Machine Learning Department
School of Computer Science
Carnegie Mellon University

K-Means
+
Principal Component Analysis (PCA)

Matt Gormley & Henry Chai
Lecture 25
Apr. 16, 2025

Reminders

- Homework 8: Deep RL
  - Out: Tue, Apr. 8
  - Due: Wed, Apr. 16 at 11:59pm
- Homework 9: Learning Paradigms
  - Out: Wed, Apr. 16
  - Due: Thu, Apr. 24 at 11:59pm

CLUSTERING

Clustering

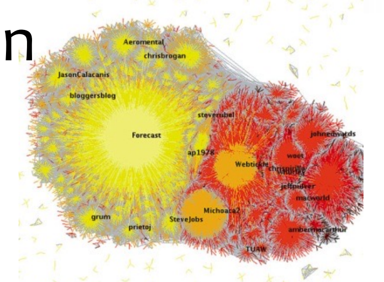
Motivation

- **Goal:** automatically partition **unlabeled** data into groups of similar points
- **Algorithms:**
  - hierarchical agglomerative clustering
  - top-down divisive clustering
  - Gaussian mixture models
  - spectral clustering
  - **K-Means**

Applications

- **Topic modeling:** Cluster news articles or web pages or search results by topic.
- **Gene expression analysis:** Cluster protein sequences by function or genes according to expression profile.
- **Community detection:** Cluster users of social networks by interest (community detection).
- **Fraud detection:** Spot unusual transaction clusters for fraud detection.
- **Sloan Digital Sky Survey analysis:** Group galaxies or nearby stars

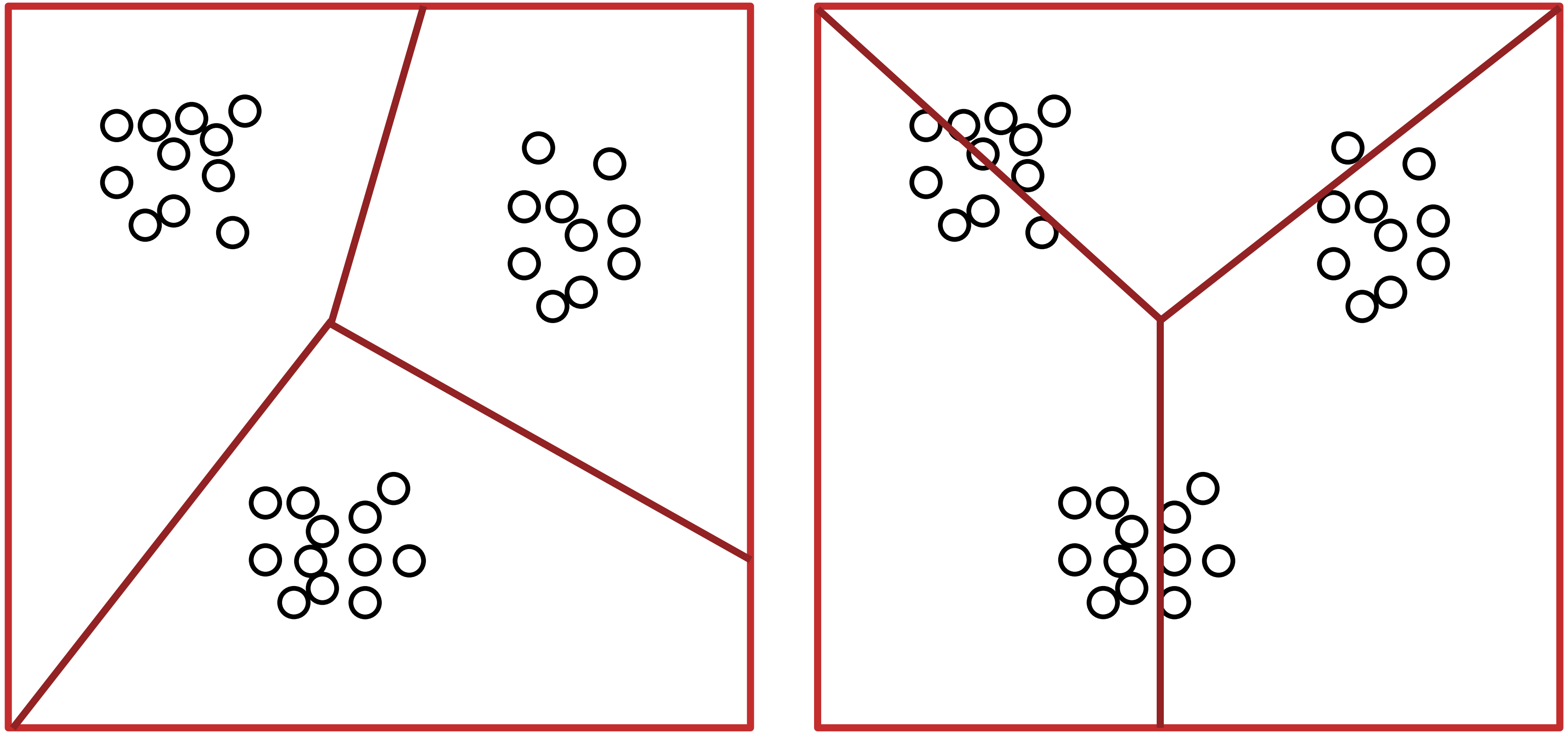




Slide adapted from Nina Balcan

Clustering

Question: Which of these partitions is "better"?



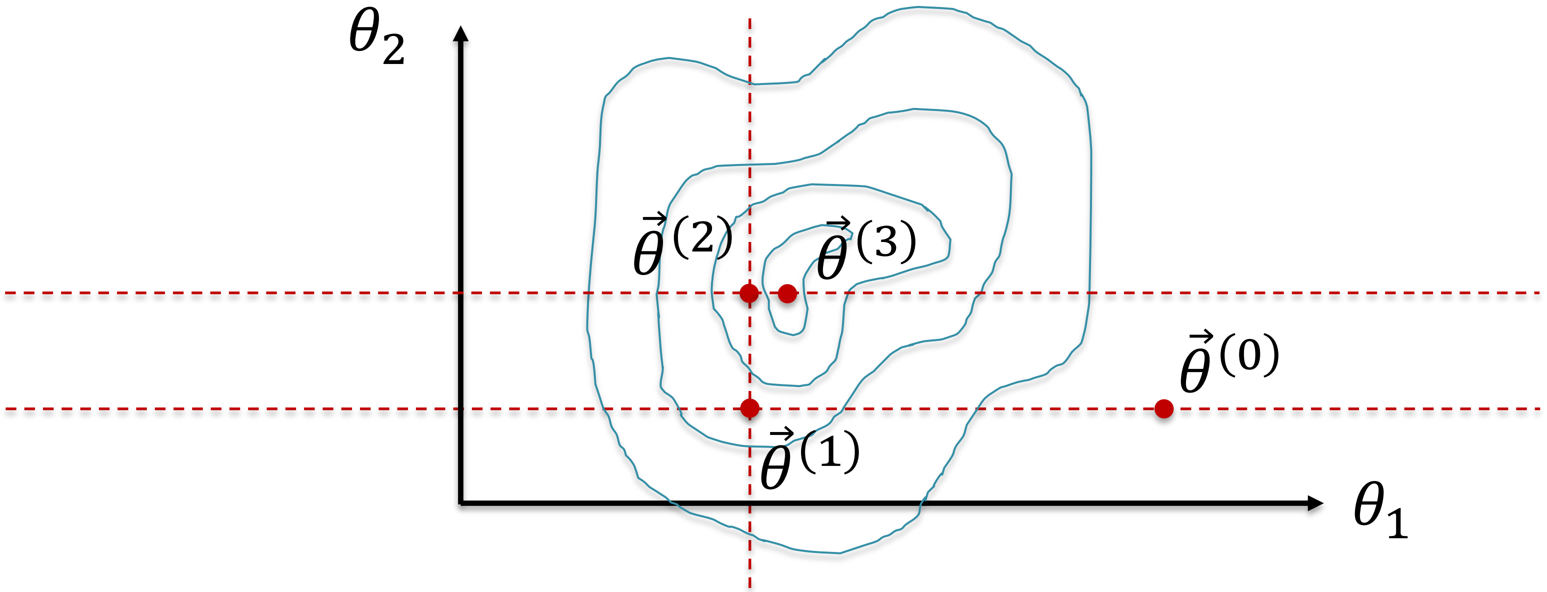
OPTIMIZATION BACKGROUND

Coordinate Descent

- Goal: minimize some objective

$$\vec{\theta}^* = \underset{\vec{\theta}}{\operatorname{argmin}}\, J(\vec{\theta})$$

- Idea: iteratively pick one variable and minimize the objective w.r.t. just that one variable, *keeping all the others fixed*.



Block Coordinate Descent

- Goal: minimize some objective (with 2 blocks)

$$\vec{\alpha}^*, \vec{\beta}^* = \operatorname*{argmin}_{\vec{\alpha}, \vec{\beta}} J(\vec{\alpha}, \vec{\beta})$$

- Idea: iteratively pick one *block* of variables ($\vec{\alpha}$ or $\vec{\beta}$) and minimize the objective w.r.t. that block, keeping the other(s) fixed.

while not converged:

$$\vec{\alpha} = \operatorname*{argmin}_{\vec{\alpha}} J(\vec{\alpha}, \vec{\beta})$$

$$\vec{\beta} = \operatorname*{argmin}_{\vec{\beta}} J(\vec{\alpha}, \vec{\beta})$$

K-MEANS

K-Means Algorithm (Derivation)

Recipe for K-Means Derivation:

1) Define a Model.
2) Choose an objective function.
3) Optimize it!

K-Means Algorithm (Derivation)

- Input: unlabeled data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \; \mathbf{x}^{(i)} \in \mathbb{R}^M$
- Goal: Find an assignment of points to clusters
- Model Paramters:
    - cluster centers: $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K], \; \mathbf{c}_j \in \mathbb{R}^M$
    - cluster assignments: $\mathbf{z} = [z^{(1)}, z^{(2)}, \ldots, z^{(N)}], \; z^{(i)} \in \{1, \ldots, K\}$
- Decision Rule: assign each point $\mathbf{x}^{(i)}$ to its nearest cluster center $\mathbf{c}_j$

K-Means Algorithm (Derivation)

- Input: unlabeled data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N},\ \mathbf{x}^{(i)} \in \mathbb{R}^M$
- Goal: Find an assignment of points to clusters
- Model Paramters:
  - cluster centers: $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K],\ \mathbf{c}_j \in \mathbb{R}^M$
  - cluster assignments: $\mathbf{z} = [z^{(1)}, z^{(2)}, \ldots, z^{(N)}],\ z^{(i)} \in \{1, \ldots, K\}$
- Decision Rule: assign each point $\mathbf{x}^{(i)}$ to its nearest cluster center $\mathbf{c}_j$
- Objective:

$$\hat{\mathbf{C}} = \operatorname*{argmin}_{\mathbf{C}} \sum_{i=1}^{N} \boxed{\min_j ||\mathbf{x}^{(i)} - \mathbf{c}_j||_2^2}$$

Poll Question 1: In English, what is this quantity?

Answer:

K-Means Algorithm (Derivation)

- Input: unlabeled data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \;\; \mathbf{x}^{(i)} \in \mathbb{R}^M$
- Goal: Find an assignment of points to clusters
- Model Paramters:
  - cluster centers: $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K], \;\; \mathbf{c}_j \in \mathbb{R}^M$
  - cluster assignments: $\mathbf{z} = [z^{(1)}, z^{(2)}, \ldots, z^{(N)}], \;\; z^{(i)} \in \{1, \ldots, K\}$
- Decision Rule: assign each point $\mathbf{x}^{(i)}$ to its nearest cluster center $\mathbf{c}_j$
- Objective:

$$\begin{aligned}
\hat{\mathbf{C}} &= \operatorname*{argmin}_{\mathbf{C}} \sum_{i=1}^{N} \min_{j} ||\mathbf{x}^{(i)} - \mathbf{c}_j||_2^2 \\
&= \operatorname*{argmin}_{\mathbf{C}} \sum_{i=1}^{N} \min_{z^{(i)}} ||\mathbf{x}^{(i)} - \mathbf{c}_{z^{(i)}}||_2^2
\end{aligned}$$

K-Means Algorithm (Derivation)

- Input: unlabeled data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N, \;\; \mathbf{x}^{(i)} \in \mathbb{R}^M$
- Goal: Find an assignment of points to clusters
- Model Paramters:
  - cluster centers: $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K], \;\; \mathbf{c}_j \in \mathbb{R}^M$
  - cluster assignments: $\mathbf{z} = [z^{(1)}, z^{(2)}, \ldots, z^{(N)}], \;\; z^{(i)} \in \{1, \ldots, K\}$
- Decision Rule: assign each point $\mathbf{x}^{(i)}$ to its nearest cluster center $\mathbf{c}_j$
- Objective:

$$\hat{\mathbf{C}} = \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{j} ||\mathbf{x}^{(i)} - \mathbf{c}_j||_2^2$$

$$= \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{z^{(i)}} ||\mathbf{x}^{(i)} - \mathbf{c}_{z^{(i)}}||_2^2$$

$$\hat{\mathbf{C}}, \hat{\mathbf{z}} = \underset{\mathbf{C}, \mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{N} ||\mathbf{x}^{(i)} - \mathbf{c}_{z^{(i)}}||_2^2$$

K-Means Algorithm (Derivation)

- Input: unlabeled data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}, \ \mathbf{x}^{(i)} \in \mathbb{R}^M$
- Goal: Find an assignment of points to clusters
- Model Paramters:
  - cluster centers: $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K], \ \mathbf{c}_j \in \mathbb{R}^M$
  - cluster assignments: $\mathbf{z} = [z^{(1)}, z^{(2)}, \dots, z^{(N)}], \ z^{(i)} \in \{1, \dots, K\}$
- Decision Rule: assign each point $\mathbf{x}^{(i)}$ to its nearest cluster center $\mathbf{c}_j$
- Objective:

$$\hat{\mathbf{C}} = \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{i=1}^{N} \min_j ||\mathbf{x}^{(i)} - \mathbf{c}_j||_2^2$$

$$= \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{z^{(i)}} ||\mathbf{x}^{(i)} - \mathbf{c}_{z^{(i)}}||_2^2$$

$$\hat{\mathbf{C}}, \hat{\mathbf{z}} = \underset{\mathbf{C}, \mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{N} ||\mathbf{x}^{(i)} - \mathbf{c}_{z^{(i)}}||_2^2$$

$$= \boxed{\underset{\mathbf{C}, \mathbf{z}}{\operatorname{argmin}} J(\mathbf{C}, \mathbf{z})}$$

Now apply
Block Coordinate Descent!

K-Means Algorithm

1) **Given** unlabeled feature vectors
   $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$
2) **Initialize** cluster centers $c = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$
3) **Repeat** until convergence:
   a) $\mathbf{z} \leftarrow \text{argmin}_{\mathbf{z}}\, J(\mathbf{C}, \mathbf{z})$
      (pick each *cluster assignment* to minimize distance)
   b) $\mathbf{C} \leftarrow \text{argmin}_{\mathbf{C}}\, J(\mathbf{C}, \mathbf{z})$
      (pick each *cluster center* to minimize distance)

This is an application of
Block Coordinate Descent!
The only remaining step is to figure out
what the argmins boil down to…

K-Means Algorithm

1) **Given** unlabeled feature vectors
   $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$
2) **Initialize** cluster centers $c = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$
3) **Repeat** until convergence:
   a) for i in {1,…, N}
      $z^{(i)} \leftarrow \textbf{argmin}_j \, (\| \mathbf{x}^{(i)} - \mathbf{c}_j \|_2)^2$
   b) for j in {1,…,K}
      $\mathbf{c}_j \leftarrow \underset{\mathbf{c}_j}{\textbf{argmin}} \sum_{i:z^{(i)} = j} (\| \mathbf{x}^{(i)} - \mathbf{c}_j \|_2)^2$

The minimization over cluster assignments decomposes, so that we can find each $z^{(i)}$ independently of the others

Likewise, the minimization over cluster centers decomposes, so we can find each $\mathbf{c}_j$ independently

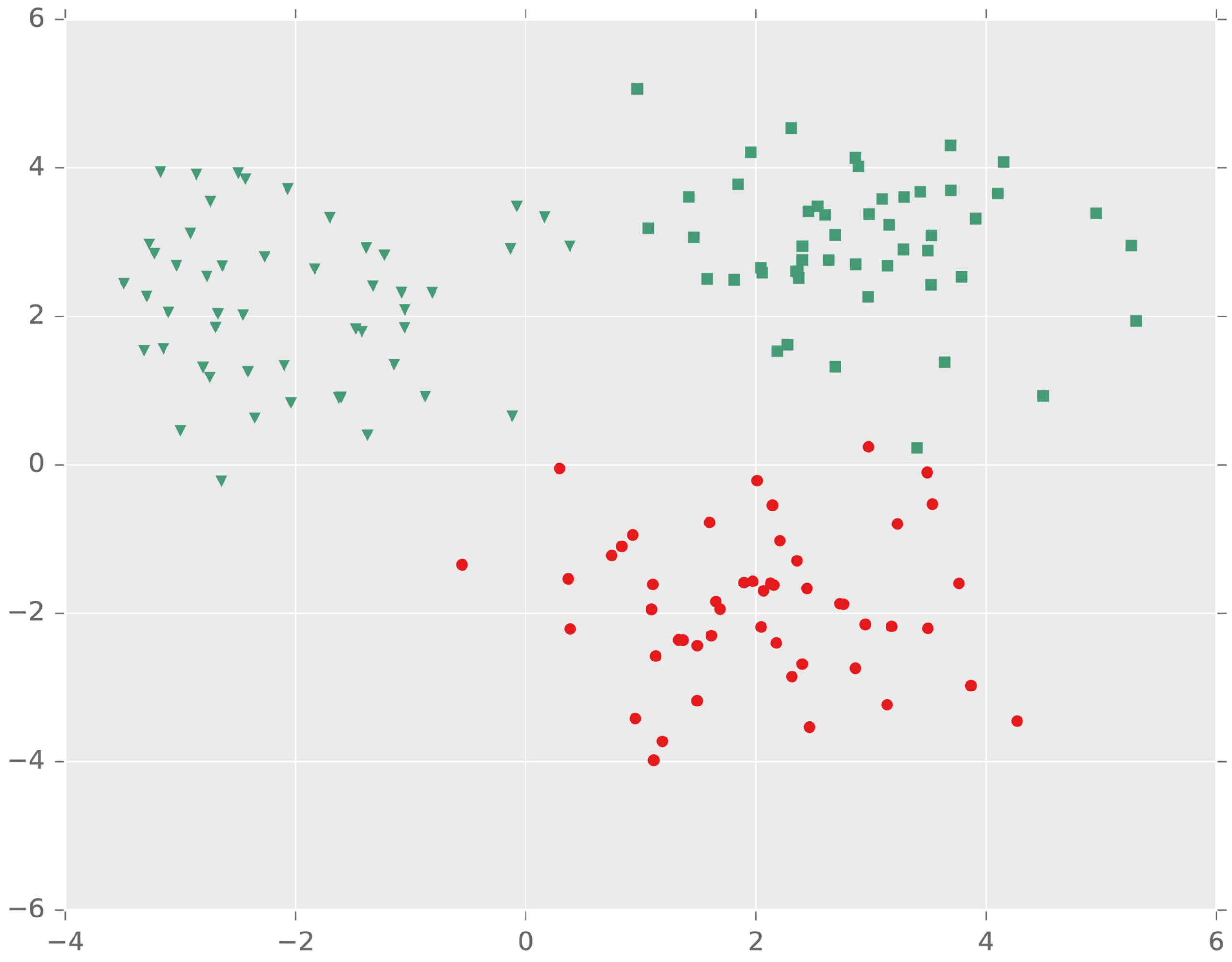
K-Means Algorithm

1) **Given** unlabeled feature vectors
   $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$
2) **Initialize** cluster centers $c = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$
3) **Repeat** until convergence:
   a) for i in $\{1, \ldots, N\}$
      $z^{(i)} \leftarrow$ **index** j of cluster center **nearest** to $\mathbf{x}^{(i)}$
   b) for j in $\{1, \ldots, K\}$
      $\mathbf{c}_j \leftarrow$ **mean** of **all** points assigned to cluster j

