University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chenhao Tan

Lecture 18: Unsupervised Learning II (Clustering)

Slides adapted from Jordan Boyd-Graber, Chris Ketelsen

Learning objectives

- Learn about general clustering
- Learn about the K-Means algorithm
- Learn about Gaussian Mixture Models

Supervised learning

Data: X

Labels: Y

Unsupervised learning

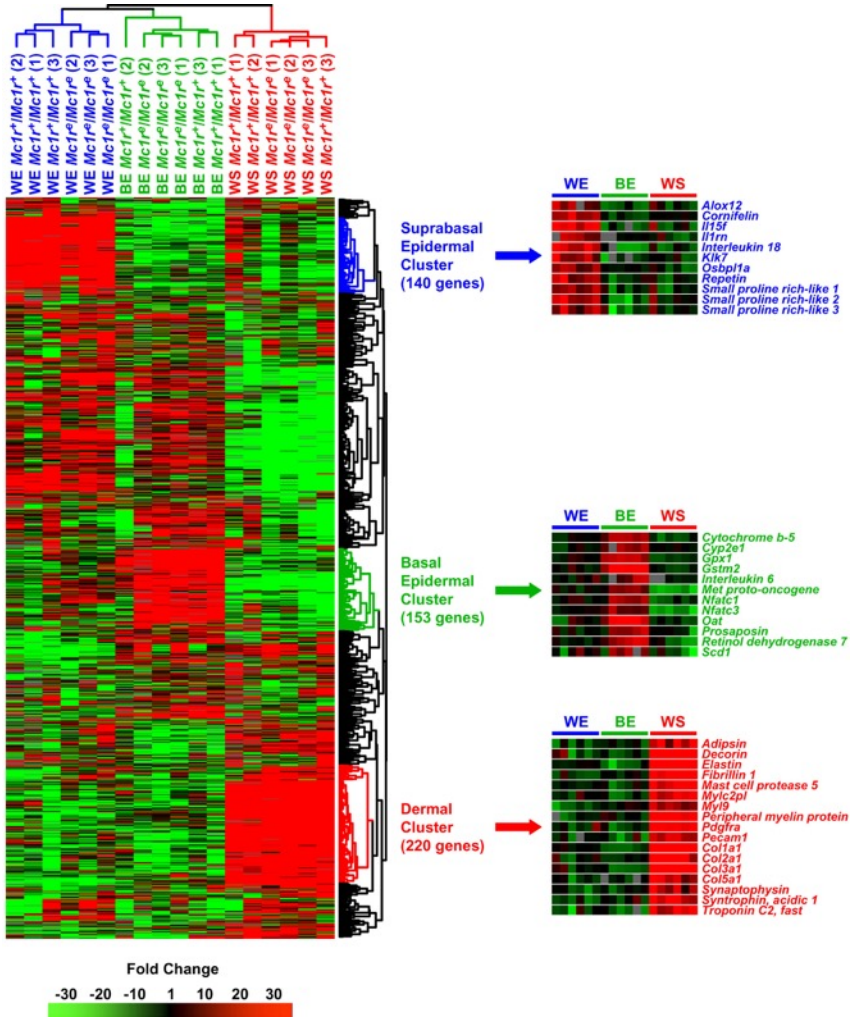
Data: X

Latent structure: Z

Clustering

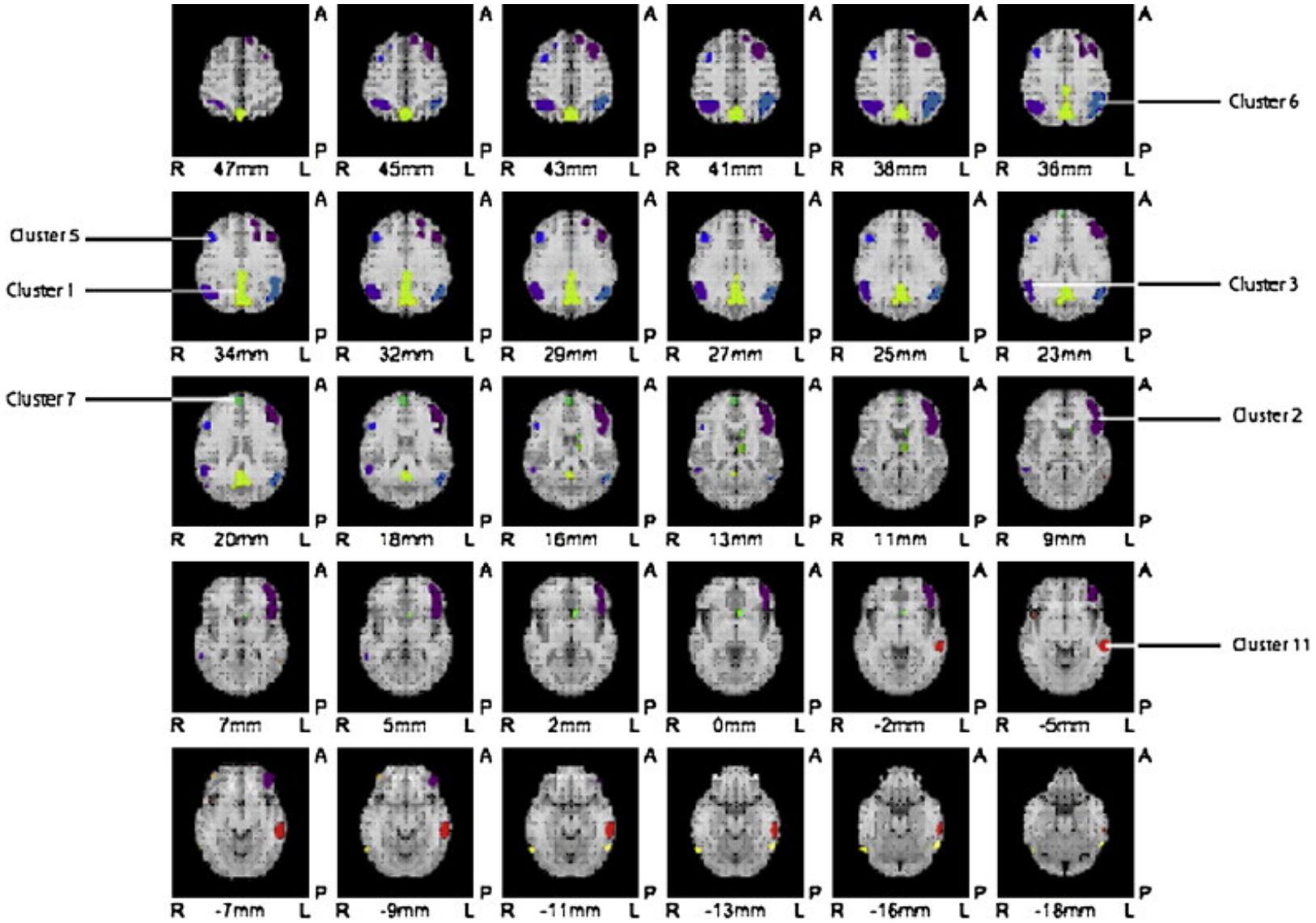
- One important unsupervised method is clustering
- Goal: Organize data in classes

