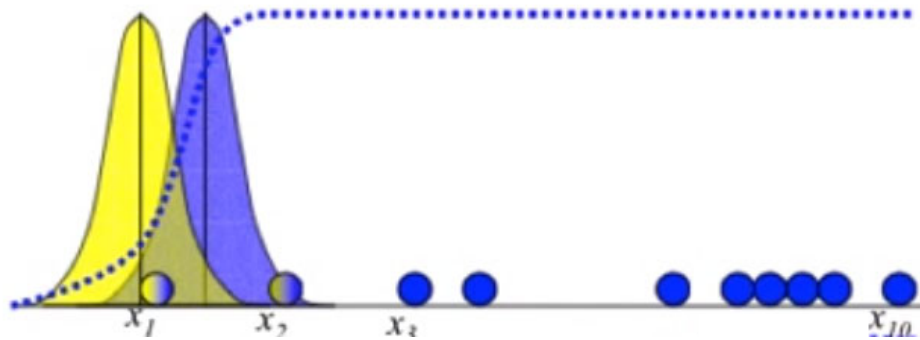
- i. use current parameter estimates to (probabilistically) estimate $\{p(\mathbf{z}_i | \mathbf{x}_i, \theta)\}$ (i.e. "fill in the missing data": E-step)
 - ii. do MLE on "fully observed" data (where \mathbf{z}^n are probabilistically filled in: M-step).
- This is a lot like the K-means algorithm, only now with a principled loss function and parameter estimation principle.
 - This procedure yields the MLE solution for MoG (and generally for latent variable models).

EM for Mixture of Gaussians

$$\theta \equiv \{\mu_k, \Sigma_k, \alpha_k\}.$$

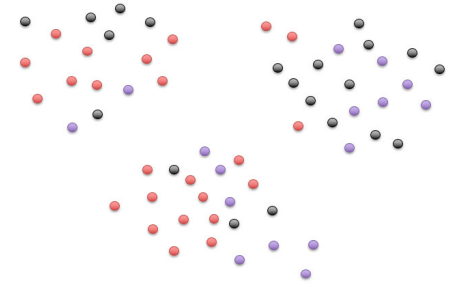


EM: 1-d example

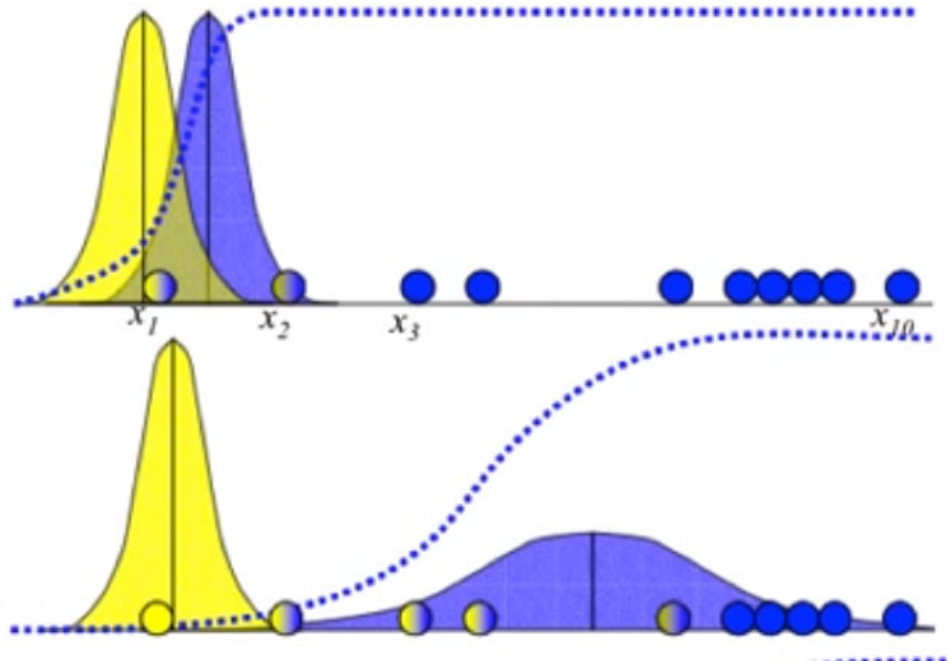


EM for Mixture of Gaussians

$$\theta \equiv \{\mu_k, \Sigma_k, \alpha_k\}.$$

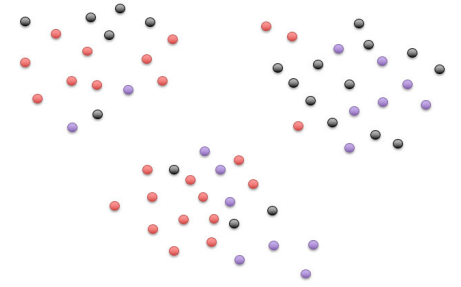


EM: 1-d example



EM for Mixture of Gaussians

$$\theta \equiv \{\mu_k, \Sigma_k, \alpha_k\}.$$



EM: 1-d example