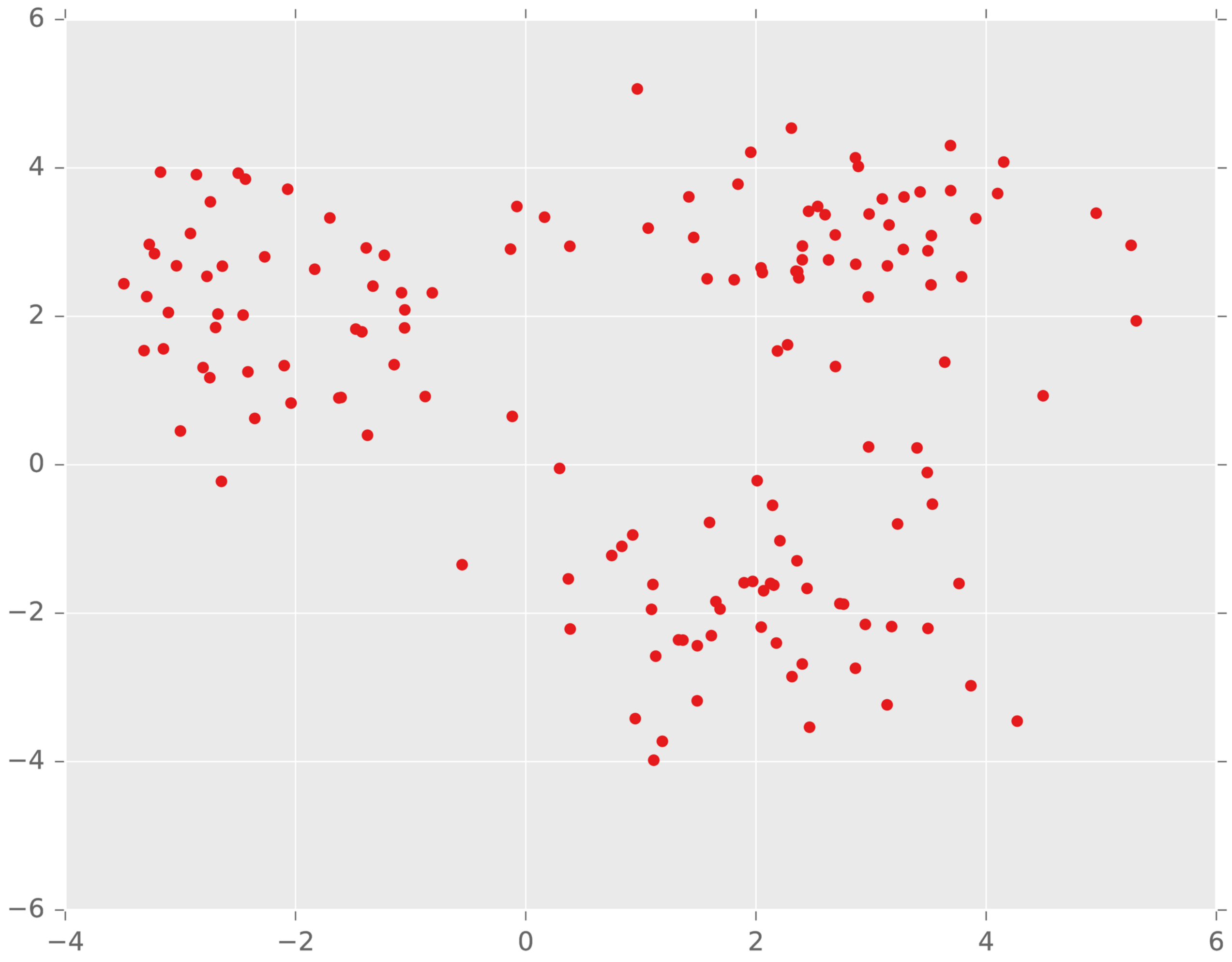
K=3 cluster centers

K-MEANS EXAMPLE

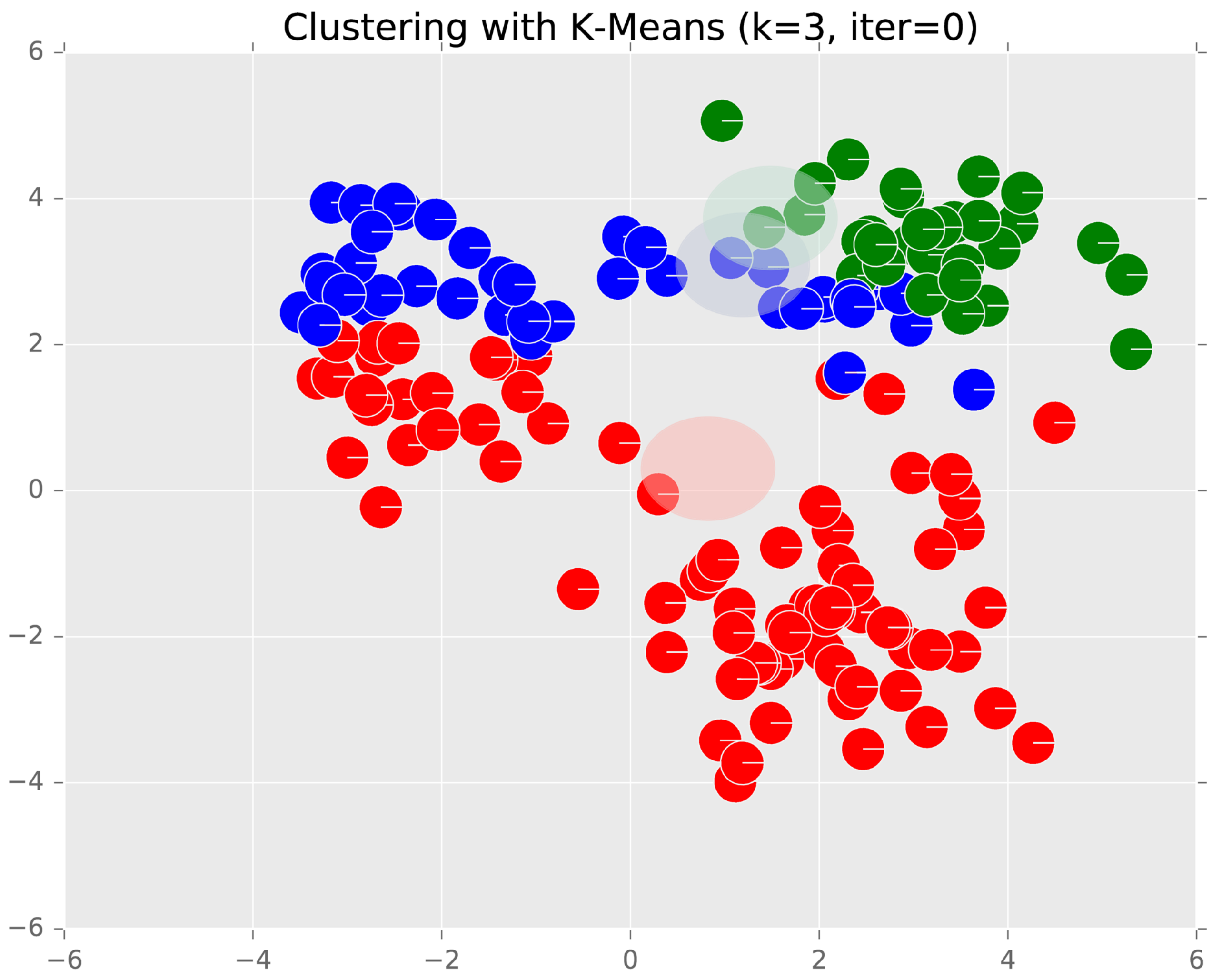
Example: K-Means



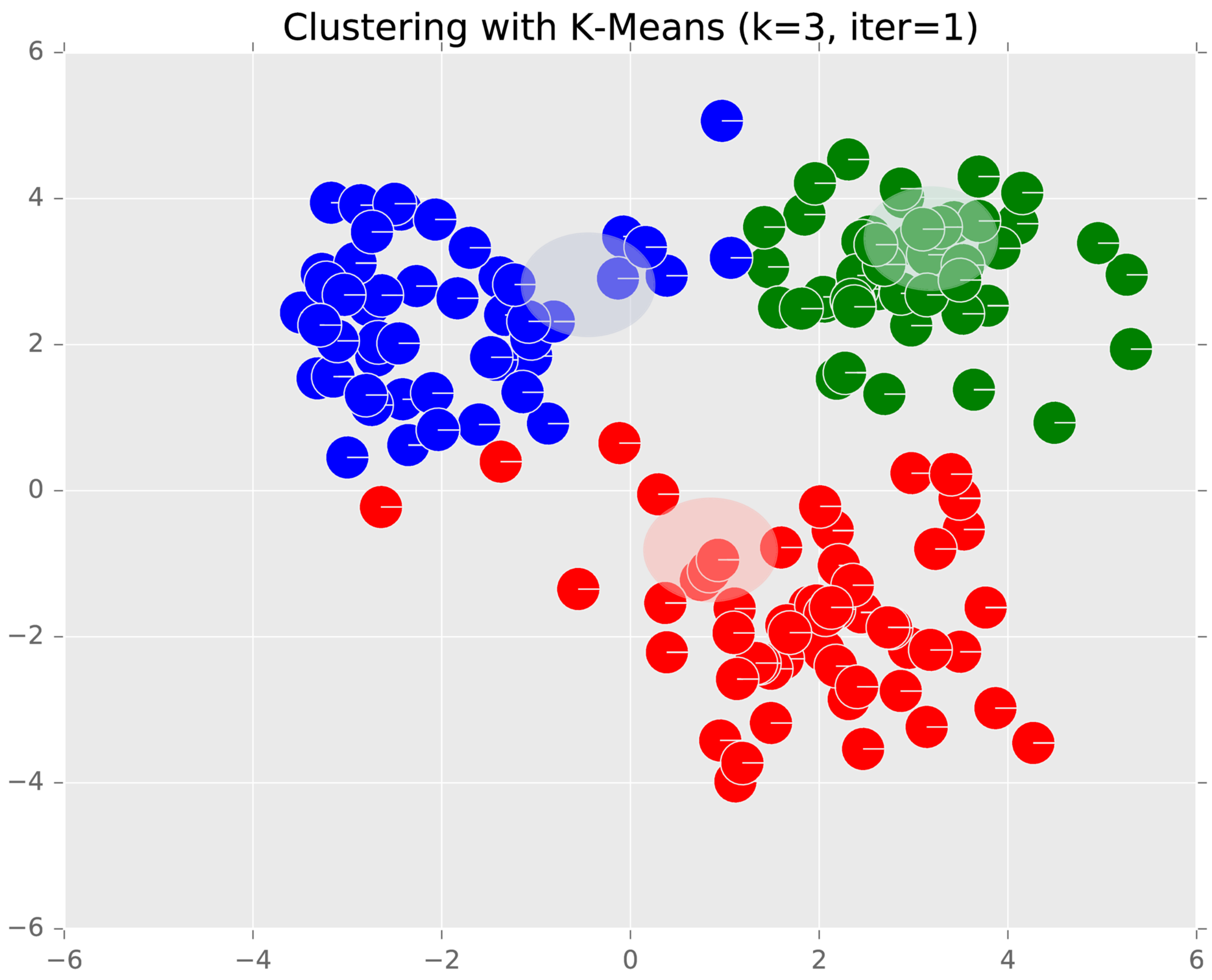
Example: K-Means



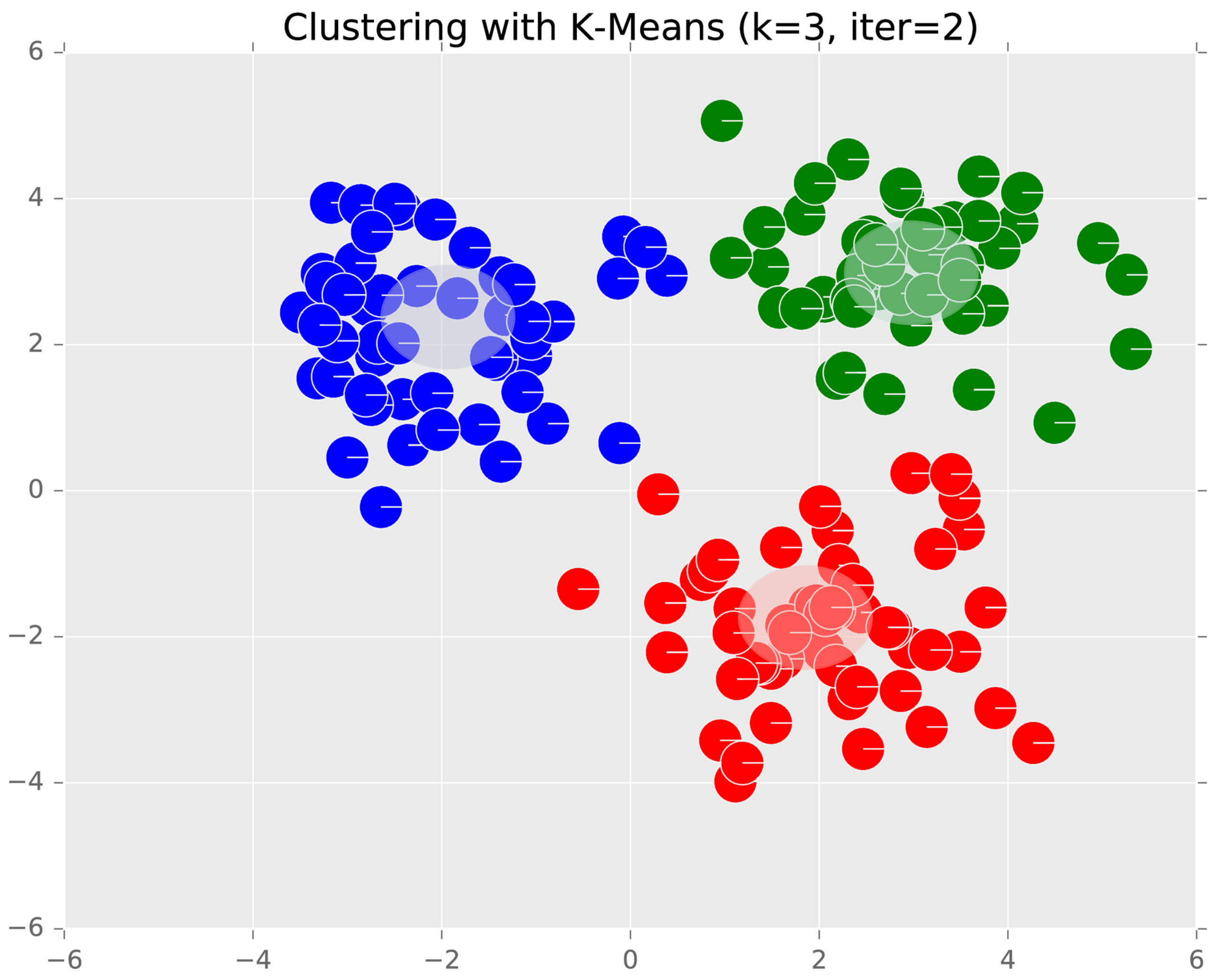
Example: K-Means



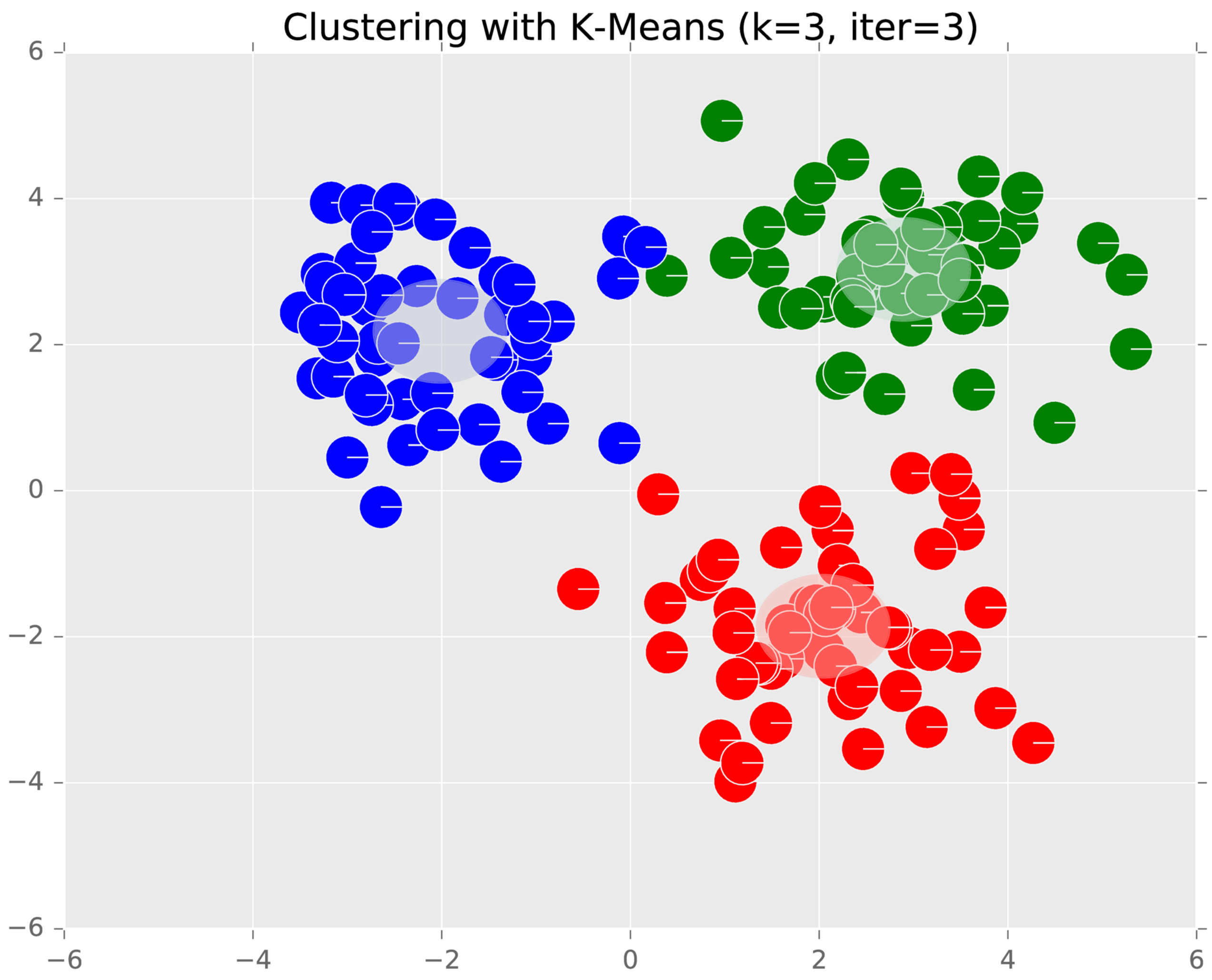
Example: K-Means



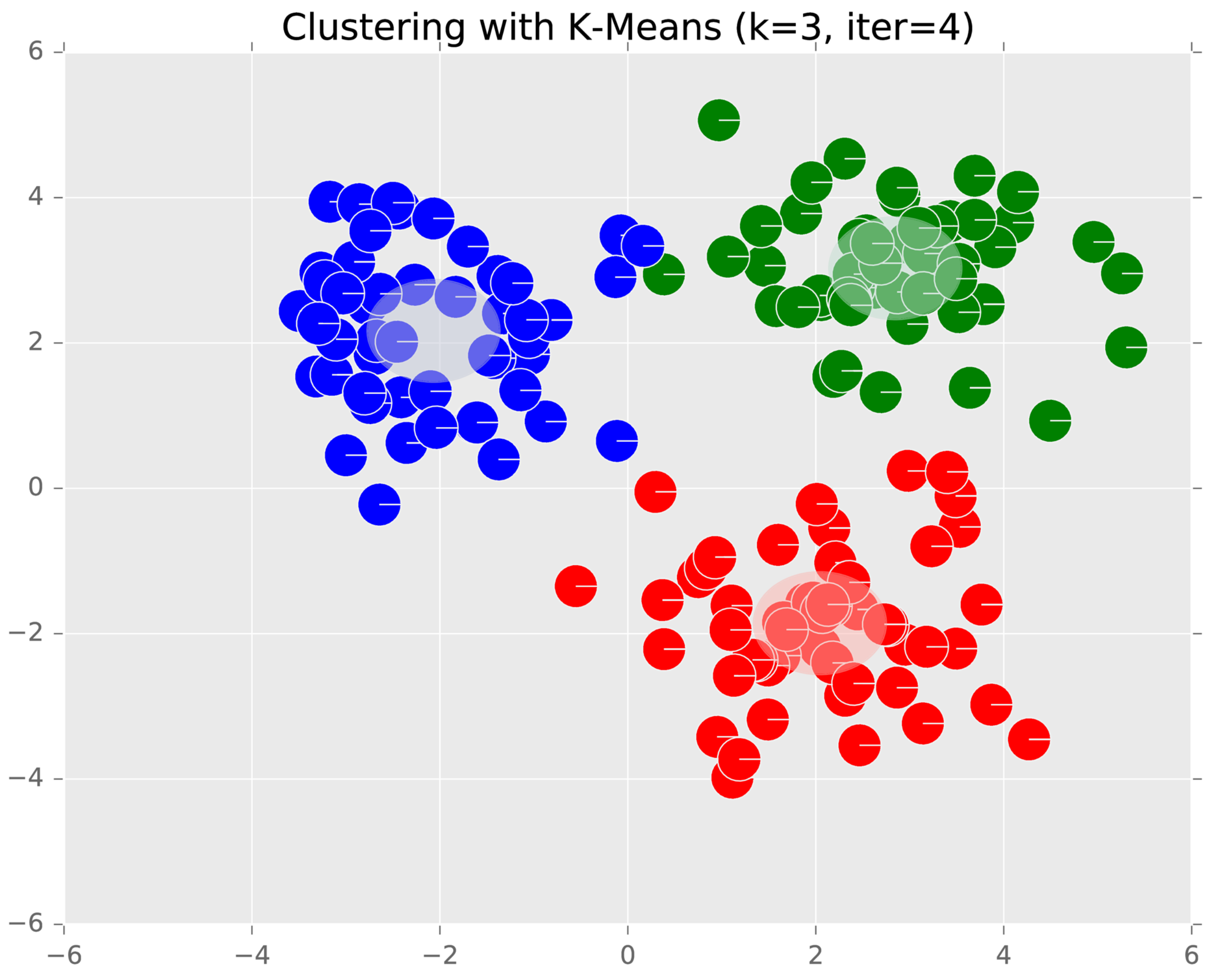
Example: K-Means



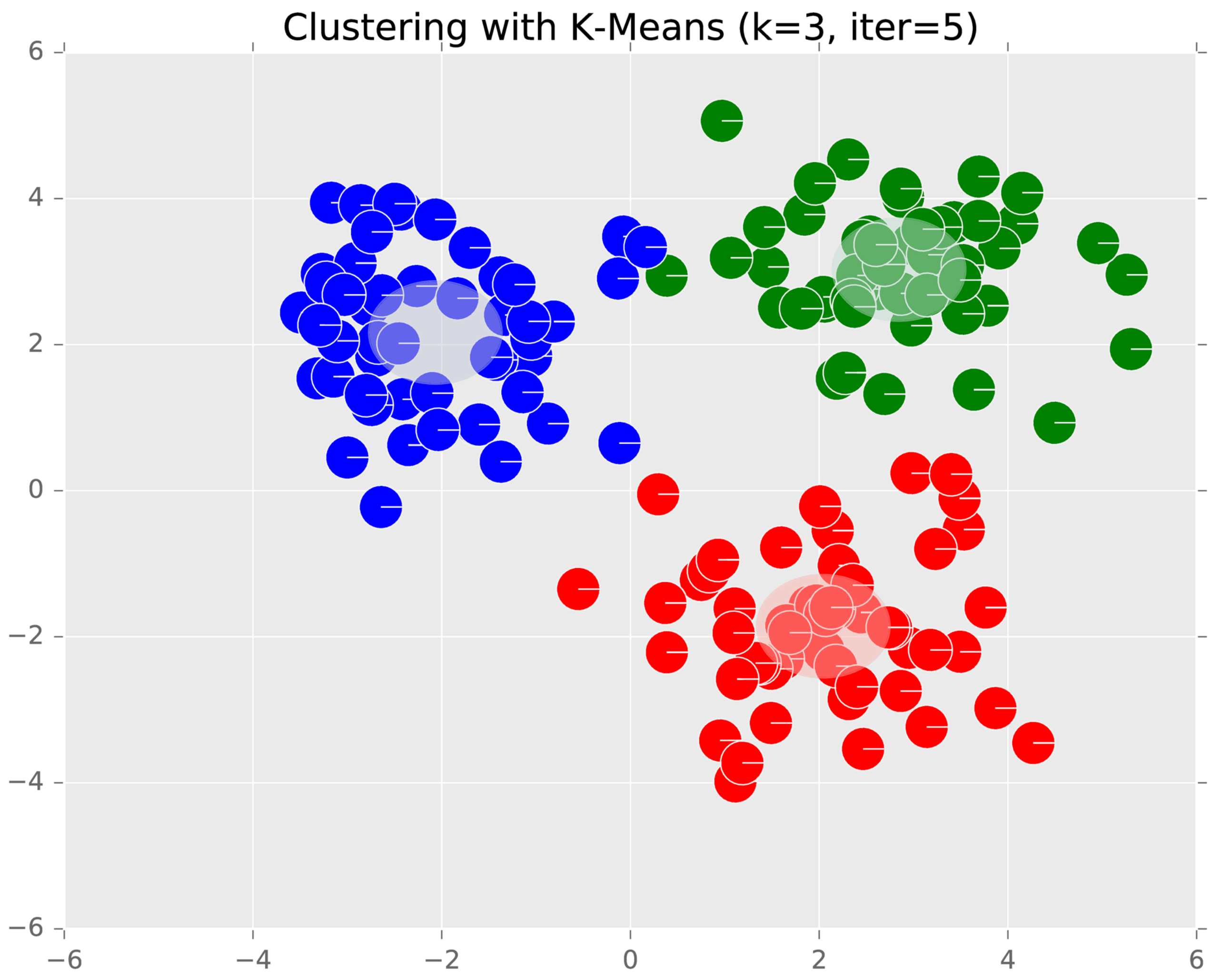
Example: K-Means



Example: K-Means



Example: K-Means

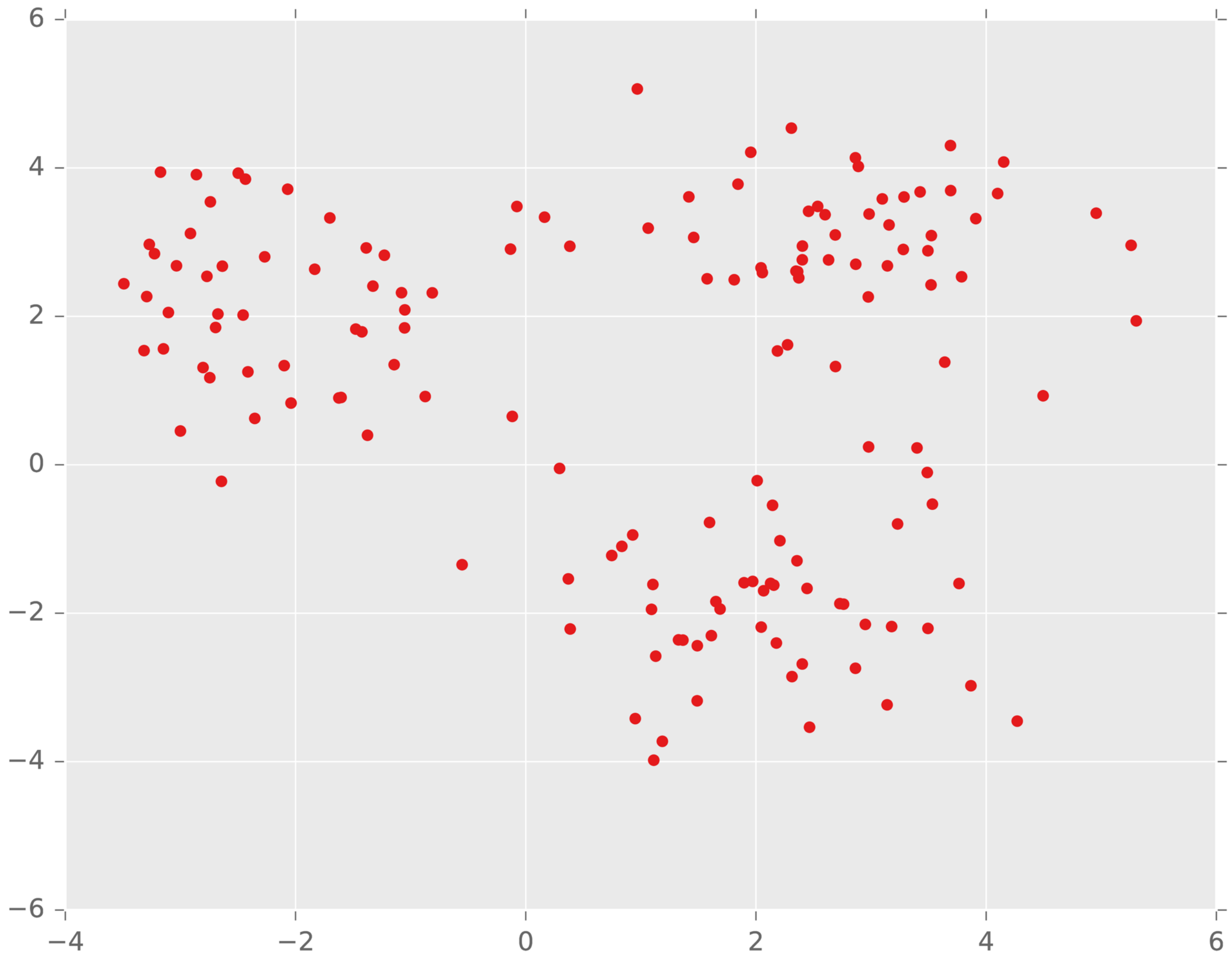