Clustering applications – Microarray Gene Expression data

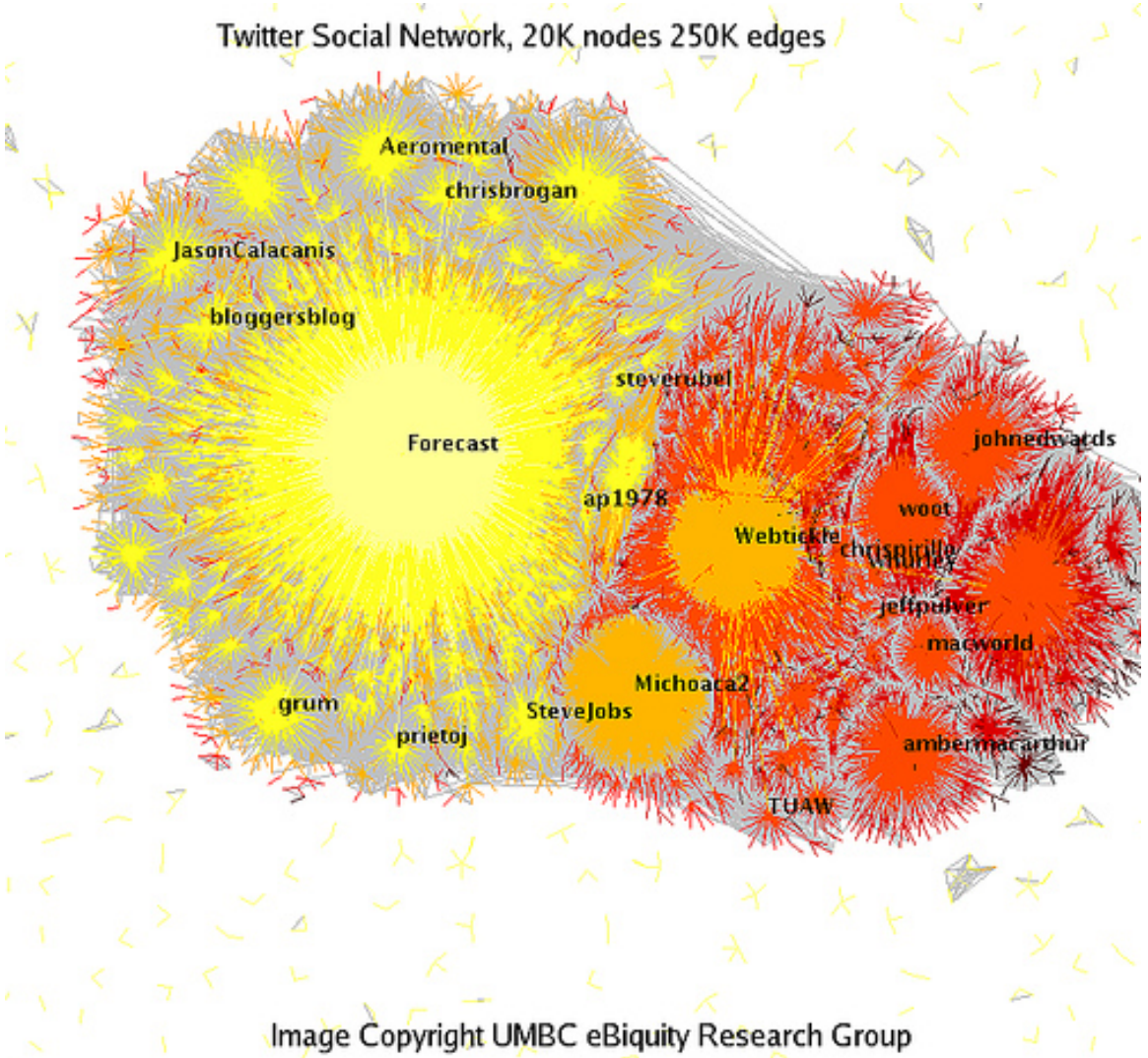


From: “Skin layer-specific transcriptional profiles in normal and recessive yellow (Mc1re/Mc1re) mice" by April and Barsh in Pigment Cell Research (2006)

Clustering applications – Medical Imaging



Clustering applications – Community detection



News Media


The New York Times


BuzzFeed


BBC
NEWS


FOX
NEWS
.com


CNN

The Washington Post


CHINA DAILY
中国日報


Mother Jones


THE
HUFFINGTON
POST


BREITBART
B


TOI


theguardian

Clustering

- One important unsupervised method is clustering
- Goal: Organize data in classes
  - Classes are hard to define
  - Different data representation may lead to different clusterings

Clustering

- One important unsupervised method is clustering
- Goal: Organize data in classes
  - Data have high in-class similarity
  - Data have low out-of-class similarity

Clustering - Similarity

We'll call $d(\mathbf{x}, \mathbf{y})$ the similarity measure of $\mathbf{x}$ and $\mathbf{y}$

Examples:

Euclidean Distance: $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$

Edit Distance: $d(\mathbf{x}, \mathbf{y}) =$ `# replace, insert, deletes to turn` $\mathbf{x}$ `into` $\mathbf{y}$

$$d(\texttt{kitten}, \texttt{sitting}) = 3$$

$$\texttt{kitten} \rightarrow \texttt{sitten} \rightarrow \texttt{sittin} \rightarrow \texttt{sitting}$$

What properties make a good similarity measure?

Clustering - Similarity

We'll call $d(\mathbf{x}, \mathbf{y})$ the similarity measure of $\mathbf{x}$ and $\mathbf{y}$
Properties:

Symmetry	$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
Self-Consistency	$d(\mathbf{x}, \mathbf{x}) = 0$
Positivity	$d(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$
Triangle Inequality	$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$

OK, so say we have a good similarity measure.
How do we find clusters in the data?

K-Means

- Simplest clustering method
- Iterative in nature
- Reasonably fast
- Very popular in practice (though with more bells and whistles)
- Requires real-valued data

K-Means

General Idea:
pick K initial cluster means
do until convergence ...

- associate examples closest to mean k with cluster k
- update cluster means with current examples in cluster k

Stop when:

- cluster assignments don't change

K-Means

General Idea:
pick K initial cluster means
do until convergence ...

- associate examples closest to mean k with cluster k
- update cluster means with current examples in cluster k

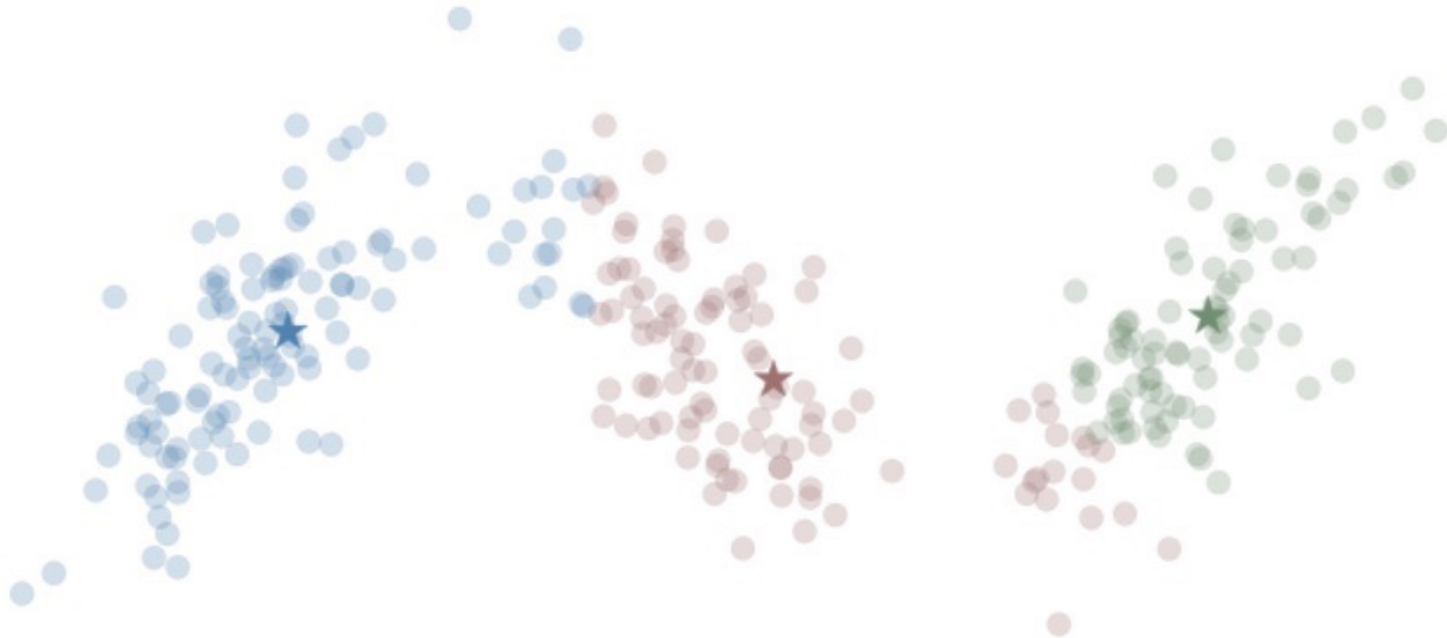
Stop when:

- cluster assignments don't change
- cluster means don't change (too much)