K=2 cluster centers

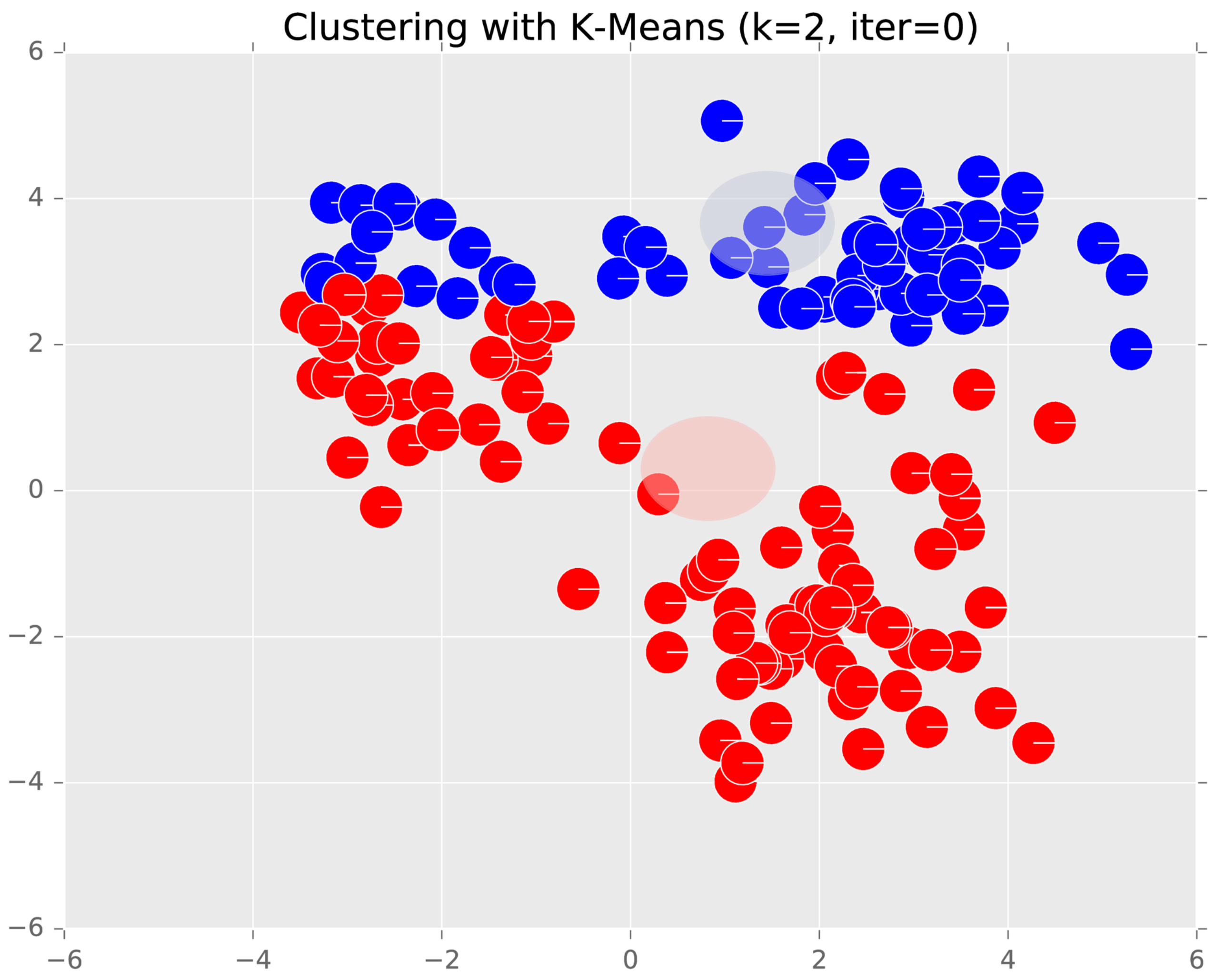
K-MEANS EXAMPLE

Example: K-Means

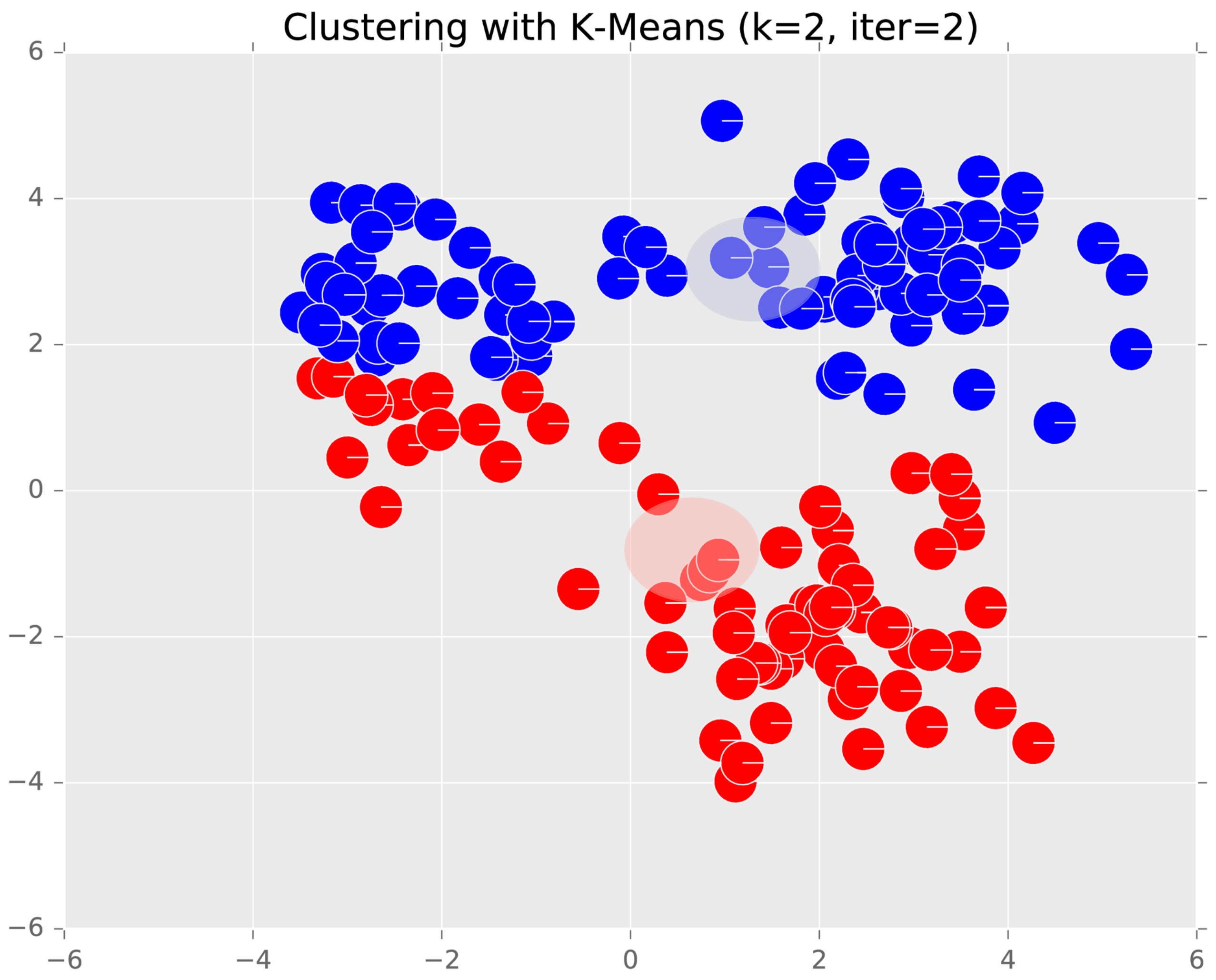
Example: K-Means



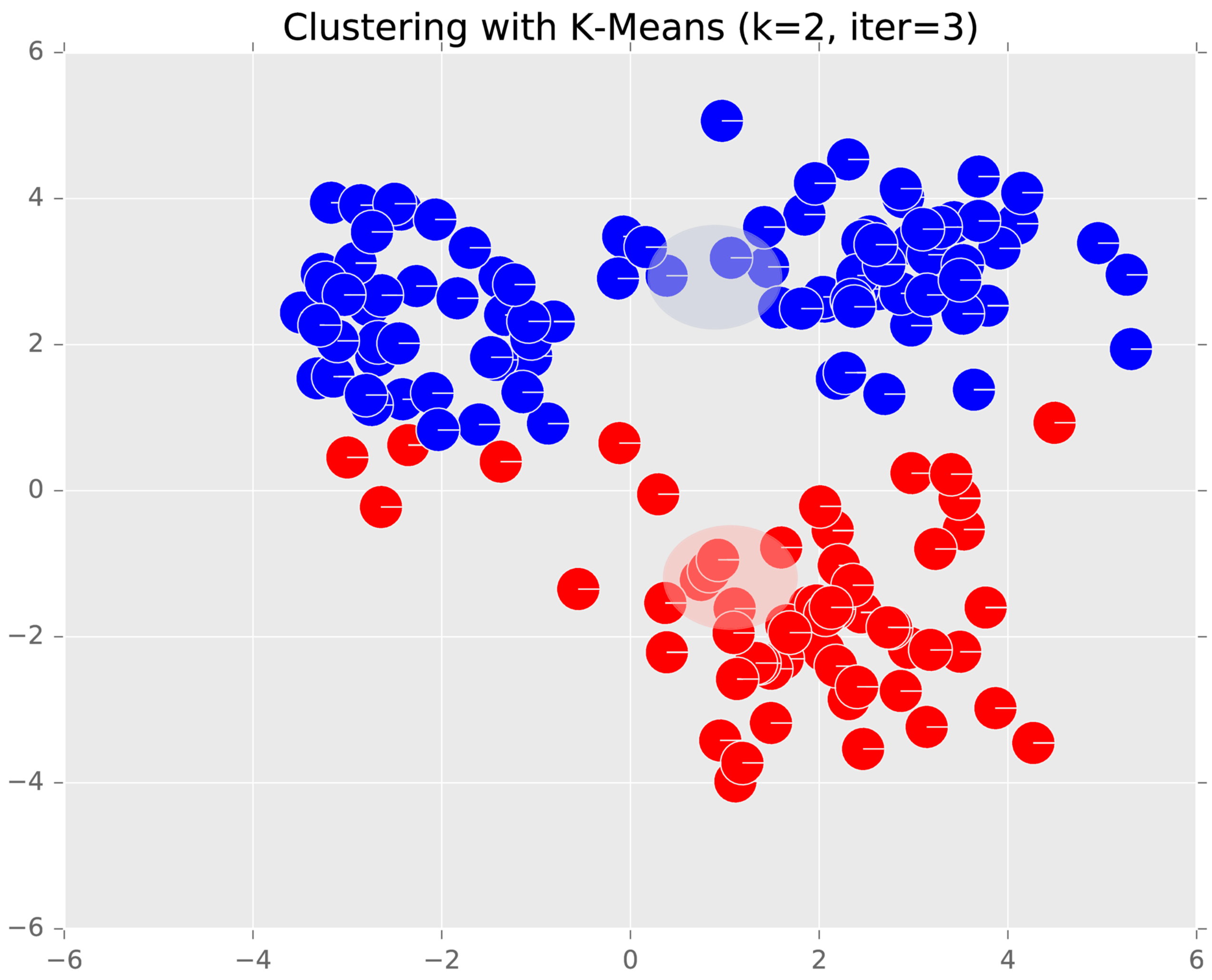
Example: K-Means



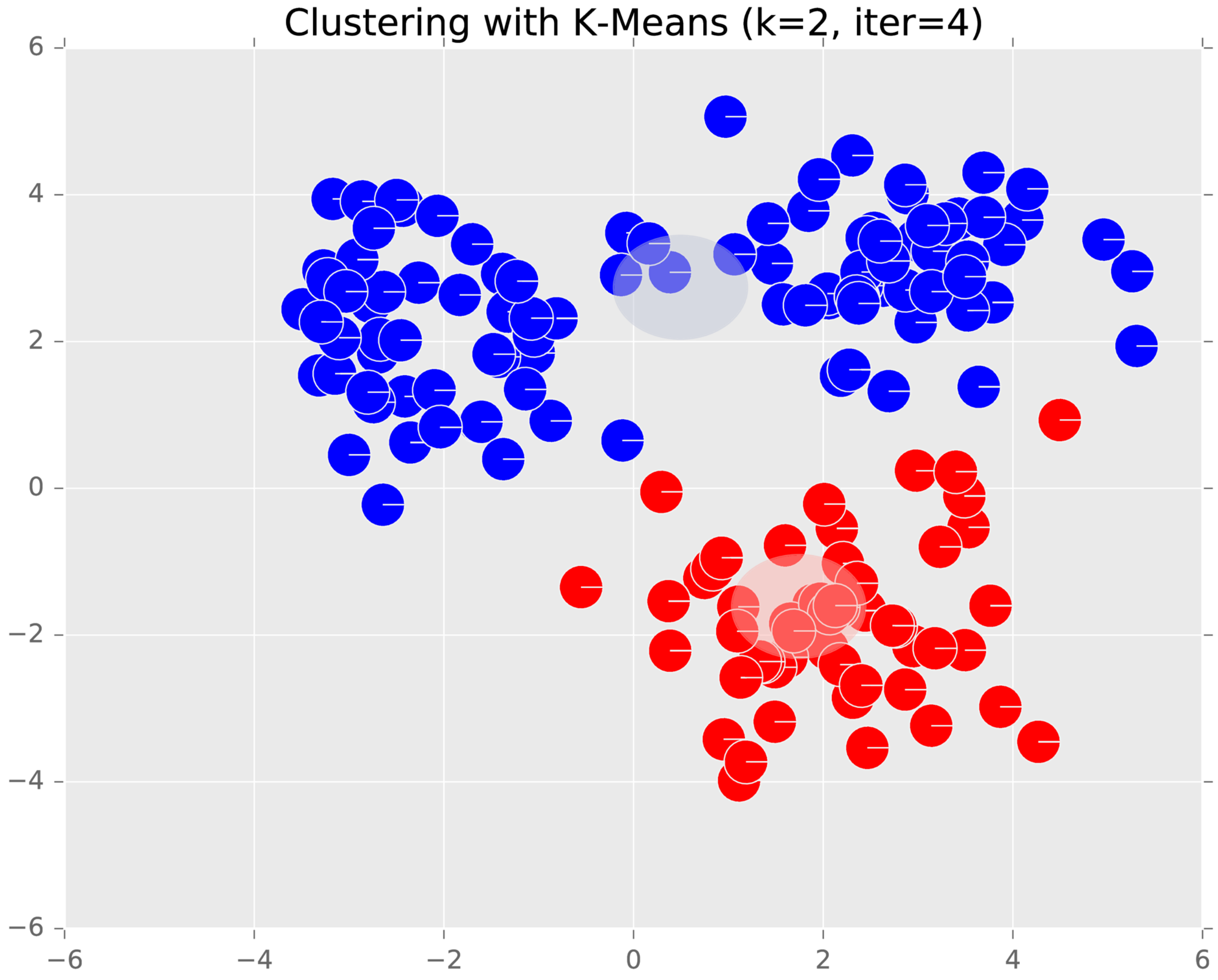
Example: K-Means



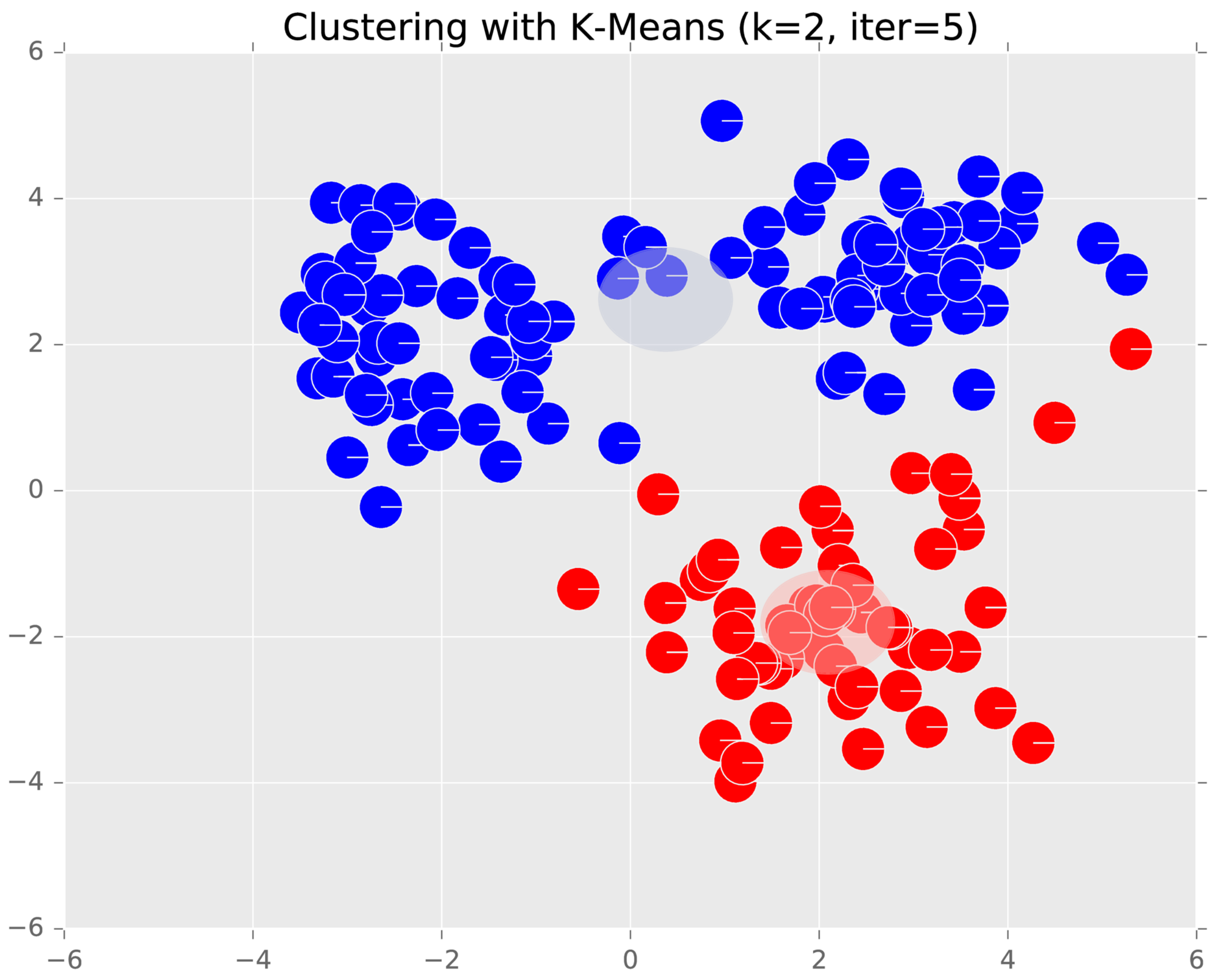
Example: K-Means



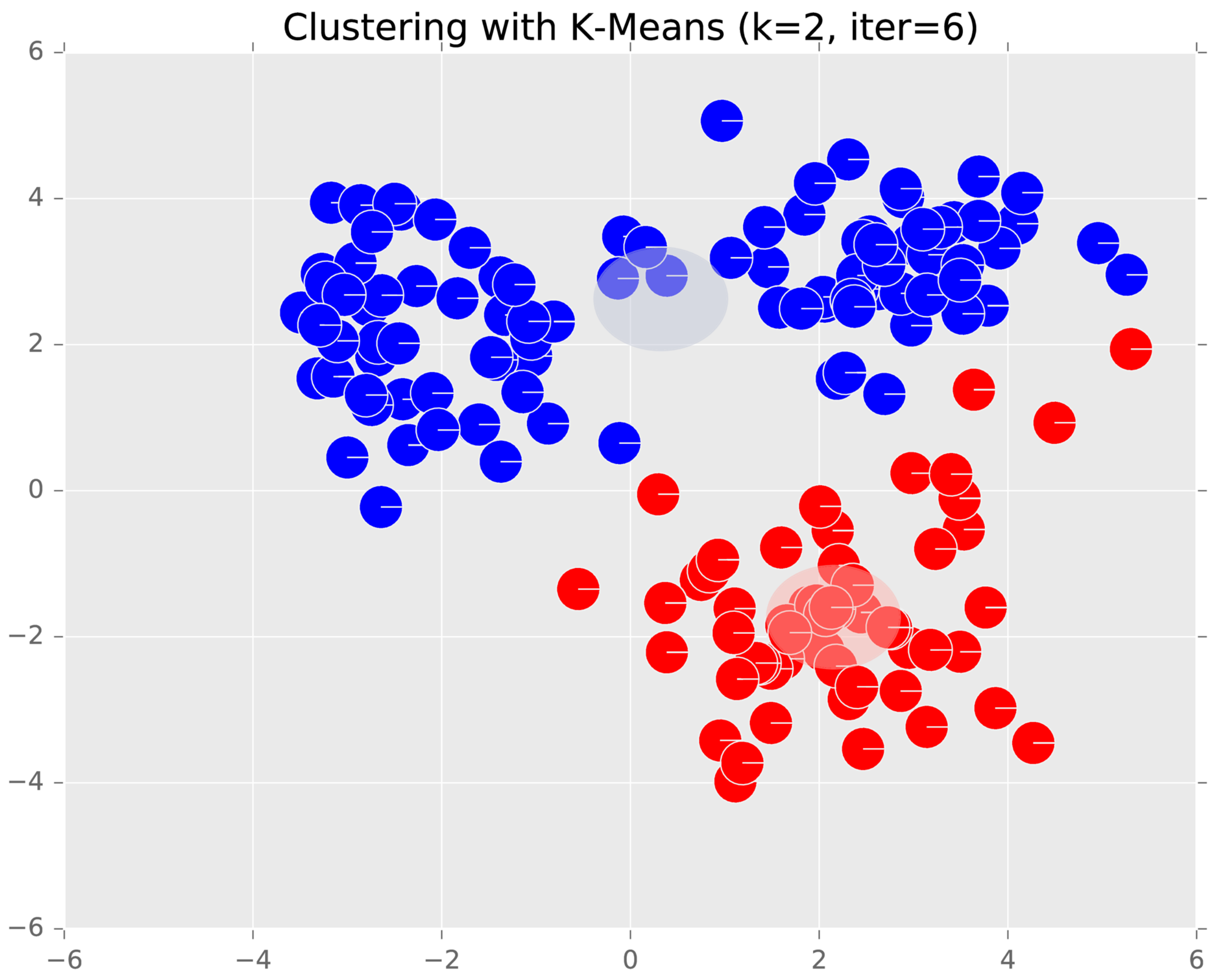
Example: K-Means



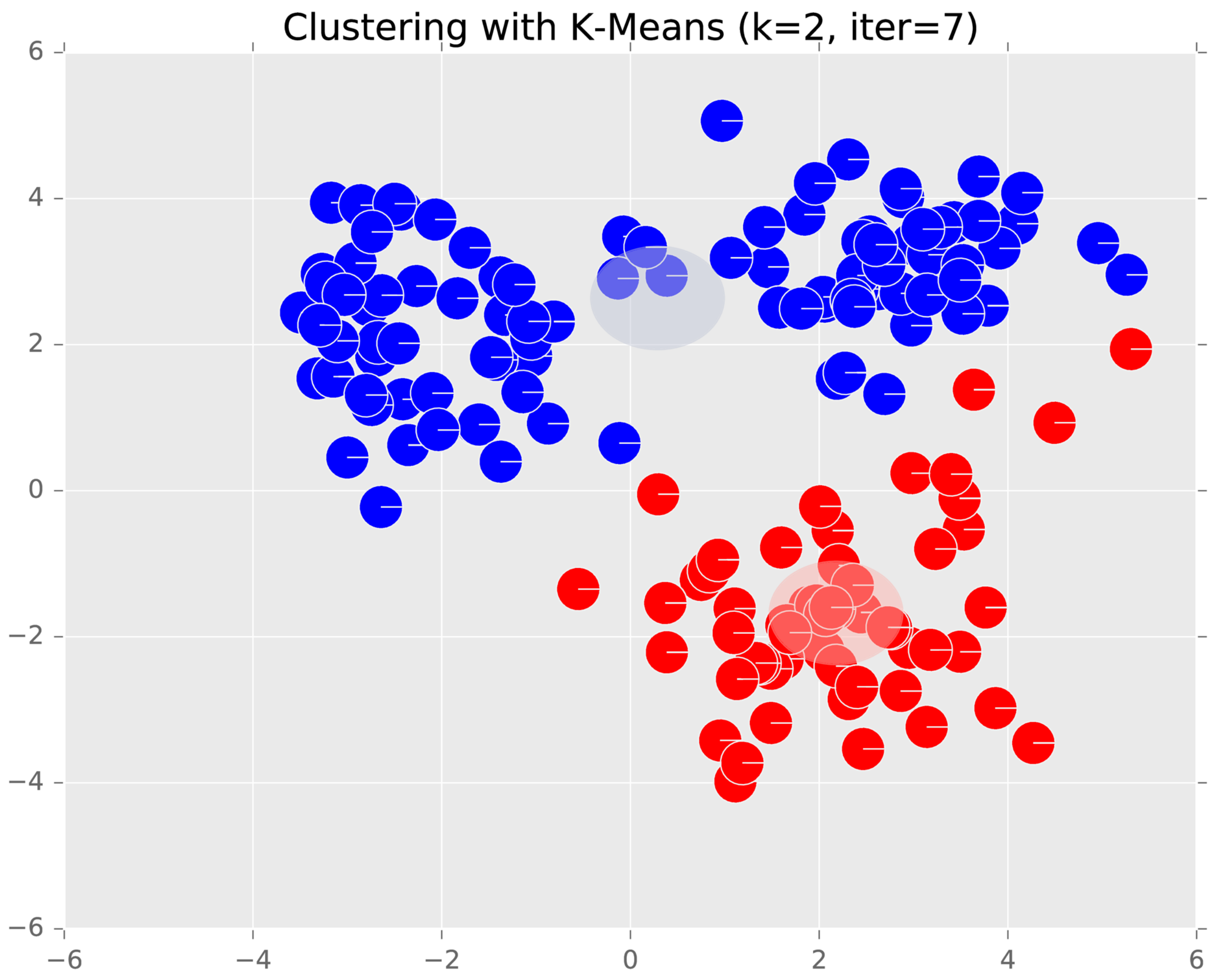
Example: K-Means



Example: K-Means



Example: K-Means



INITIALIZING K-MEANS

Initialization of K-Means

K-Means Algorithm

1) **Given** unlabeled feature vectors
   $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$
2) **Initialize** cluster centers $c = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$
3) **Repeat** until
   a) for i in $\{1, \ldots, N\}$
      $z^{(i)} \leftarrow$ **inde**
   b) for j in $\{1, \ldots, K\}$
      $\mathbf{c}_j \leftarrow$ **mean**

Remaining Question:
How should we initialize the cluster centers?

Three Solutions:

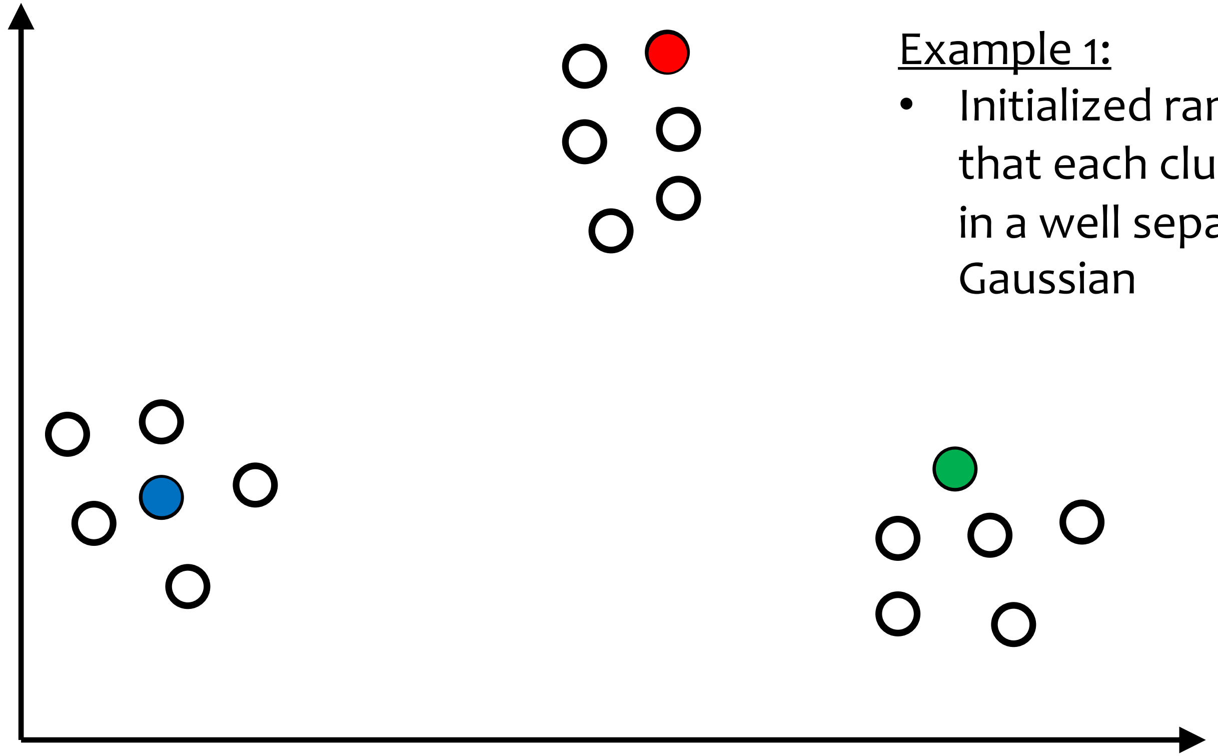
1. Random centers (picked from the data points)
2. Furthest point heuristic
3. K-Means++

Initialization for K-Means

<u>Algorithm #1: Random Initialization</u>
Select each cluster center uniformly at random from the data points in the training data

<u>Observations:</u>
Even when data comes from well-separated Gaussians…

- …sometimes works great!
- …sometimes get stuck in poor local optima.



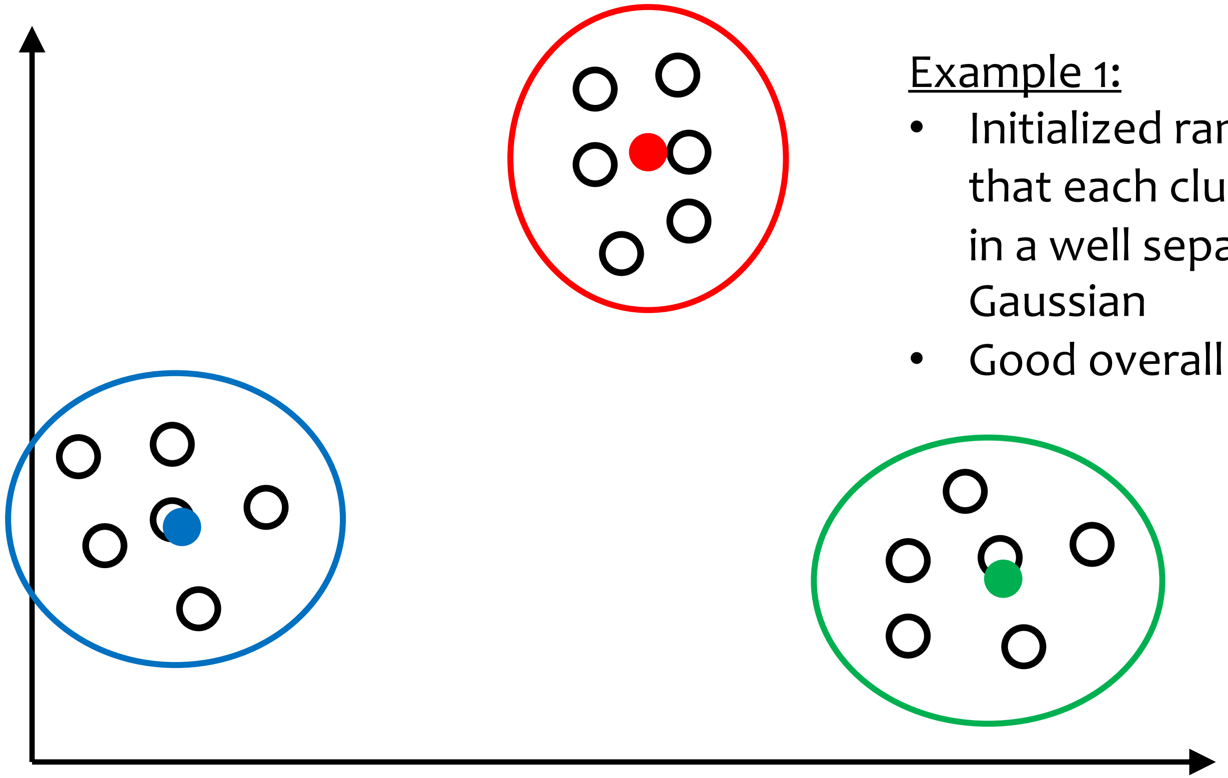
<u>Example 1:</u>

- Initialized randomly such that each cluster center is in a well separated Gaussian

Initialization for K-Means

<u>Algorithm #1: Random Initialization</u>
Select each cluster center uniformly at random from the data points in the training data

<u>Observations:</u>
Even when data comes from well-separated Gaussians…
- …sometimes works great!
- …sometimes get stuck in poor local optima.

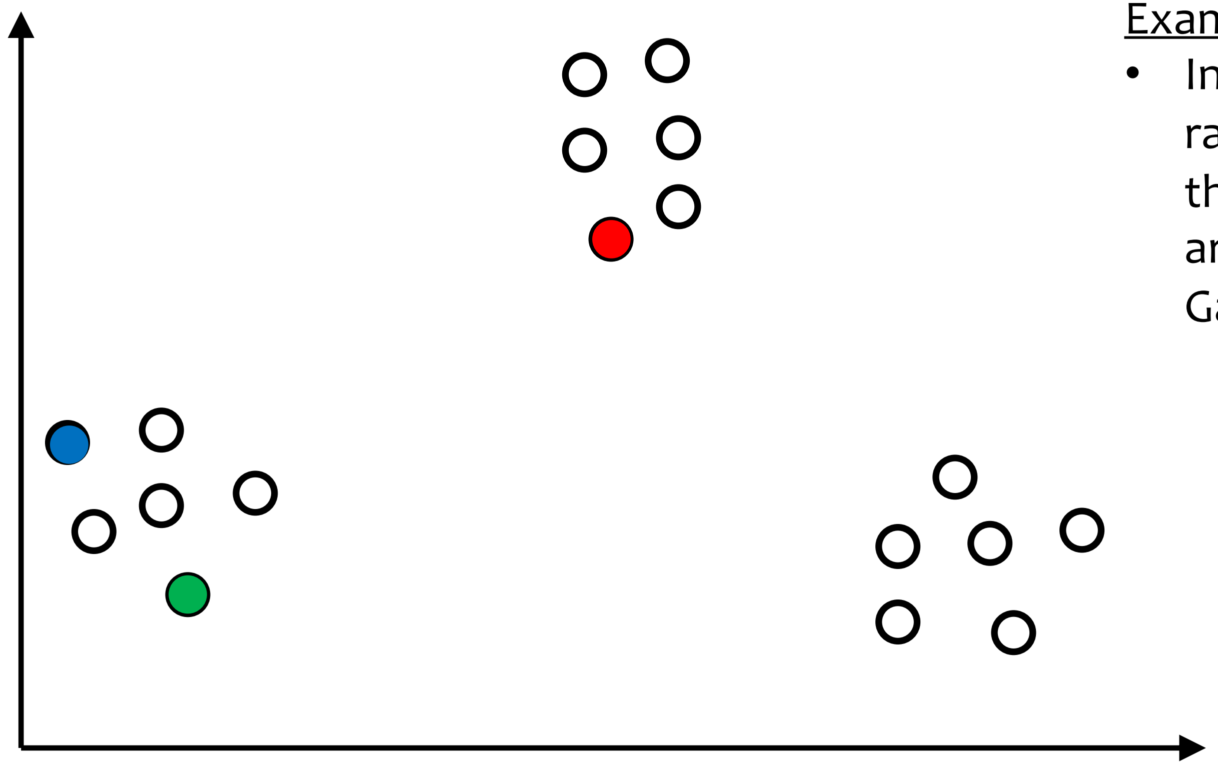


<u>Example 1:</u>
- Initialized randomly such that each cluster center is in a well separated Gaussian
- Good overall performance

Initialization for K-Means

<u>Algorithm #1: Random Initialization</u>
Select each cluster center uniformly at random from the data points in the training data

<u>Observations:</u>
Even when data comes from well-separated Gaussians…
- …sometimes works great!
- …sometimes get stuck in poor local optima.

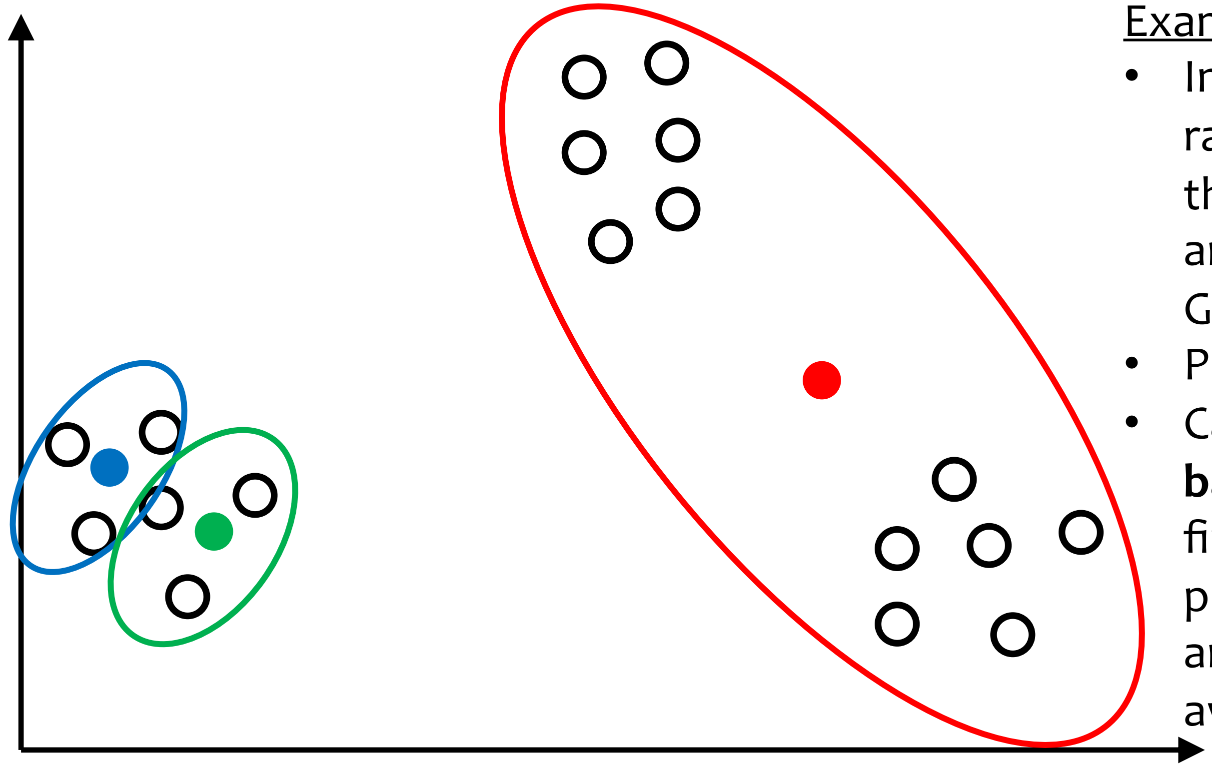


<u>Example 2:</u>
- Initialized randomly such that two centers are in the same Gaussian cluster

Initialization for K-Means

<u>Algorithm #1: Random Initialization</u>
Select each cluster center uniformly at random from the data points in the training data

<u>Observations:</u>
Even when data comes from well-separated Gaussians…
- …sometimes works great!
- …sometimes get stuck in poor local optima.



<u>Example 2:</u>
- Initialized randomly such that two centers are in the same Gaussian cluster
- Poor performance
- Can be **arbitrarily bad** (imagine the final red cluster points moving arbitrarily far away!)

Initialization for K-Means

K-Mean Performance (with Random Initialization)

If we do **random initialization**, as k increases, it becomes more likely we won't have perfectly picked one center per Gaussian in our initialization (so K-Means will output a bad solution).

- For k equal-sized Gaussians,

$$\Pr[\text{each initial center is in a different Gaussian}] \approx \frac{k!}{k^k} \approx \frac{1}{e^k}$$

- Becomes unlikely as k gets large.

Slide courtesy of Nina Balcan

Initialization for K-Means

Algorithm #2: Furthest Point Heuristic

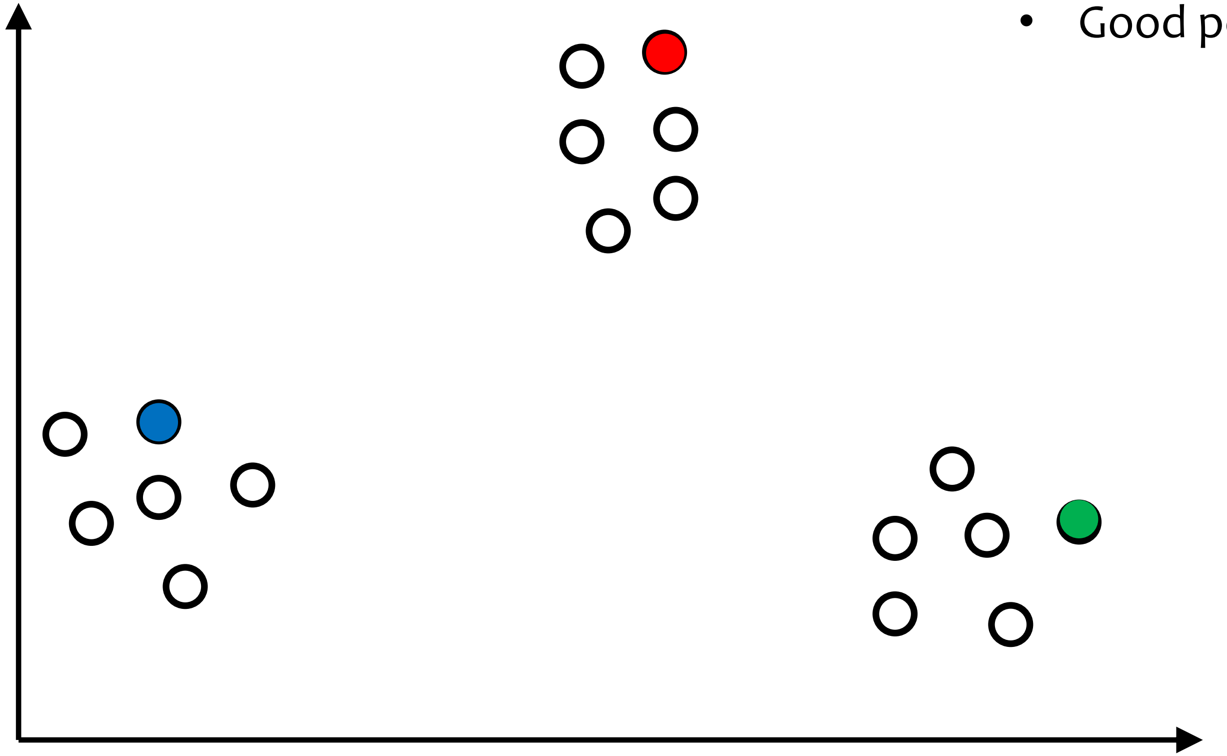
1. Pick the first cluster center $\mathbf{c}_1$ **randomly**
2. Pick each subsequent center $\mathbf{c}_j$ so that it is **as far as possible** from the previously chosen centers $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{j-1}$

Observations:

- Solves the problem with Gaussian data
- But outliers pose a new problem!