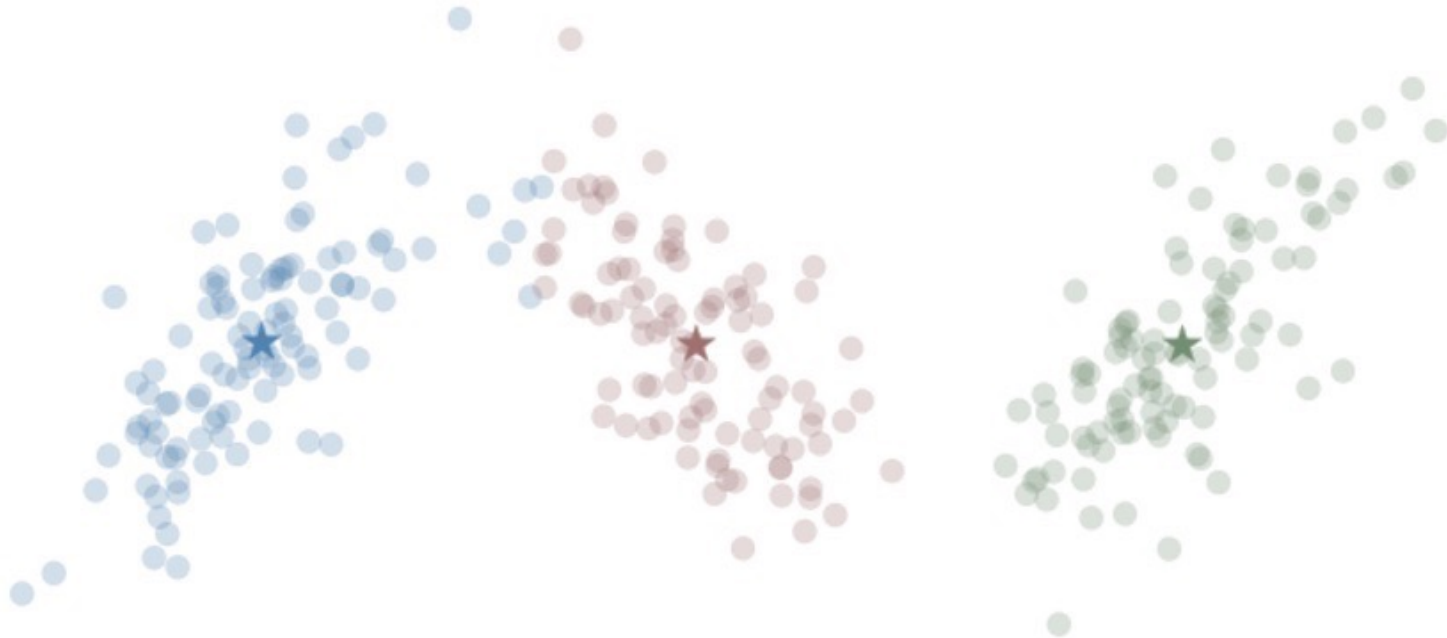
K-Means Example

K-Means Example

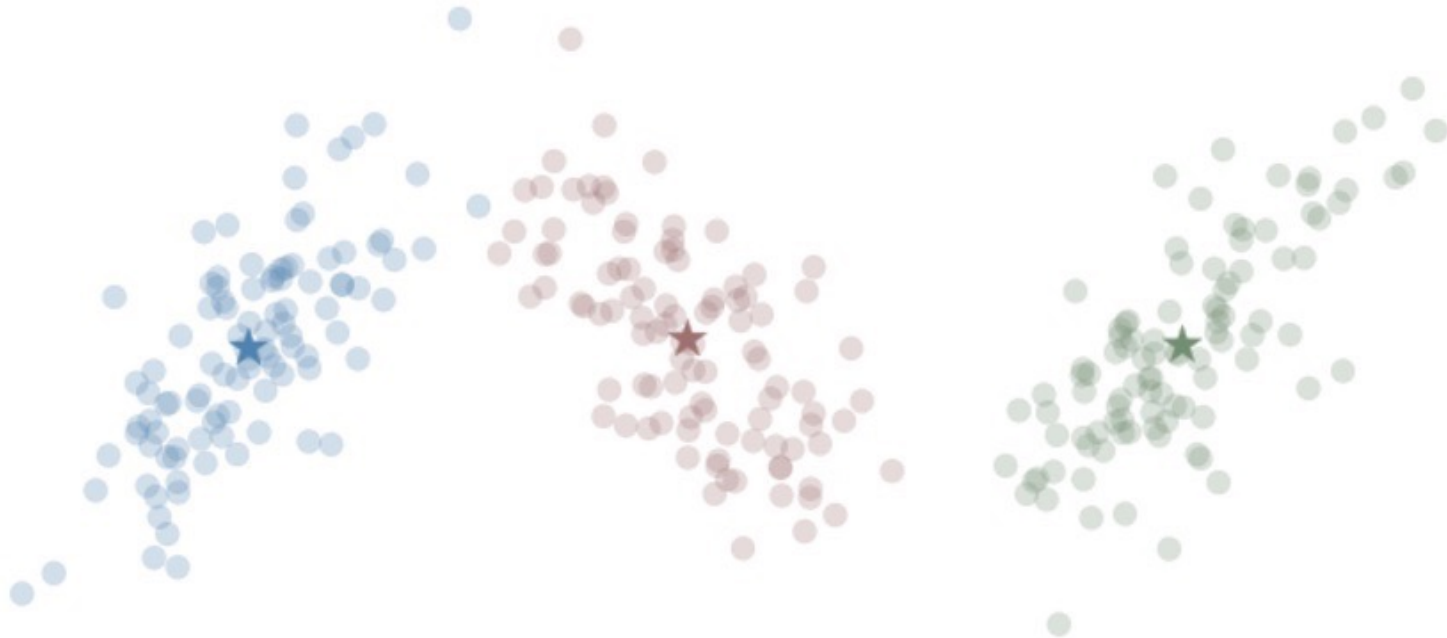
K-Means Example



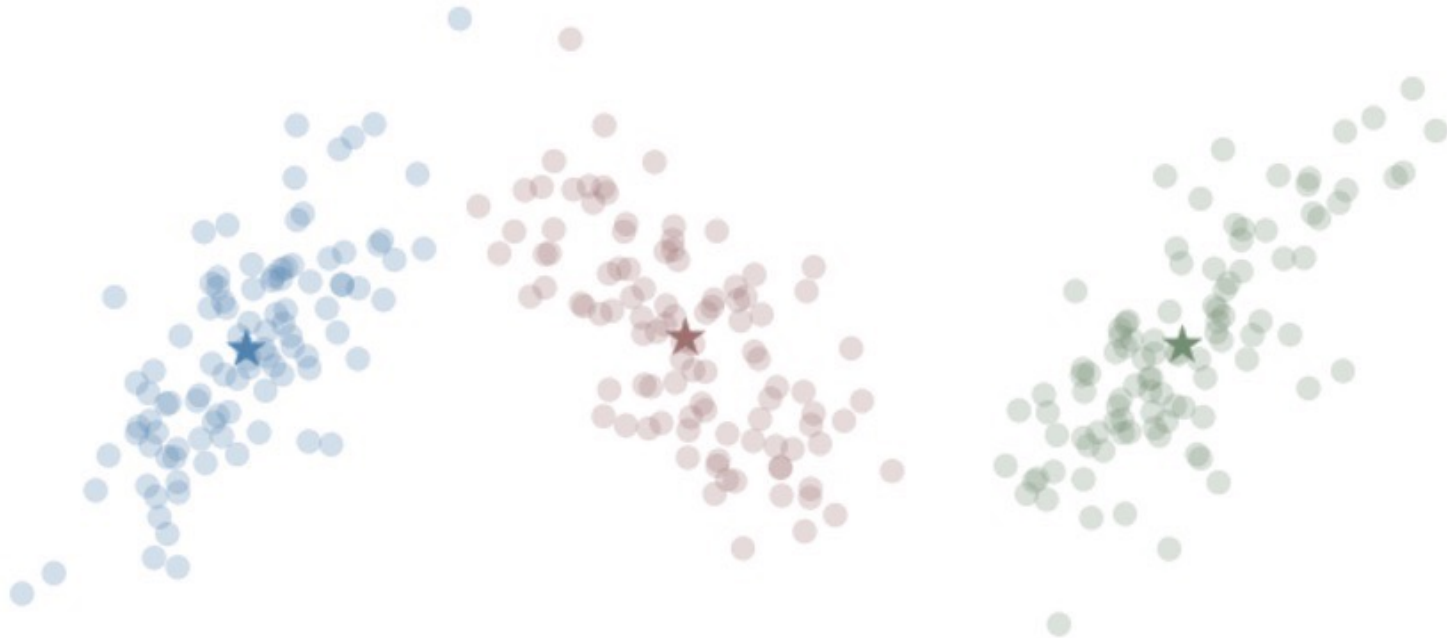
K-Means Example



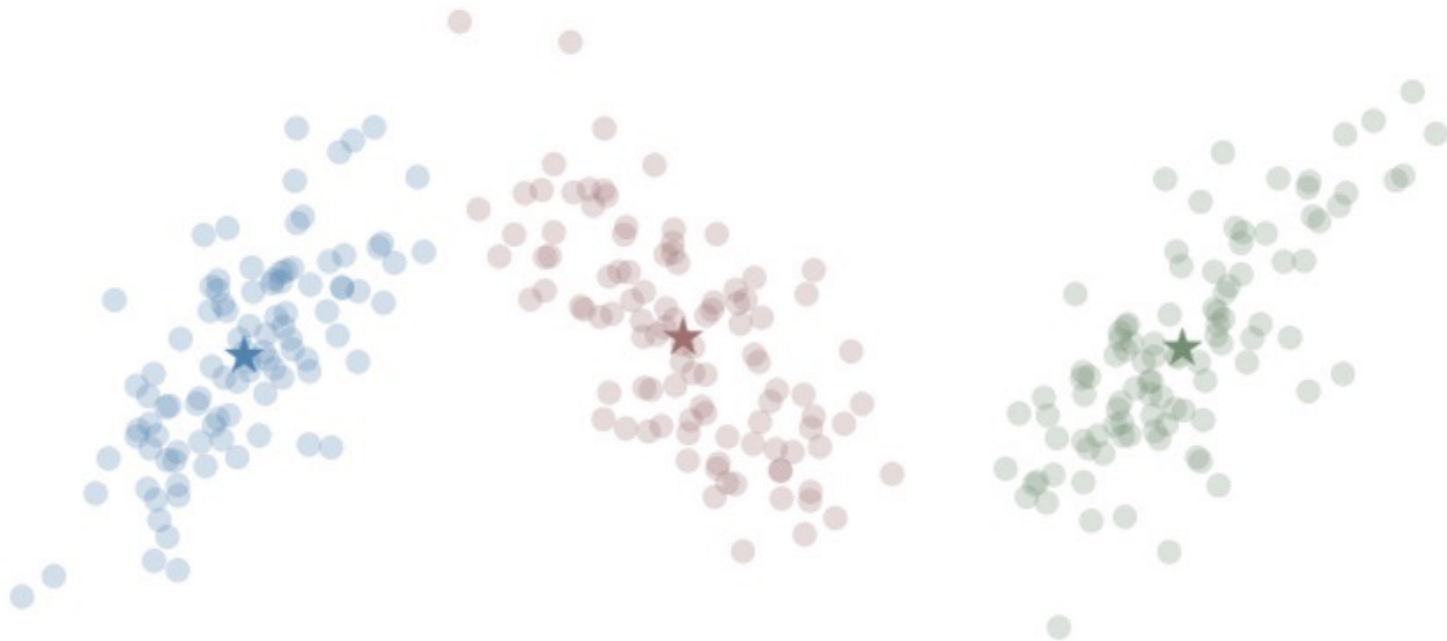
K-Means Example



K-Means Example



K-Means Example



More K-means

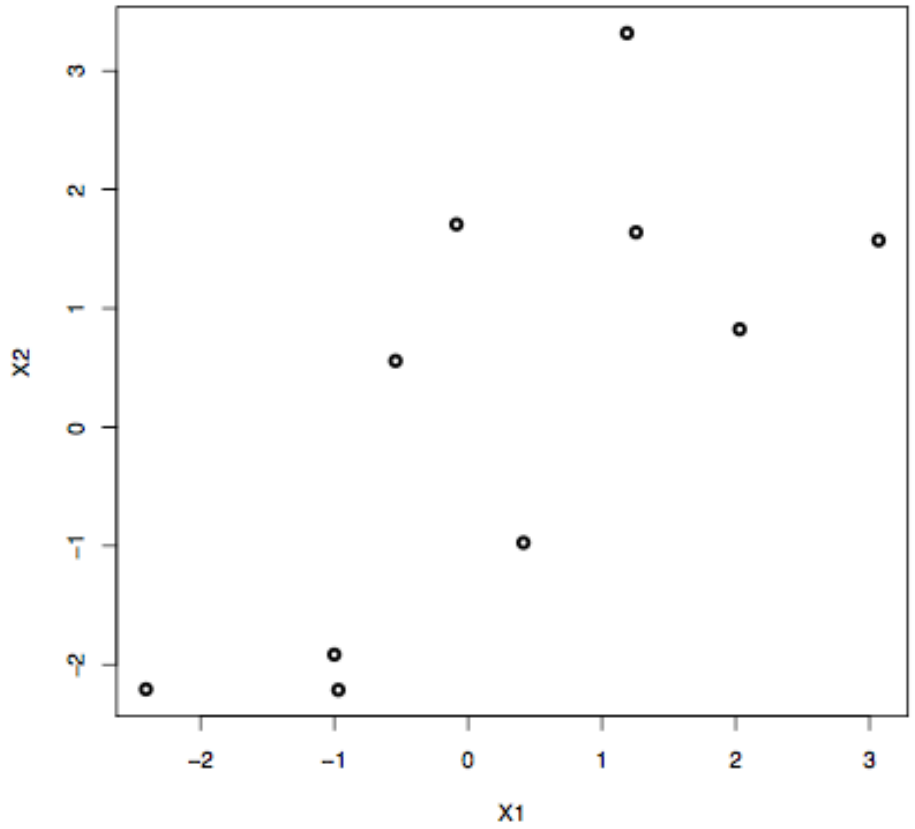
- Animations:

http://shabal.in/visuals/kmeans/4.html

K-Means in numbers

Data:

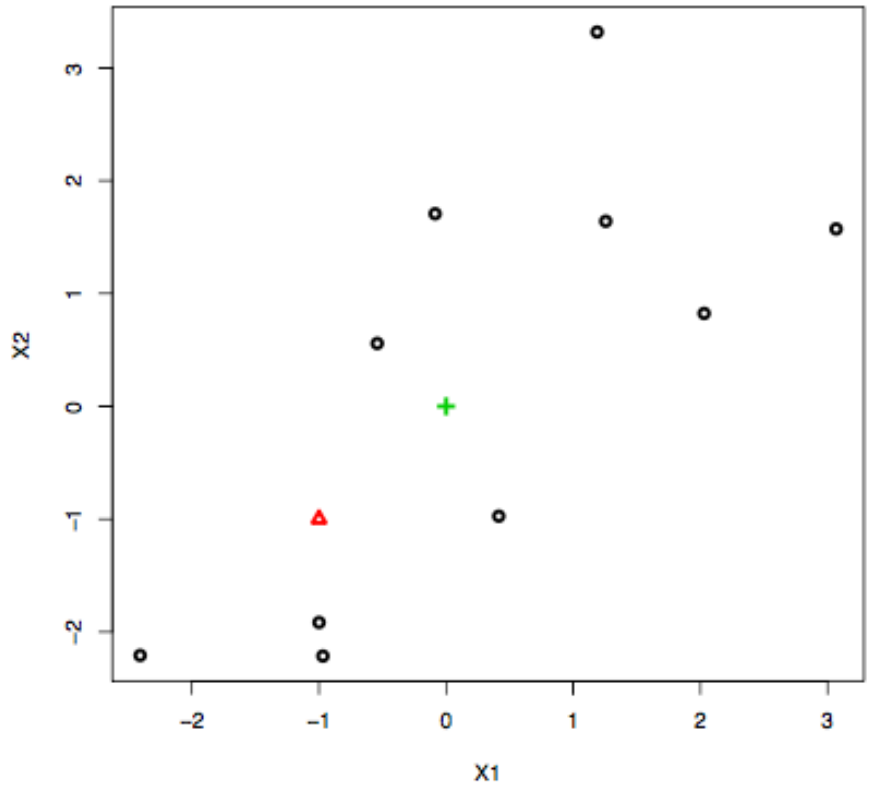
x_1	x_2
0.4	-1.0
-1.0	-2.2
-2.4	-2.2
-1.0	-1.9
-0.5	0.6
-0.1	1.7
1.2	3.3
3.1	1.6
1.3	1.6
2.0	0.8



K-Means in numbers

Pick K centers (randomly):

$$(-1, -1) \text{ and } (0, 0)$$



K-Means in numbers

Calculate distance between points and those centers:

x_1	x_2	$(-1,-1)$	$(0,0)$
0.4	-1.0	1.4	1.1
-1.0	-2.2	1.2	2.4
-2.4	-2.2	1.9	3.3
-1.0	-1.9	0.9	2.2
-0.5	0.6	1.6	0.8
-0.1	1.7	2.9	1.7
1.2	3.3	4.8	3.5
3.1	1.6	4.8	3.4
1.3	1.6	3.5	2.1
2.0	0.8	3.5	2.2