Example 1:

- No outliers
- Good performance



Initialization for K-Means

<u>Algorithm #2: Furthest Point Heuristic</u>

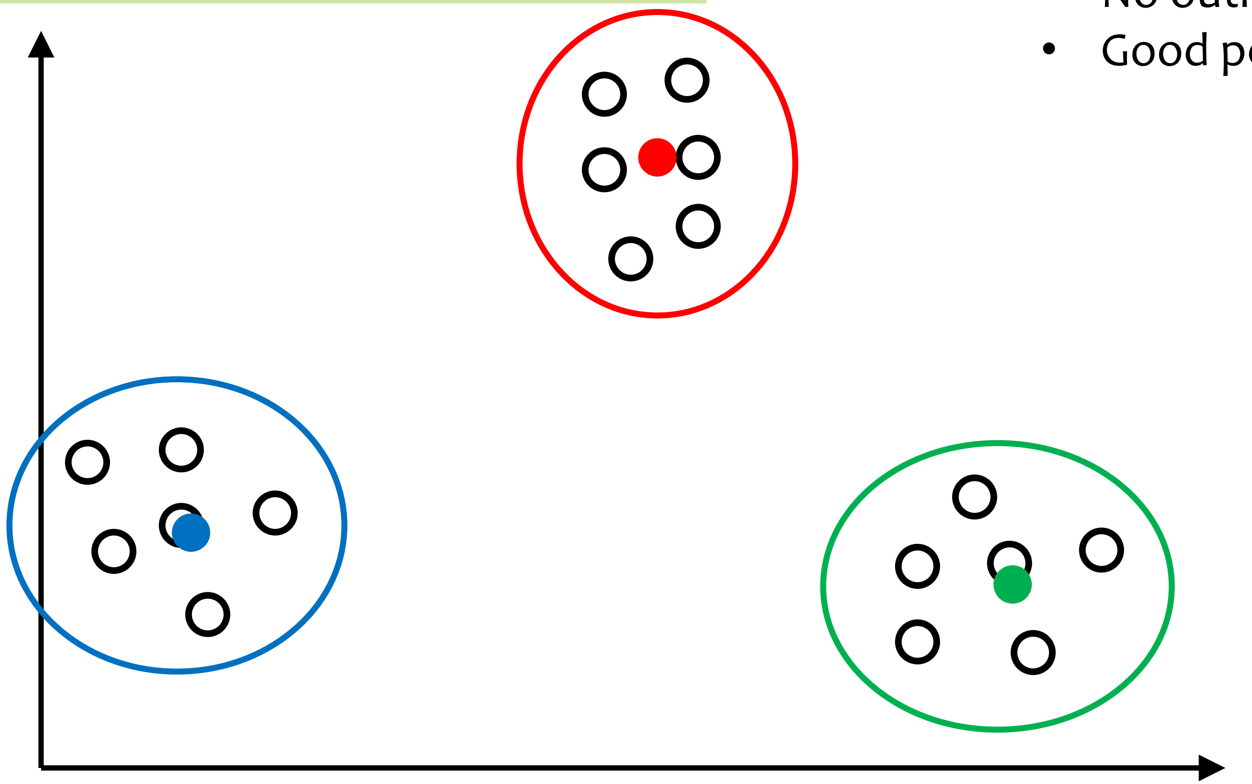
1. Pick the first cluster center $\mathbf{c}_1$ **randomly**
2. Pick each subsequent center $\mathbf{c}_j$ so that it is **as far as possible** from the previously chosen centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{j-1}$

<u>Observations:</u>

- Solves the problem with Gaussian data
- But outliers pose a new problem!

<u>Example 1:</u>

- No outliers
- Good performance



Initialization for K-Means

<u>Algorithm #2: Furthest Point Heuristic</u>

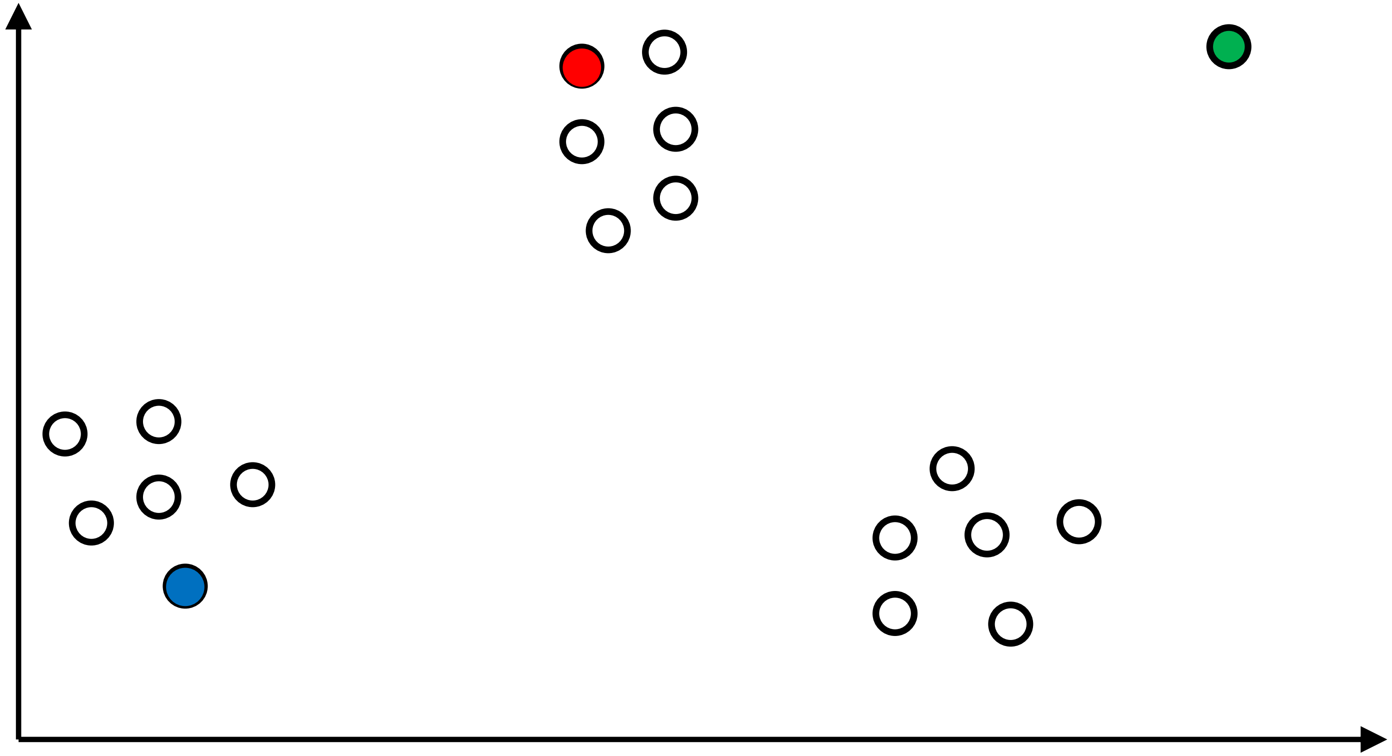
1. Pick the first cluster center $\mathbf{c}_1$ **randomly**
2. Pick each subsequent center $\mathbf{c}_j$ so that it is **as far as possible** from the previously chosen centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{j-1}$

<u>Observations:</u>

- Solves the problem with Gaussian data
- But outliers pose a new problem!

<u>Example 2:</u>

- One outlier throws off the algorithm
- Poor performance



Initialization for K-Means

<u>Algorithm #2: Furthest Point Heuristic</u>

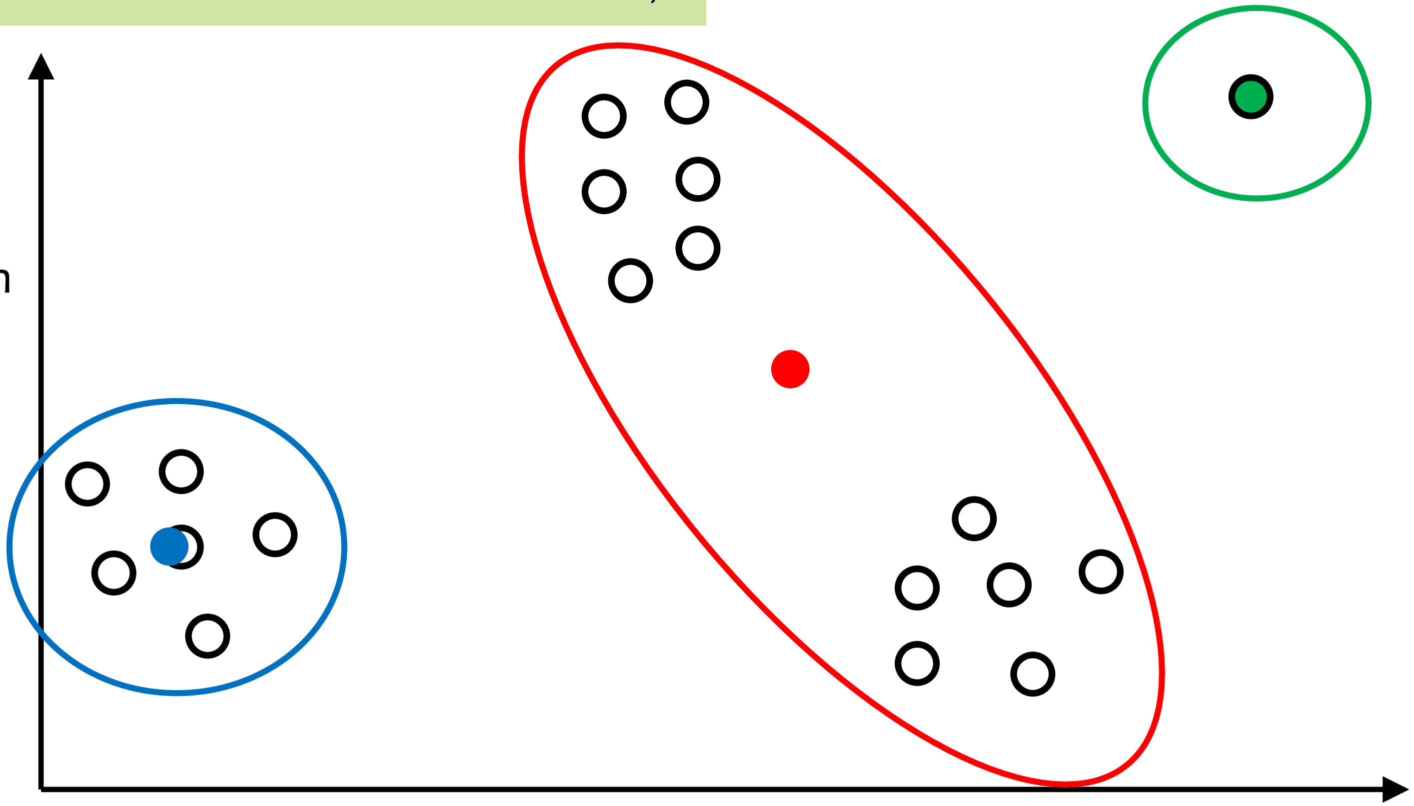
1. Pick the first cluster center $\mathbf{c}_1$ **randomly**
2. Pick each subsequent center $\mathbf{c}_j$ so that it is **as far as possible** from the previously chosen centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{j-1}$

<u>Observations:</u>

- Solves the problem with Gaussian data
- But outliers pose a new problem!

<u>Example 2:</u>

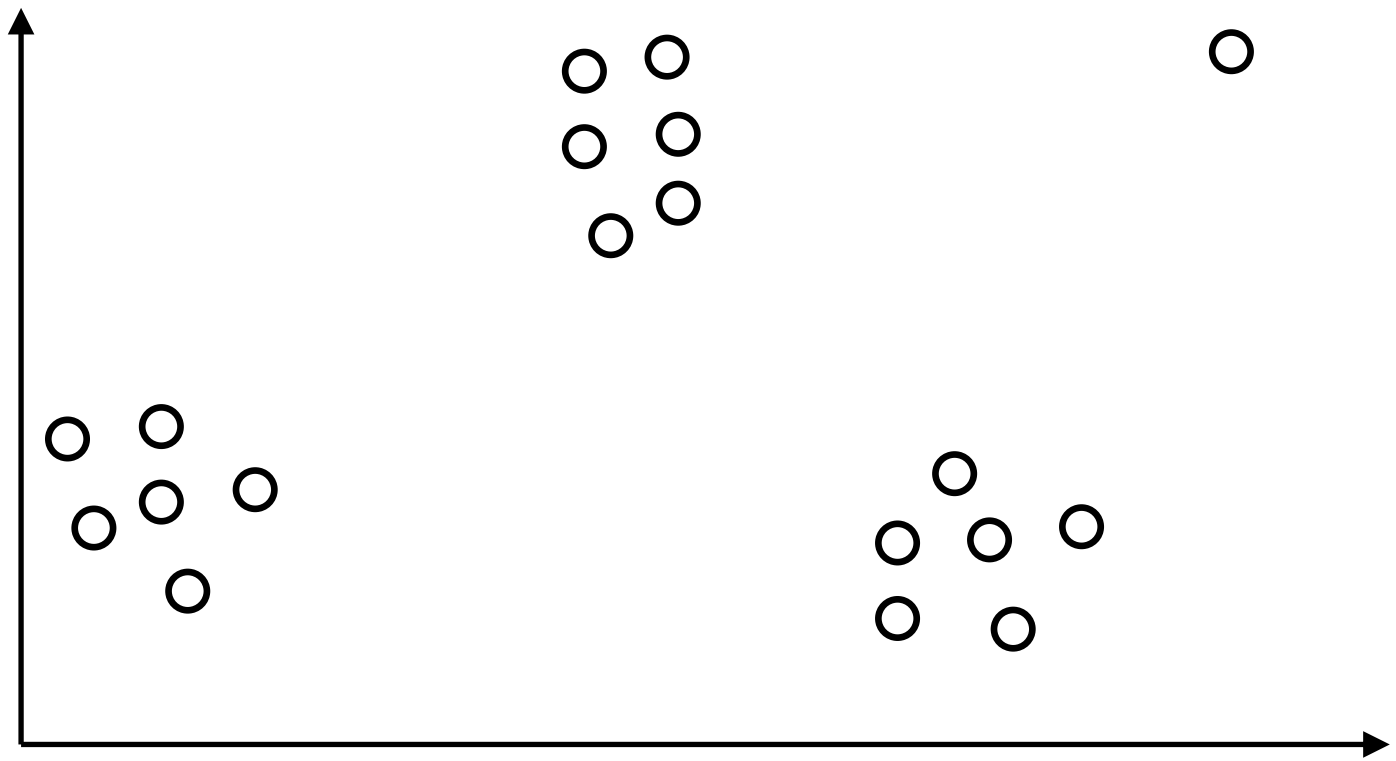
- One outlier throws off the algorithm
- Poor performance



Initialization for K-Means

<u>Algorithm #3: K-Means++</u>

- Let $\mathrm{D}(\mathbf{x})$ be the distance between a point x and its nearest center. Chose the next center proportional to $\mathrm{D}^2(\mathbf{x})$.



Initialization for K-M

i	$D(x)$	$D^2(x)$	$P(c_2 = x^{(i)})$
1	3	9	9/137
2	2	4	4/137
…			
7	4	16	16/137
…			
N	3	9	9/137
	Sum:	137	1.0

<u>Algorithm #3: K-Means++</u>

- Let $D(\mathbf{x})$ be the distance between a point x and its nearest center. Chose the next center proportional to $D^2(\mathbf{x})$.

- Choose $\mathbf{c_1}$ at random.
- For $j = 2, \dots, K$
  - Pick $\mathbf{c_j}$ among $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$ according to the distribution

$$\mathbf{P}(\mathbf{c_j} = \mathbf{x}^{(i)}) \propto \mathbf{min}_{j'<j} \left|\left|\mathbf{x}^{(i)} - \mathbf{c}_{j'}\right|\right|^2 \quad D^2(\mathbf{x}^i)$$

Theorem: K-Means++ always attains an $O(\log k)$ approximation to optimal K-Means solution in expectation.

Slide adapted from Nina Balcan

Initialization for K-M

i	D(x)	$D^2(x)$	$P(c_2 = x^{(i)})$
1	3	9	9/137
2	2	4	4/137
…			
7	4	16	16/137
…			
N	3	9	9/137
	Sum:	137	1.0

<u>Algorithm #3: K-Means++</u>

- Let $D(\mathbf{x})$ be the distance between a point x and its nearest center. Chose the next center proportional to $D^2(\mathbf{x})$.

- Choose $\mathbf{c_1}$ at random.
- For $j = 2, \ldots, K$
  - Pick $\mathbf{c_j}$ among $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$ according to the distribution

$$\mathbf{P(c_j = x^{(i)}) \propto \min_{j'<j} \left|\left|x^{(i)} - c_{j'}\right|\right|^2}$$

Theorem: K-Means++ always attains an $O(\log k)$ approximation to optimal K-Means solution in expectation.

Initialization for K-M

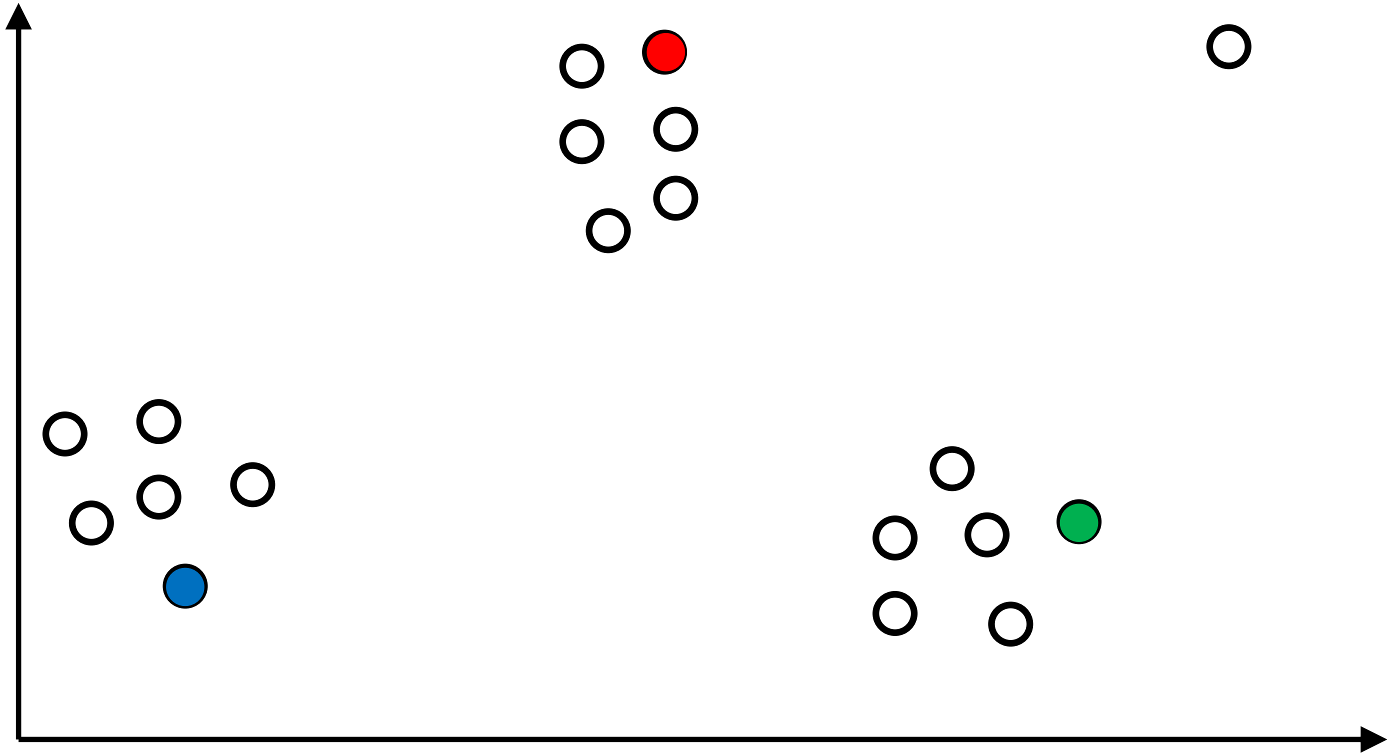
i	D(x)	D²(x)	P(c₂ = x⁽ⁱ⁾)
1	3	9	9/137
2	2	4	4/137
…			
7	4	16	16/137
…			
N	3	9	9/137
	Sum:	137	1.0

<u>Algorithm #3: K-Means++</u>

- Let $\mathrm{D}(\mathbf{x})$ be the distance between a point x and its nearest center. Chose the next center proportional to $\mathrm{D}^2(\mathbf{x})$.

<u>Example 1:</u>

- One outlier
- Good performance



Initialization for K-M

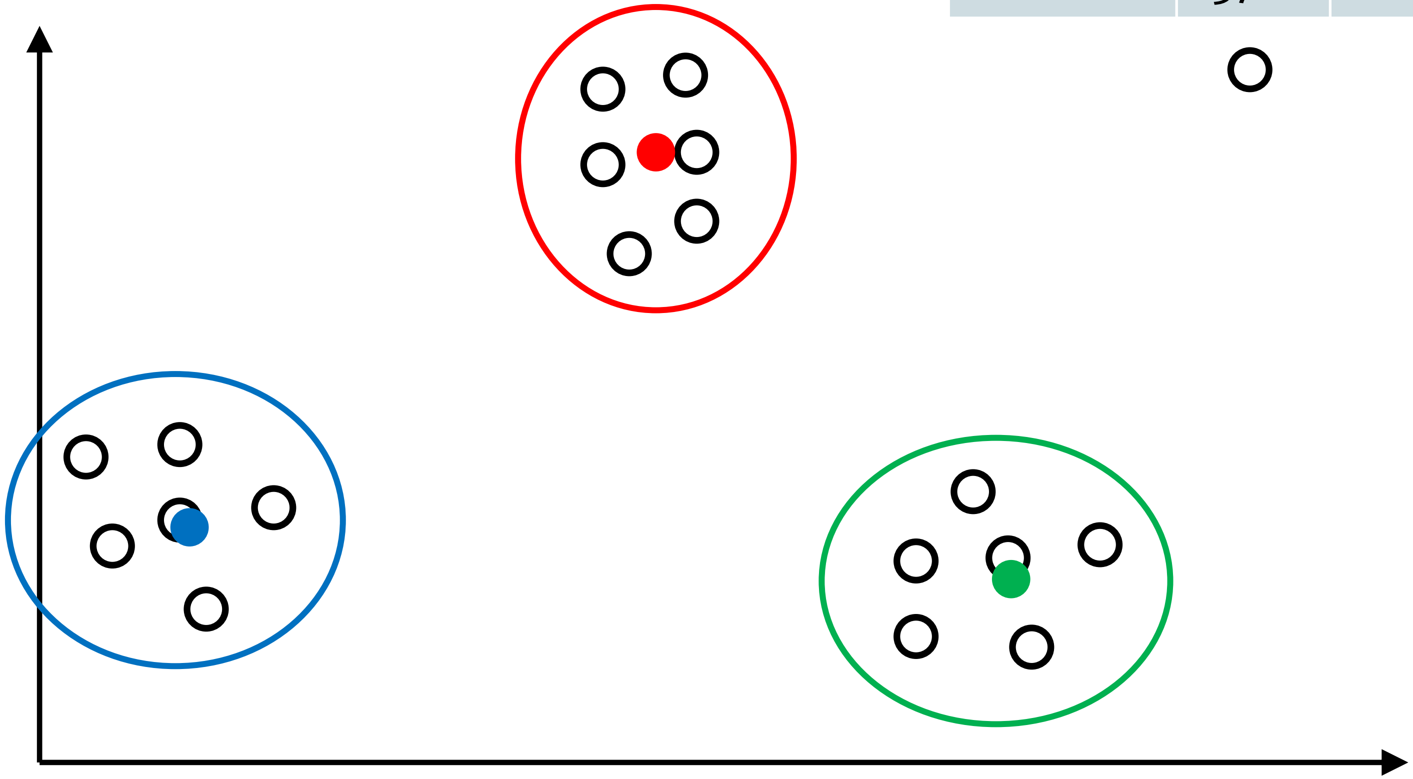
i	D(x)	$D^2(x)$	$P(c_2 = x^{(i)})$
1	3	9	9/137
2	2	4	4/137
…			
7	4	16	16/137
…			
N	3	9	9/137
	Sum:	137	1.0

Algorithm #3: K-Means++

- Let $\mathrm{D}(\mathbf{x})$ be the distance between a point x and its nearest center. Chose the next center proportional to $\mathrm{D}^2(\mathbf{x})$.

Example 1:

- One outlier
- Good performance



Initialization for K-Means

<u>Algorithm #3: K-Means++</u>

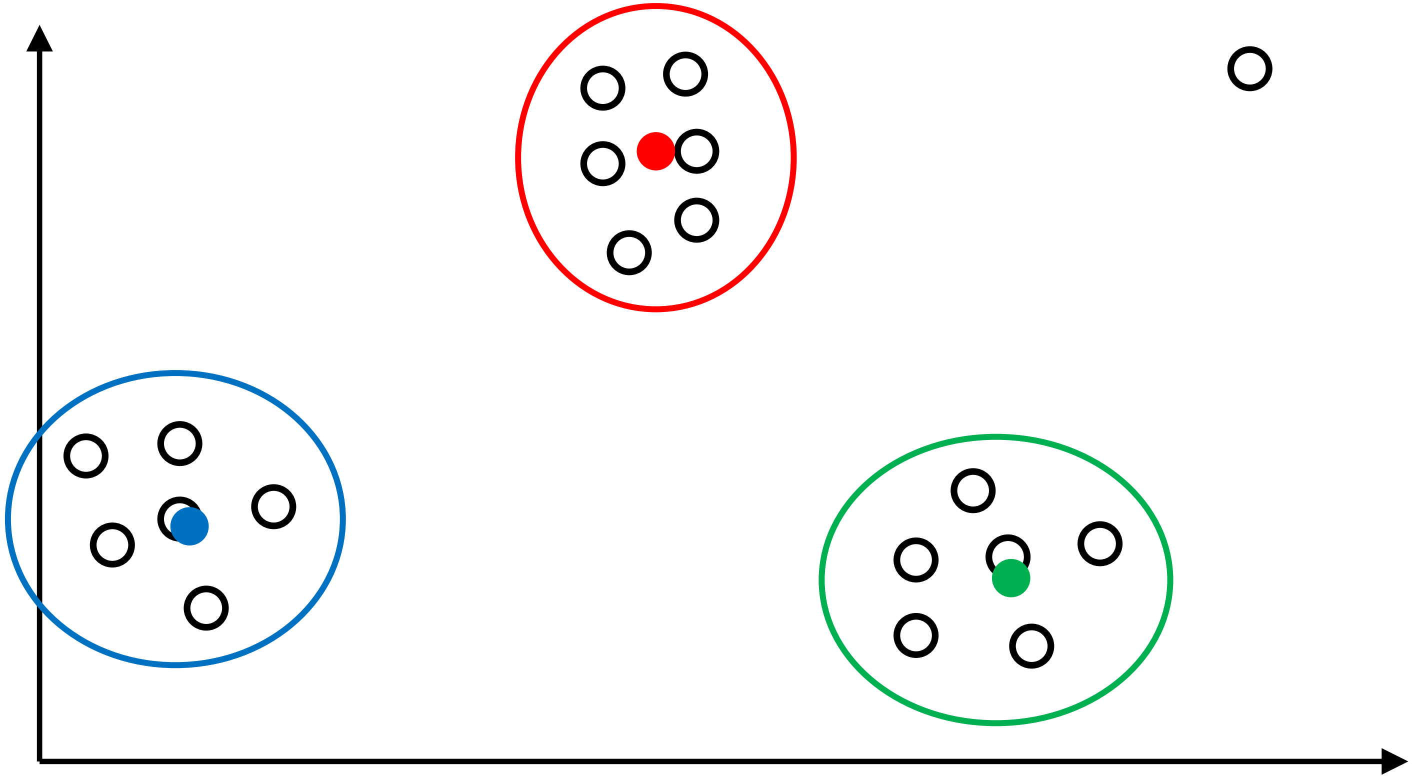
- Let $\mathrm{D}(\mathbf{x})$ be the distance between a point x and its nearest center. Chose the next center proportional to $\mathrm{D}^2(\mathbf{x})$.

<u>Observations:</u>

- Interpolates between random and farthest point initialization
- Solves the problem with Gaussian data
- **And** solves the outlier problem

<u>Example 1:</u>

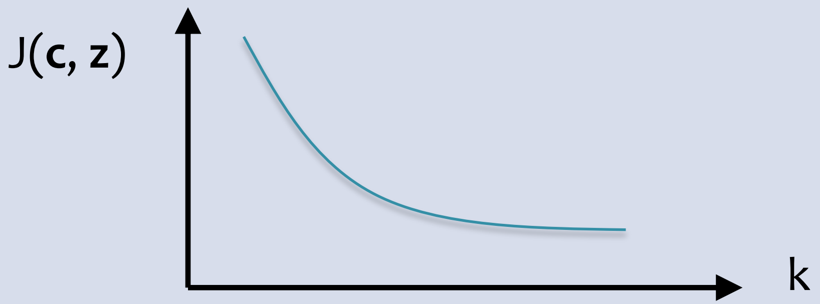
- One outlier
- Good performance



Q&A

Q: In k-Means, since we don't have a validation set, how do we pick k?

A: Look at the training objective function as a function of k and pick the value at the "elbo" of the curve.



Q: What if our random initialization for k-Means gives us poor performance?

A: Do **random restarts**: that is, run k-means from scratch, say, 10 times and pick the run that gives the lowest training objective function value.

The objective function is **nonconvex**, so we're just looking for the best local minimum.

Learning Objectives

K-Means

You should be able to…

1. Distinguish between coordinate descent and block coordinate descent
2. Define an objective function that gives rise to a "good" clustering
3. Apply block coordinate descent to an objective function preferring each point to be close to its nearest objective function to obtain the K-Means algorithm
4. Implement the K-Means algorithm
5. Connect the non-convexity of the K-Means objective function with the (possibly) poor performance of random initialization

DIMENSIONALITY REDUCTION

High Dimension Data

Examples of high dimensional data:

- High resolution images (millions of pixels)

High Dimension Data

Examples of high dimensional data:

– Multilingual News Stories
  (vocabulary of hundreds of thousands of words)

Hilo Tribune.

KILAUEA CRATER CONTINUES IN ERUPTION

FOUR DAYS OF ACTIVITY

ANOTHER OUTBREAK OCCURS

HONOLULU GETS APPROPRIATIONS FOR HARBOR AND LIGHTHOUSE

River and Harbor Bill Passes.

Postmaster Pratt Confirmed.

Latest Sugar Quotations.

Fire At New Orleans.

叻報

Македония

PRAVA BRAĆA

Истински братя

DEKLARACIJA.

Декларация

குடி அரசு

"KUDI-ARASU"

High Dimension Data

Examples of high dimensional data:

– Brain Imaging Data (100s of MBs per scan)

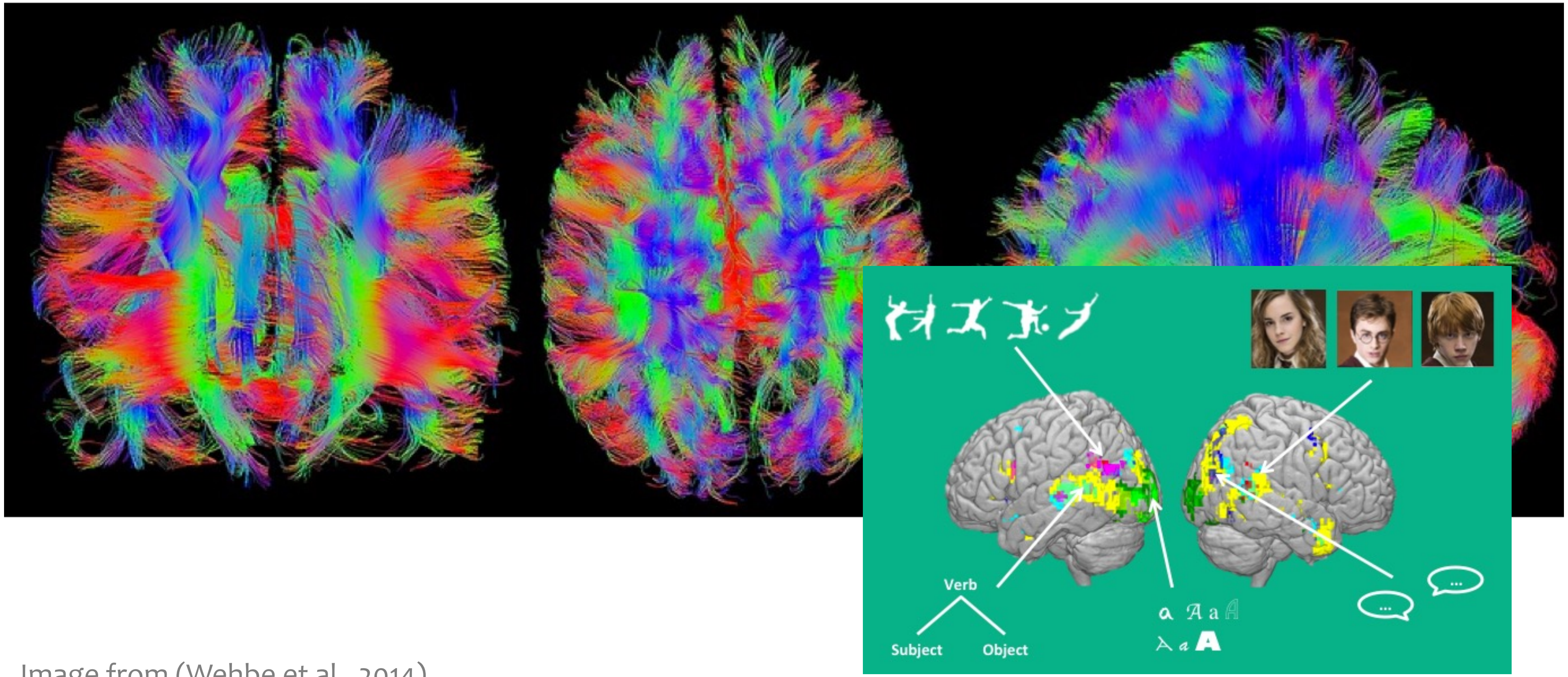


Image from (Wehbe et al., 2014)

Image from https://pixabay.com/en/brain-mrt-magnetic-resonance-imaging-1728449/

Learning Representations

Dimensionality Reduction Algorithms:

Powerful (often unsupervised) learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

Examples:

PCA, Kernel PCA, ICA, CCA, t-SNE, Autoencoders, VAEs

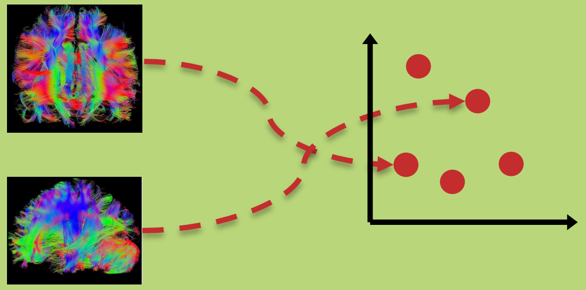
Useful for:

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)

Slide adapted from Nina Balcan

This section in one slide…

1. Dimensionality reduction:



2. Random Projection:

1. Randomly sample matrix: $\mathbf{V} \in \mathbb{R}^{K \times M}$
   $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down: $\underbrace{\mathbf{u}^{(i)}}_{K\times 1} = \underbrace{\mathbf{V}}_{K\times M} \underbrace{\mathbf{x}^{(i)}}_{M\times 1}$

3. Definition of PCA:

Choose the matrix V that either…

1. minimizes reconstruction error
2. consists of the K eigenvectors with largest eigenvalue

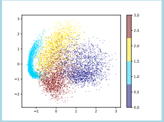
The above are equivalent definitions.

4. Algorithm for PCA:

The option we'll focus on:

Run Singular Value Decomposition (SVD) to obtain all the eigenvectors. Keep just the top-K to form V. Play some tricks to keep things efficient.