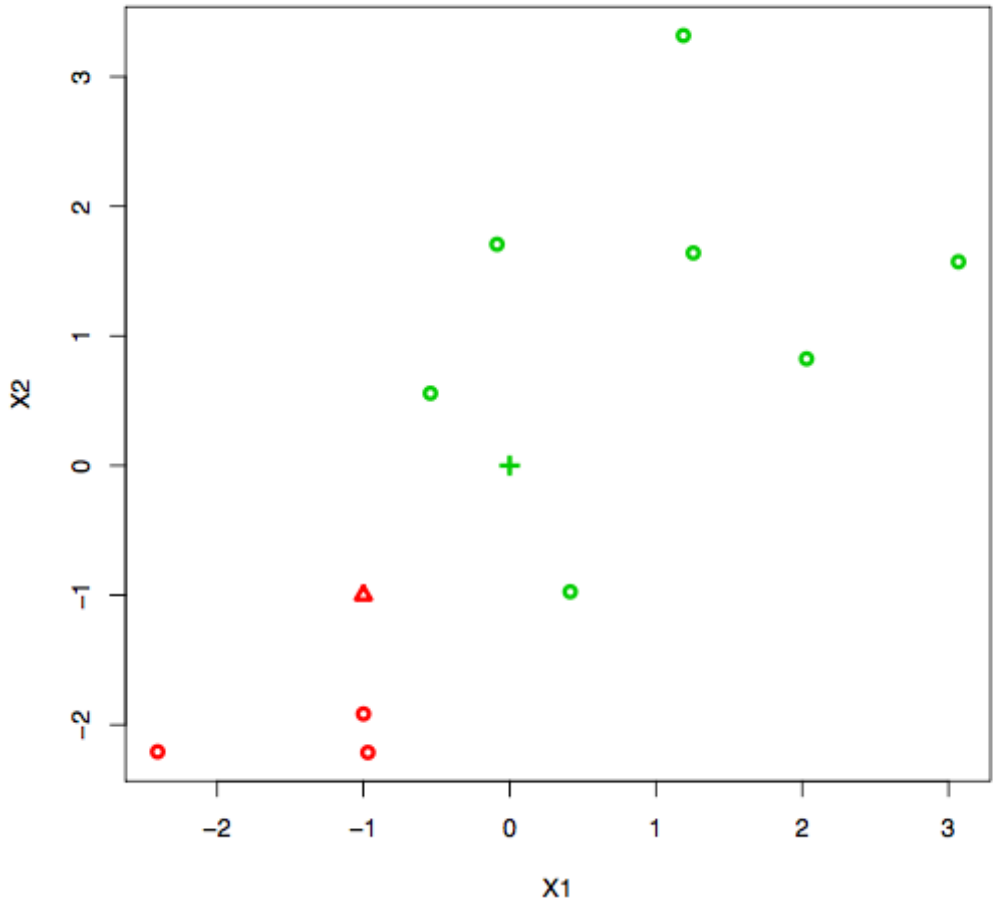
K-Means in numbers

Choose mean with smaller distance:

x_1	x_2	$(-1,-1)$	$(0,0)$
0.4	-1.0	1.4	**1.1**
-1.0	-2.2	**1.2**	2.4
-2.4	-2.2	**1.9**	3.3
-1.0	-1.9	**0.9**	2.2
-0.5	0.6	1.6	**0.8**
-0.1	1.7	2.9	**1.7**
1.2	3.3	4.8	**3.5**
3.1	1.6	4.8	**3.4**
1.3	1.6	3.5	**2.1**
2.0	0.8	3.5	**2.2**

K-Means in numbers

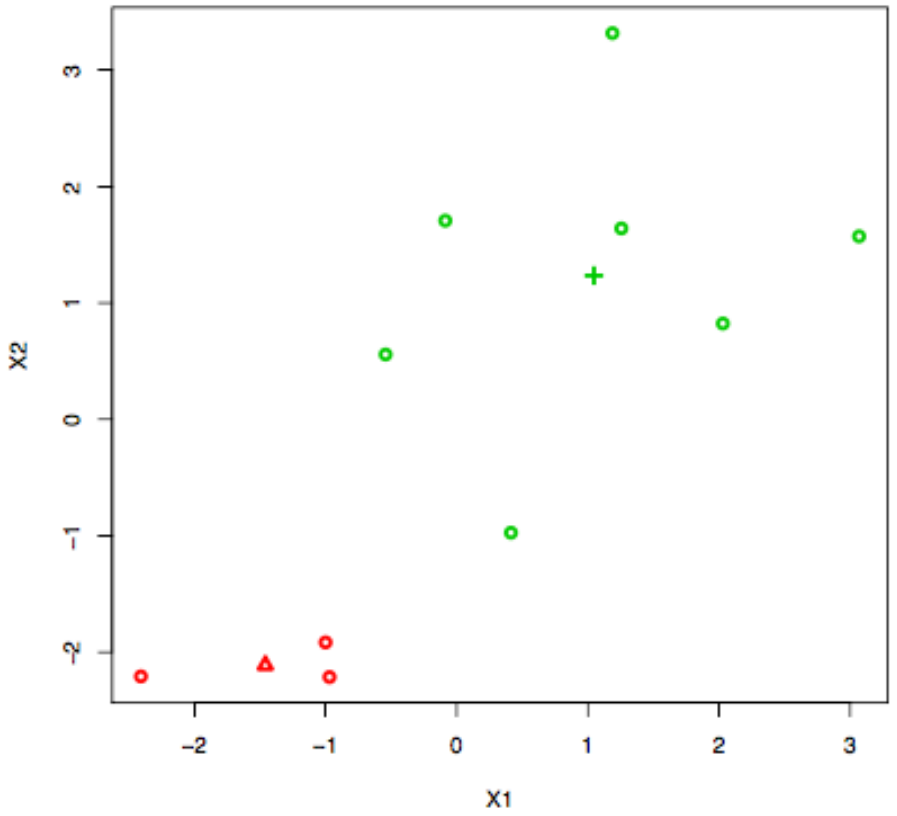
New clusters:



K-Means in numbers

Refit means for each cluster:

- cluster 1: $(-1.0, -2.2)$, $(-2.4, -2.2)$, $(-1.0, -1.9)$
- new mean: $(-1.5, -2.1)$
- cluster 2: $(0.4, -1.0)$, $(-0.5, 0.6)$, $(-0.1, 1.7)$, $(1.2, 3.3)$, $(3.1, 1.6)$, $(1.3, 1.6)$, $(2.0, 0.8)$
- new mean: $(1.0, 1.2)$



K-Means in numbers

Recalculate distances for each cluster:

x_1	x_2	$(-1.5, -2.1)$	$(1.0, 1.2)$
0.4	-1.0	2.2	2.3
-1.0	-2.2	0.5	4.0
-2.4	-2.2	1.0	4.9
-1.0	-1.9	0.5	3.8
-0.5	0.6	2.8	1.7
-0.1	1.7	4.1	1.2
1.2	3.3	6.0	2.1
3.1	1.6	5.8	2.0
1.3	1.6	4.6	0.5
2.0	0.8	4.6	1.1

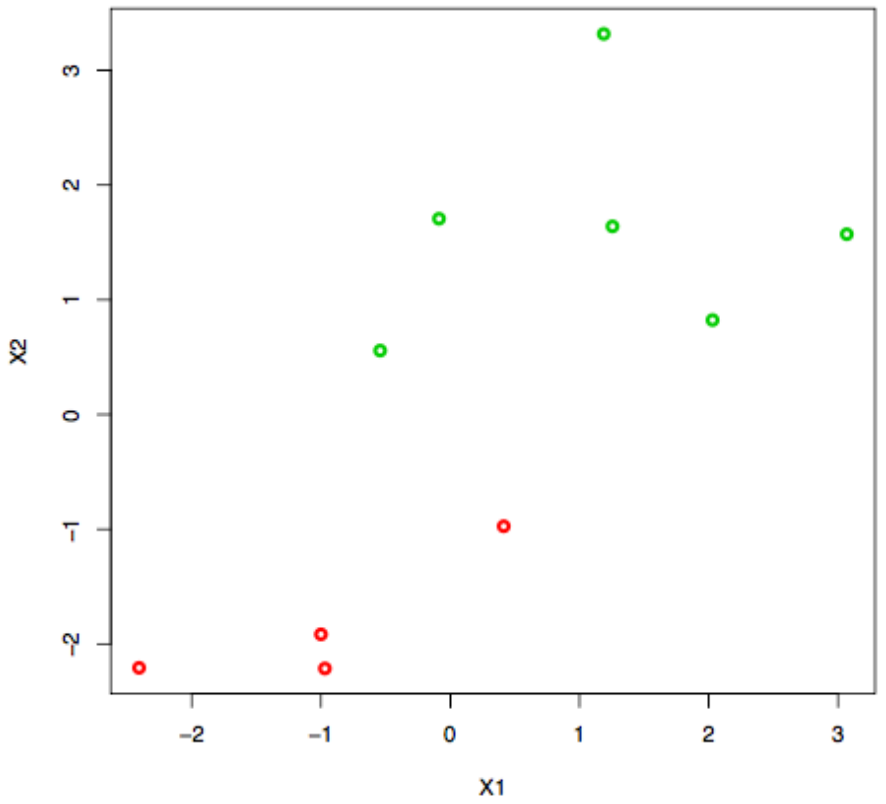
K-Means in numbers

Choose mean with smaller distance:

x_1	x_2	$(-1.5, -2.1)$	$(1.0, 1.2)$
0.4	-1.0	**2.2**	2.3
-1.0	-2.2	**0.5**	4.0
-2.4	-2.2	**1.0**	4.9
-1.0	-1.9	**0.5**	3.8
-0.5	0.6	2.8	**1.7**
-0.1	1.7	4.1	**1.2**
1.2	3.3	6.0	**2.1**
3.1	1.6	5.8	**2.0**
1.3	1.6	4.6	**0.5**
2.0	0.8	4.6	**1.1**

K-Means in numbers

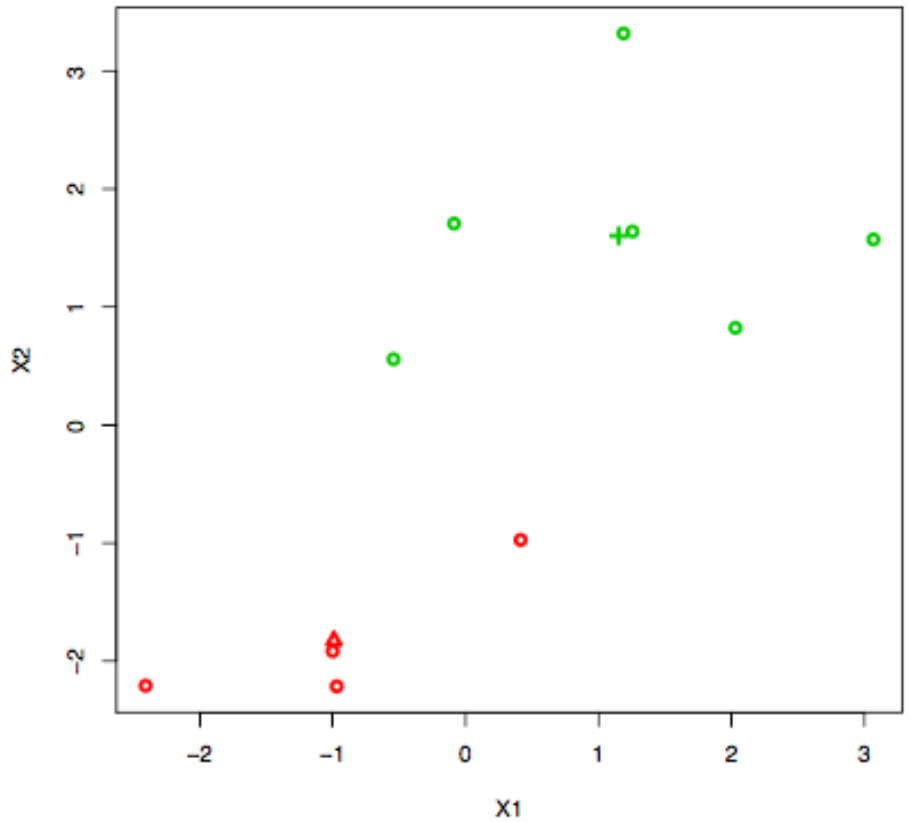
New clusters:



K-Means in numbers

Refit means for each cluster:

- cluster 1: $(0.4, -1.0)$, $(-1.0, -2.2)$, $(-2.4, -2.2)$, $(-1.0, -1.9)$
- new mean: $(-1.0, -1.8)$
- cluster 2: $(-0.5, 0.6)$, $(-0.1, 1.7)$, $(1.2, 3.3)$, $(3.1, 1.6)$, $(1.3, 1.6)$, $(2.0, 0.8)$
- new mean: $(1.2, 1.6)$



K-Means in numbers

Recalculate distances for each cluster:

x_1	x_2	$(-1.0, -1.8)$	$(1.2, 1.6)$
0.4	-1.0	1.6	2.7
-1.0	-2.2	0.4	4.4
-2.4	-2.2	1.5	5.2
-1.0	-1.9	0.1	4.1
-0.5	0.6	2.4	2.0
-0.1	1.7	3.6	1.2
1.2	3.3	5.6	1.7
3.1	1.6	5.3	1.9
1.3	1.6	4.1	0.1
2.0	0.8	4.0	1.2

K-Means in numbers

Select smallest distance and compare these clusters with previous:

Table: New Clusters

x_1	x_2	$(-1.0, -1.8)$	$(1.2, 1.6)$
0.4	-1.0	**1.6**	2.7
-1.0	-2.2	**0.4**	4.4
-2.4	-2.2	**1.5**	5.2
-1.0	-1.9	**0.1**	4.1
-0.5	0.6	2.4	**2.0**
-0.1	1.7	3.6	**1.2**
1.2	3.3	5.6	**1.7**
3.1	1.6	5.3	**1.9**
1.3	1.6	4.1	**0.1**
2.0	0.8	4.0	**1.2**

Table: Old Clusters

$(-1.5, -2.1)$	$(1.0, 1.2)$
2.2	2.3
0.5	4.0
1.0	4.9
0.5	3.8
2.8	**1.7**
4.1	**1.2**
6.0	**2.1**
5.8	**2.0**
4.6	**0.5**
4.6	**1.1**

K-Means

Strengths:

- Simple to understand
- Efficient - time complexity $\mathcal{O}(dNKT)$ for $\mathbf{x} \in \mathbb{R}^d$
- Simple to implement

K-Means

```
def KMeans(X, K, max_it=500):

    # Initialize cluster means to K samples from data
    rstart = choice(range(X.shape[0]), size=(K), replace=False)
    mu, muold = X[rstart,:], 1000*np.ones(X[rstart,:].shape)

    its = 0
    while its < max_it and np.linalg.norm(mu-muold) > 1e-4 :

        # compute distance b/w each point and centroid
        dist = np.array([[np.linalg.norm(x-m) for m in mu] for x in X]

        # compute new cluster assignments
        z = np.array([np.argmin(d) for d in dist])

        # move centroids
        muold = mu
        mu = np.array([np.mean(X[z==k, :], axis=0) for k in range(K)])
        its += 1

    return mu, z
```

K-Means

- **Weaknesses**
  - Doesn't really work with categorical data
  - Usually only converges to local minimum
  - Have to determine number of clusters
  - Can be sensitive to outliers
  - Only generates convex clusters

K-Means - Weaknesses

- Doesn't really work with categorical data

K-Means - Weaknesses

- Doesn't really work with categorical data
- **Fix**: Do K-Modes instead

K-Means - Weaknesses

- Usually only converges to local minimum

K-Means - Weaknesses

- Usually only converges to local minimum
- **Fix**: Do several runs with random inits. and choose best

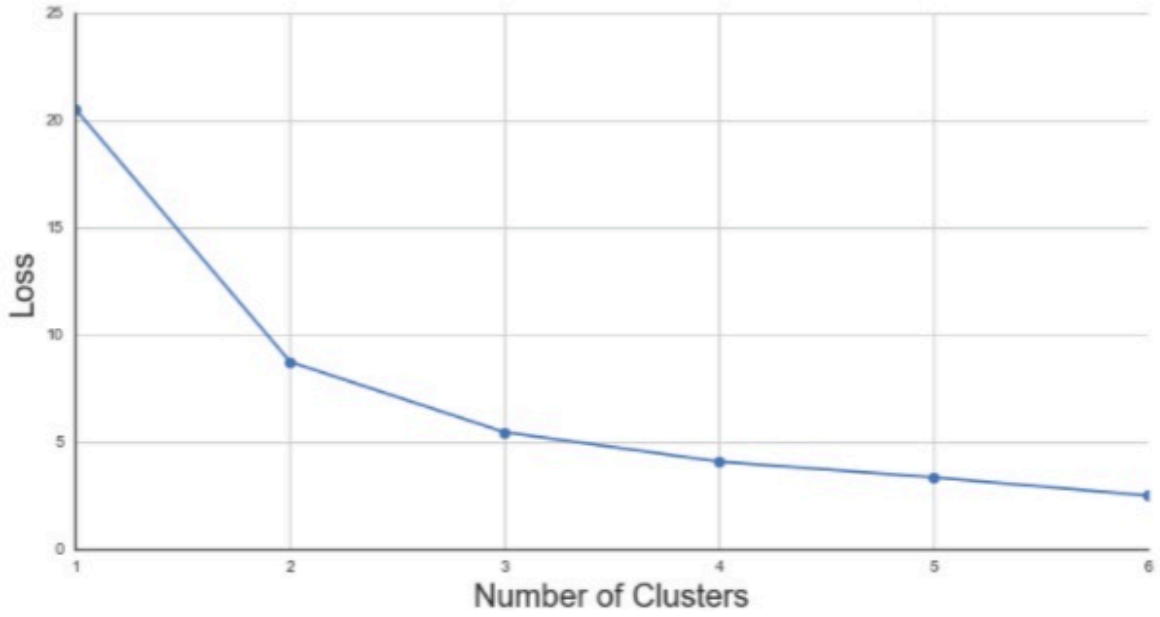
K-Means - Weaknesses

- Have to determine number of clusters

K-Means - Weaknesses

- Have to determine number of clusters
- Fix: Use the elbow method

Run K-Means for different values of k and look at loss function



Weaknesses - Outlier Sensitivity

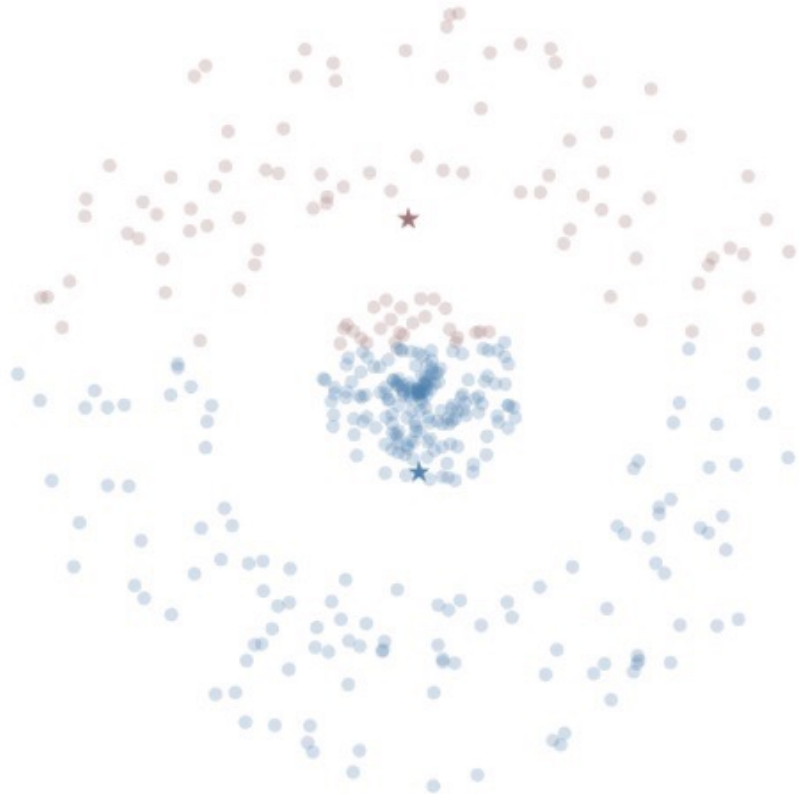
Weaknesses - Outlier Sensitivity

Weaknesses - Outlier Sensitivity

Weaknesses - Outlier Sensitivity

Weaknesses - Convex Clusters

Weaknesses - Convex Clusters



Gaussian Mixture Models

Guassian Mixture Models (or GMMs) are a probabilistic generalization of K-means.

In K-means we made **hard** cluster assignments.

That is, we said $\mathbf{x}_i$ definitely belongs to cluster k.

GMM utilizes **soft** cluster assignments.

That is, we'll say $\mathbf{x}_i$ belongs to cluster $k = \{1, \ldots, K\}$ with some probability.

We can then estimate that probability for all k and, if need be, assign $\mathbf{x}_i$ to the cluster with the highest probability.

Gaussian Mixture Models

The motivation behind GMMs is a generative one

Gaussian Mixture Models

The motivation behind GMMs is a generative one
We'll impose on the data a distribution of the form

$$p(\mathbf{x}_i, z_i) = p(\mathbf{x}_i \mid z_i)\, p(z_i)$$

where here z_i is the cluster that $\mathbf{x}_i$ belongs to (though, keep in mind that z_i is a random variable taking on all values in $\{1, \ldots, K\}$
We'll assume:
z_i is multinomial (think rolling a die)
$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ (given a k, $\mathbf{x}_i$ is Multivariate Gaussian)

Gaussian Mixture Models

$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$

μ_k is a mean vector (just like in K-Means)

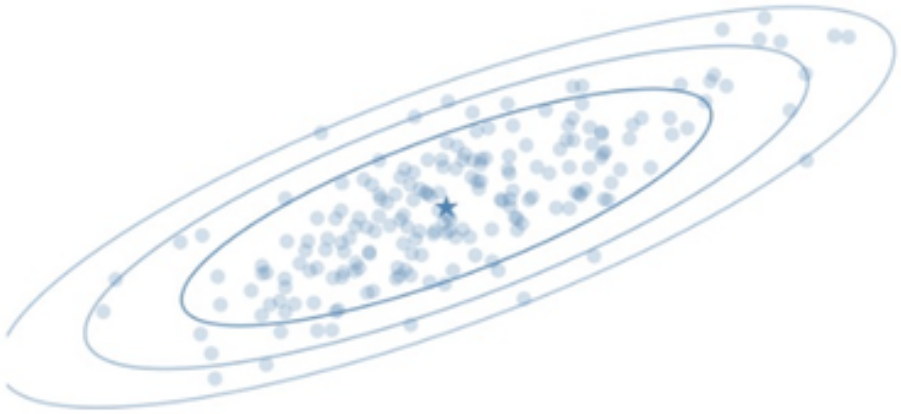
Σ_k is a covariance matrix

Gaussian Mixture Models

$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$

μ_k is a mean vector (just like in K-Means)

Σ_k is a covariance matrix



Gaussian Mixture Models

$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$

μ_k is a mean vector (just like in K-Means)

Σ_k is a covariance matrix

Density function for $\mathbf{x} \in \mathbb{R}^n$ and cluster k is given by

$$p(\mathbf{x} \mid z_i = k) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1}(\mathbf{x} - \mu_k)\right\}$$

Gaussian Mixture Models

Can generate data from model by marginalizing over k

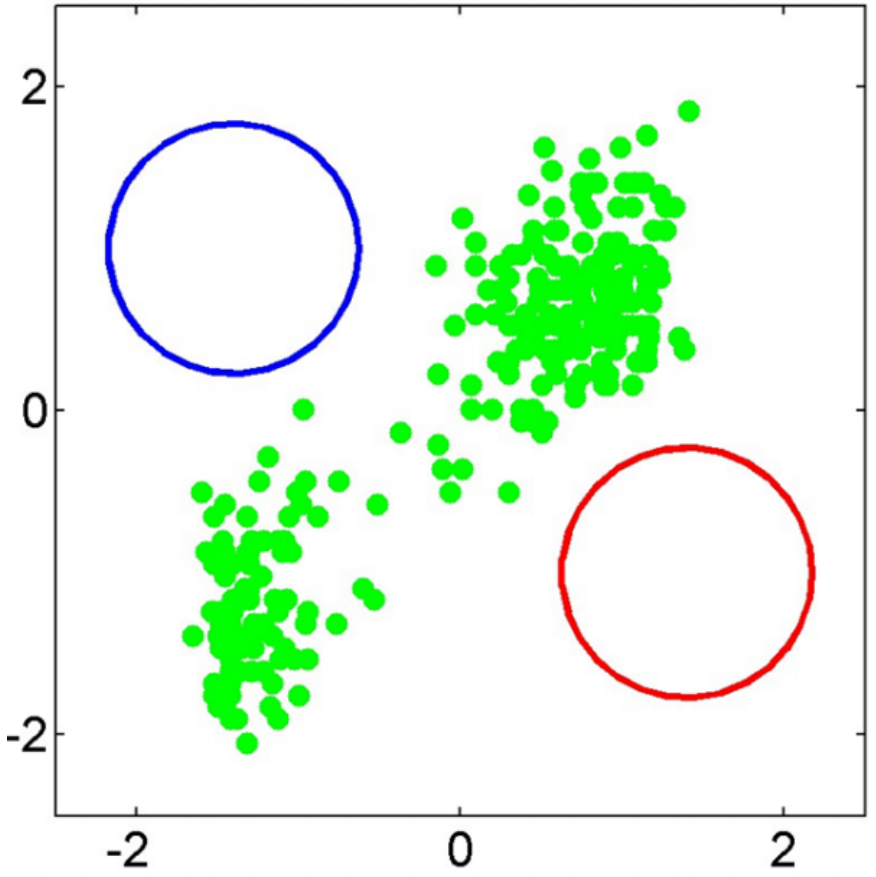
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, z = k) = \sum_{k=1}^{K} p(\mathbf{x} \mid z = k) p(z = k)$$

Recap

- K-Means is the most commonly used clustering algorithm
- We learned the Gaussian Mixture Model's generative story
  - We will learn EM-algorithm next week
  - But here's a quick preview (time permitting)

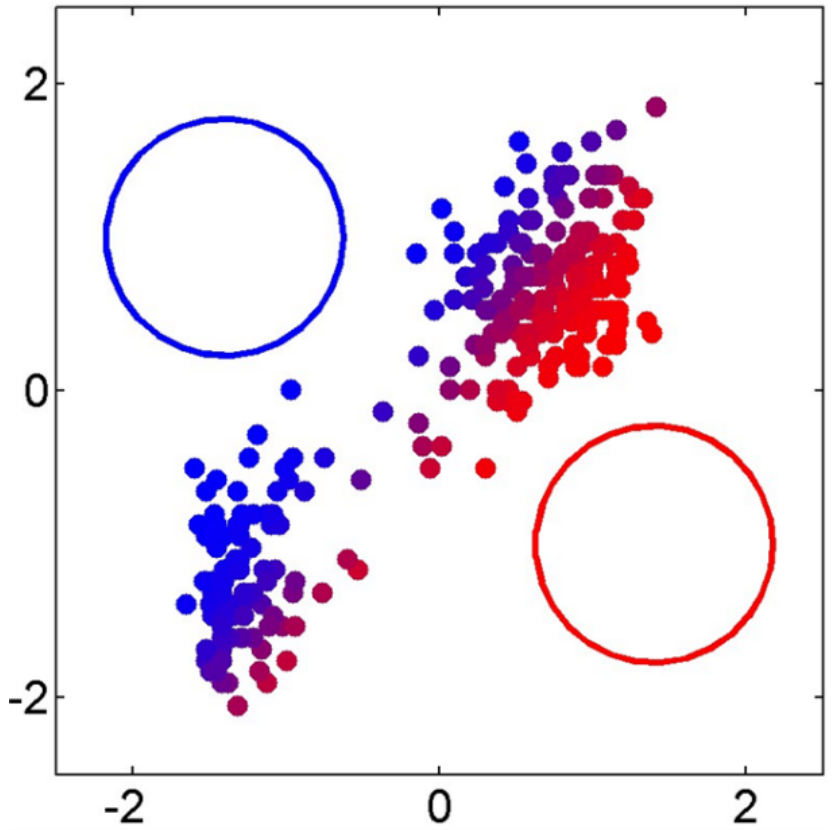
Expectation Maximization: an Intuitive Introduction

- Variables:
  - Data:
    - Data points **X** (known)
    - Cluster assignments **Z** (unknown)
  - Model:
    - Cluster assignment priors **π** (unknown)
    - Cluster means **μ** (unknown)
    - Cluster covariate matrices **Σ** (unknown)



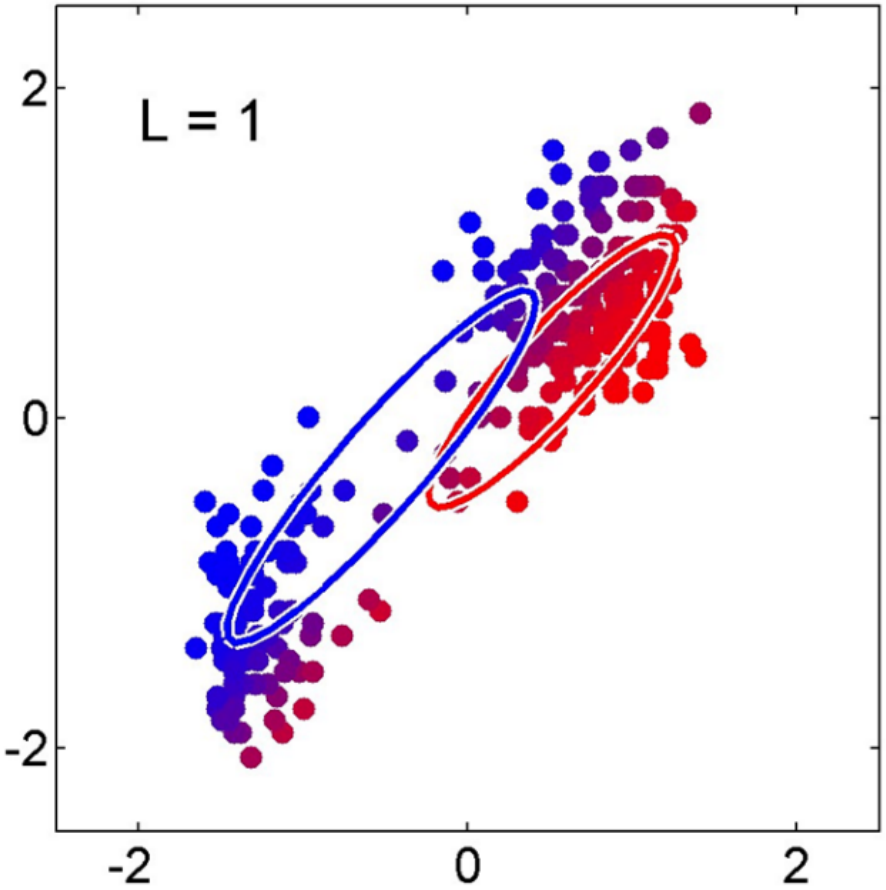
Expectation

- Given the data points **X** and an estimate of the model parameters **π, μ, Σ,** it is easy to calculate **expected** cluster assignments **z**



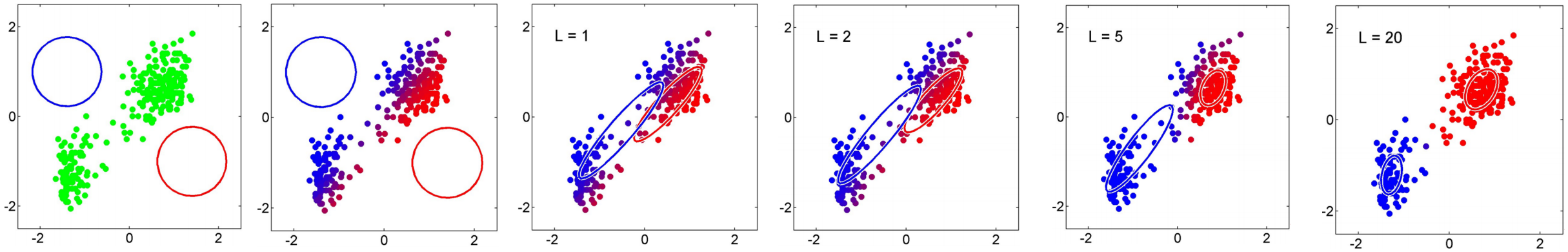
Maximization

- Given the data points **X** and an estimate of the cluster assignments **z**, we can calculate the values of **π, μ,** and **Σ** that would **maximize** the likelihood of those clusters



EM algorithm

- Keep iterating between calculating the expected cluster assignments and finding the maximum likelihood model parameters until all variables converge
- Should seem familiar…



EM algorithm

- Black magic
- All over the place-shows up pretty much any time you want to do probabilistic modeling
- Proof boils down to “it can’t hurt”