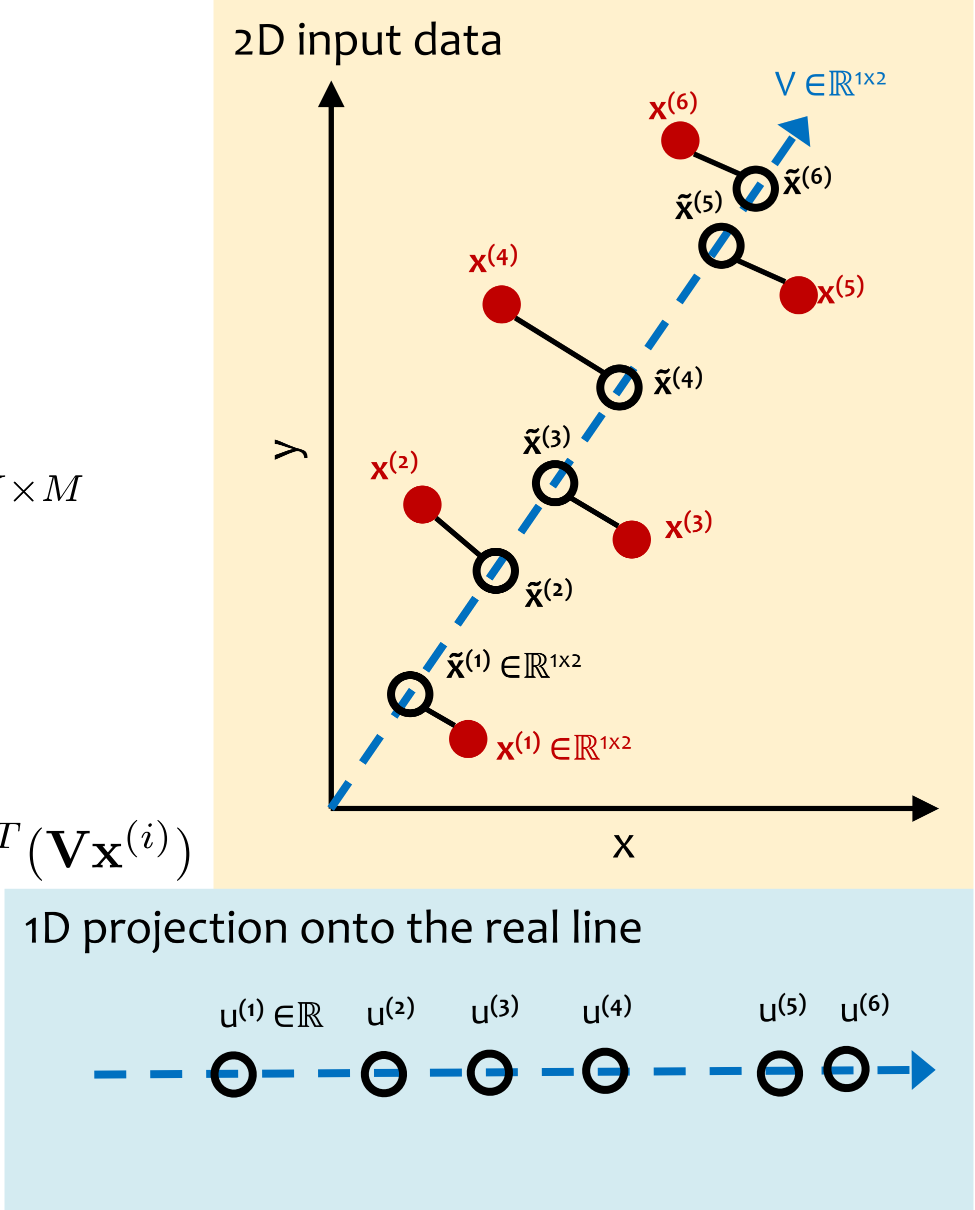
5. An Example



DIMENSIONALITY REDUCTION BY RANDOM PROJECTION

Random Projection

Example: 2D to 1D



Goal: project from M-dimensions down to K-dimensions

Data:

$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Algorithm:

1. Randomly sample matrix: $\mathbf{V} \in \mathbb{R}^{K \times M}$
   $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down: $\underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \underbrace{\mathbf{V}}_{K \times M} \underbrace{\mathbf{x}^{(i)}}_{M \times 1}$
3. Project up: $\underbrace{\tilde{\mathbf{x}}^{(i)}}_{M \times 1} = \underbrace{\mathbf{V}^T}_{M \times K} \underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \mathbf{V}^T(\mathbf{V}\mathbf{x}^{(i)})$

Random Projection

Goal: project from M-dimensions down to K-dimensions

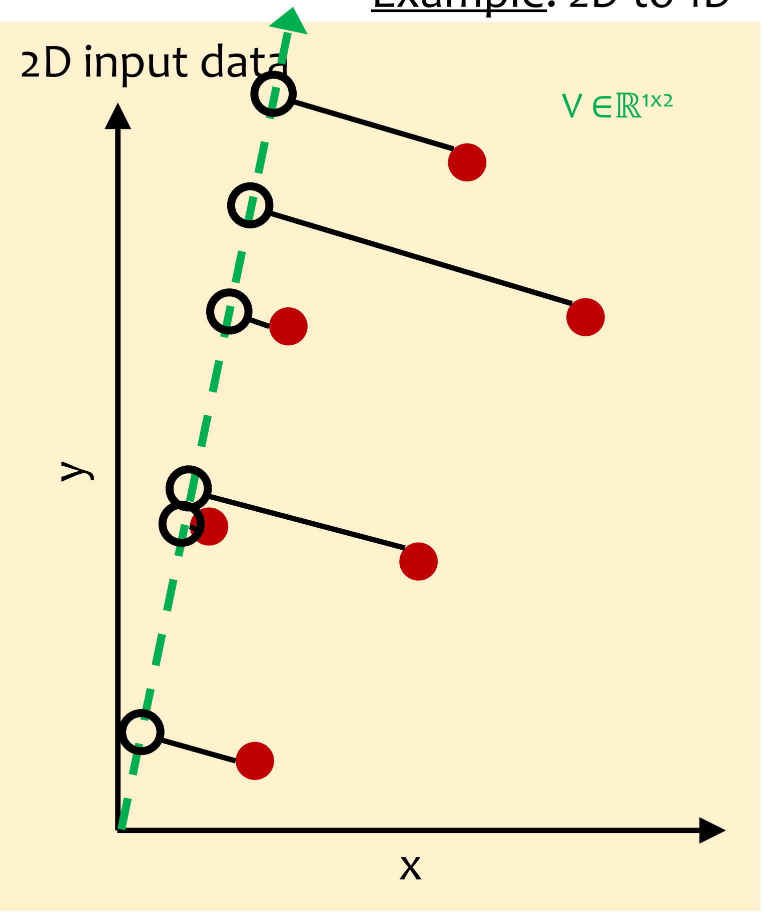
Data:

$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Algorithm:

1. Randomly sample matrix: $\mathbf{V} \in \mathbb{R}^{K \times M}$
   $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down: $\underbrace{\mathbf{u}^{(i)}}_{K\times 1} = \underbrace{\mathbf{V}}_{K \times M} \underbrace{\mathbf{x}^{(i)}}_{M \times 1}$
3. Project up: $\underbrace{\mathbf{x}^{(i)}}_{M\times 1} = \underbrace{\mathbf{V}^T}_{M \times K} \underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \mathbf{V}^T(\mathbf{V}\mathbf{x}^{(i)})$

Example: 2D to 1D



Problem: a random projection might give us a poor low dimensional representation of the data

Johnson-Lindenstrauss Lemma

Q: But how could we ever hope to preserve any useful information by randomly projecting into a low-dimensional space?

A: Even random projection enjoys some surprisingly impressive properties. In fact, a standard of the J-L lemma starts by assuming we have a random linear projection obtained by sampling each matrix entry from a Gaussian(0,1).

An Elementary Proof of a Theorem of Johnson and Lindenstrauss

Sanjoy Dasgupta,[1] Anupam Gupta[2]

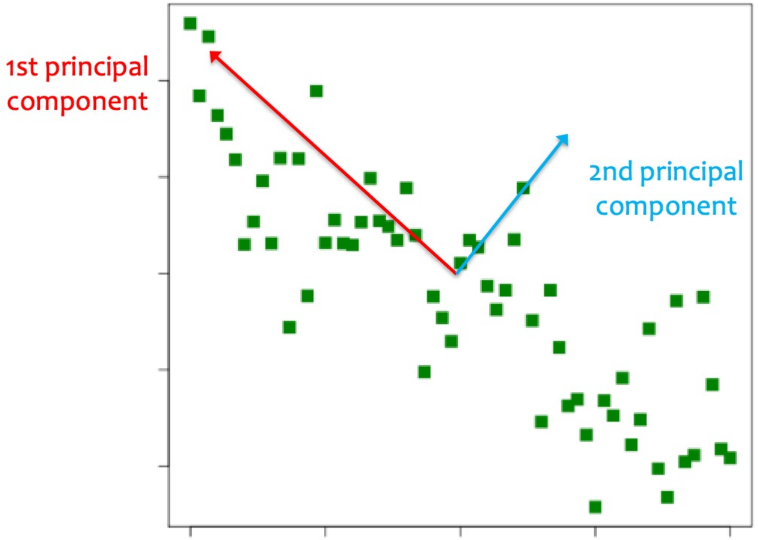
ABSTRACT: A result of Johnson and Lindenstrauss [13] shows that a set of n points in high dimensional Euclidean space can be mapped into an $O(\log n/\epsilon^2)$-dimensional Euclidean space such that the distance between any two points changes by only a factor of $(1 \pm \epsilon)$. In this note, we prove this theorem using elementary probabilistic techniques. Random Struct. Alg., 22: 60–65, 2002

http://www.cs.cmu.edu/~anupamg/papers/jl.pdf

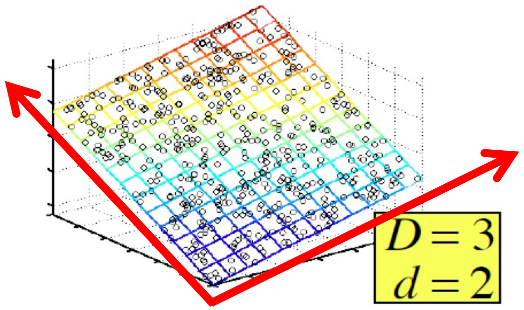
DEFINITION OF PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (PCA)

- **Assumption:** the data lies on a low K-dimensional linear subspace
- **Goal:** identify the axes of that subspace, and project each point onto hyperplane
- **Algorithm:** find the K eigenvectors with largest eigenvalue using classic matrix decomposition tools



PCA Example: 2D Gaussian Data



https://commons.wikimedia.org/wiki/File:Scatter_diagram_for_quality_characteristic_XXX.svg

Data for PCA

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$

$$\mathbf{x}^{(i)} \in \mathbb{R}^M$$

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(N)})^T \end{bmatrix}$$

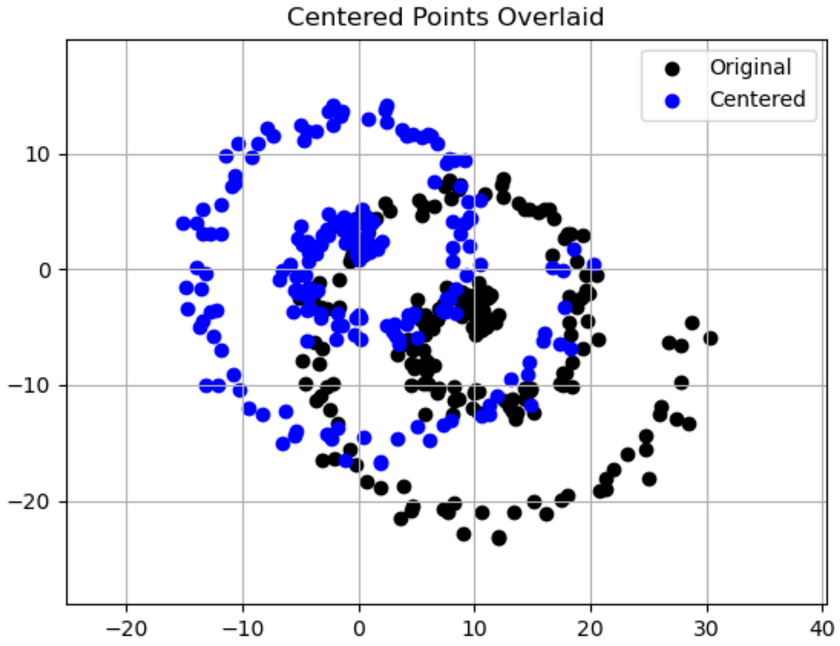
We assume the data is **centered**, i.e. the **sample mean** is zero

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{(i)} = \mathbf{0}$$

Q: What if your data is **not** centered?

A: Subtract off the sample mean

$$\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}, \forall i$$



Sample Covariance Matrix

Background: Sample Variance

Suppose we have a sequence of random samples $\{x^{(1)}, \ldots, x^{(N)}\}$ from a random variable X.

The (biased) **sample variance** $\hat{\sigma}^2$ is given by:

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \hat{\mu})^2$$

where $\hat{\mu}$ is the sample mean.

The **sample covariance matrix** $\mathbf{\Sigma} \in \mathbb{R}^{M \times M}$ is given by:

$$\Sigma_{jk} = \frac{1}{N} \sum_{i=1}^{N} (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$$

Since the data matrix is centered, we can rewrite as:

$$\mathbf{\Sigma} = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

where

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(N)})^T \end{bmatrix}$$

Principal Component Analysis (PCA)

Linear Projection:
Given KxM matrix **V**, and Mx1 vector $\mathbf{x}^{(i)}$ we obtain the Kx1 projection $\mathbf{u}^{(i)}$ by:

$$\mathbf{u}^{(i)} = \mathbf{V}\,\mathbf{x}^{(i)}$$

$$\mathbf{V} = \begin{bmatrix} — \mathbf{v}_1^T — \\ — \mathbf{v}_2^T — \\ \vdots \\ — \mathbf{v}_K^T — \end{bmatrix}$$

$$\mathbf{u}^{(i)} = \begin{bmatrix} u_1^{(i)} \\ u_2^{(i)} \\ \vdots \\ u_k^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1^\top \mathbf{x}^{(i)} \\ \mathbf{v}_2^\top \mathbf{x}^{(i)} \\ \vdots \\ \mathbf{v}_k^\top \mathbf{x}^{(i)} \end{bmatrix} = \mathbf{V}\mathbf{x}^{(i)}$$

Definition of PCA:

PCA repeatedly chooses a next vector $\mathbf{v}_j$ that minimizes the reconstruction error s.t. $\mathbf{v}_j$ is orthogonal to $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{j-1}$.

Vector $\mathbf{v}_j$ is called the **jth principal component.**

Notice: Two vectors **a** and **b** are **orthogonal** if $\mathbf{a}^T\mathbf{b} = 0$.
➔the K-dimensions in PCA are uncorrelated

Vector Projection

length of projection of $\mathbf{x}$ onto $\mathbf{v}$

$$a = \frac{(\mathbf{v}^\top \mathbf{x})}{\|\mathbf{v}\|_2^2}$$

if $\|\mathbf{v}\|_2 = 1$

otherwise

projection of $\mathbf{x}$ onto $\mathbf{v}$

$$\mathbf{u} = \frac{(\mathbf{v}^\top \mathbf{x})\mathbf{v}}{\|\mathbf{v}\|_2^2}$$

if $\|\mathbf{v}\|_2 = 1$

otherwise

Objectives for PCA

Minimize the Reconstruction Error

Maximize the Variance

Objectives for PCA

Minimize the Reconstruction Error

$$\mathbf{v}_1 = \underset{\mathbf{v}}{\operatorname{argmin}} \; \frac{1}{N}\sum_{i=1}^{N} \text{distance}\left(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}\right)^2$$

$$= \underset{\mathbf{v}}{\operatorname{argmin}} \; \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - \left(\begin{array}{c} \text{vector projection} \\ \text{of } \mathbf{x}^{(i)} \text{ onto } \mathbf{v} \end{array} \right) \right\|_2^2$$

$$= \underset{\mathbf{v} \text{ s.t. } \|\mathbf{v}\|_2 = 1}{\operatorname{argmin}} \; \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - (\mathbf{v}^\top \mathbf{x}^{(i)})\mathbf{v} \right\|_2^2$$

Maximize the Variance

$$\mathbf{v}_1 = \underset{\mathbf{v}}{\operatorname{argmax}} \; \frac{1}{N}\sum_{i=1}^{N} \left(\begin{array}{c} \text{length of vector} \\ \text{projection of } \mathbf{x}^{(i)} \text{ onto } \mathbf{v} \end{array} \right)^2$$

$$= \underset{\mathbf{v} \text{ s.t. } \|\mathbf{v}\|_2 = 1}{\operatorname{argmax}} \; \frac{1}{N}\sum_{i=1}^{N} \left(\mathbf{v}^\top \mathbf{x}^{(i)}\right)^2$$

$$= \underset{\mathbf{v} \text{ s.t. } \|\mathbf{v}\|_2 = 1}{\operatorname{argmax}} \; \frac{1}{N} \left(\mathbf{v}^\top \mathbf{X}^\top\right)(\mathbf{X}\mathbf{v})$$

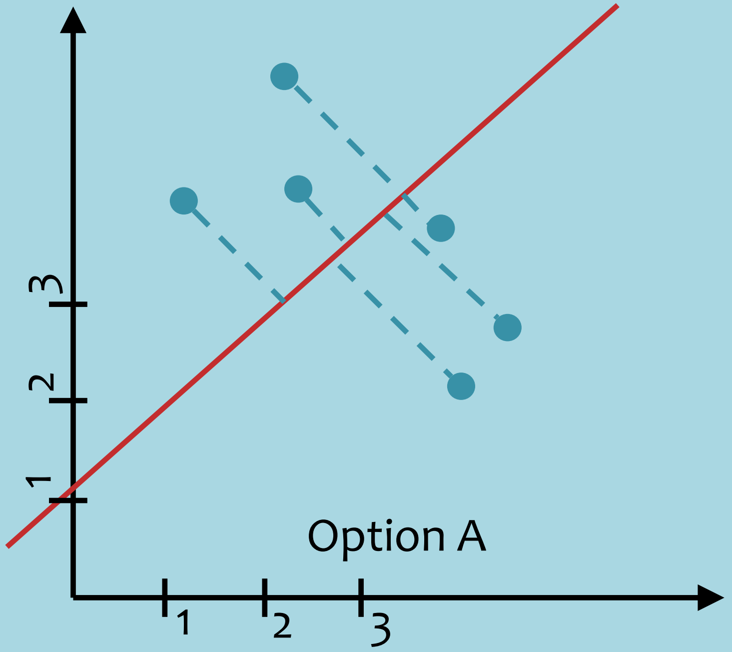
$$= \underset{\mathbf{v} \text{ s.t. } \|\mathbf{v}\|_2 = 1}{\operatorname{argmax}} \; \mathbf{v}^\top \boldsymbol{\Sigma} \mathbf{v}$$

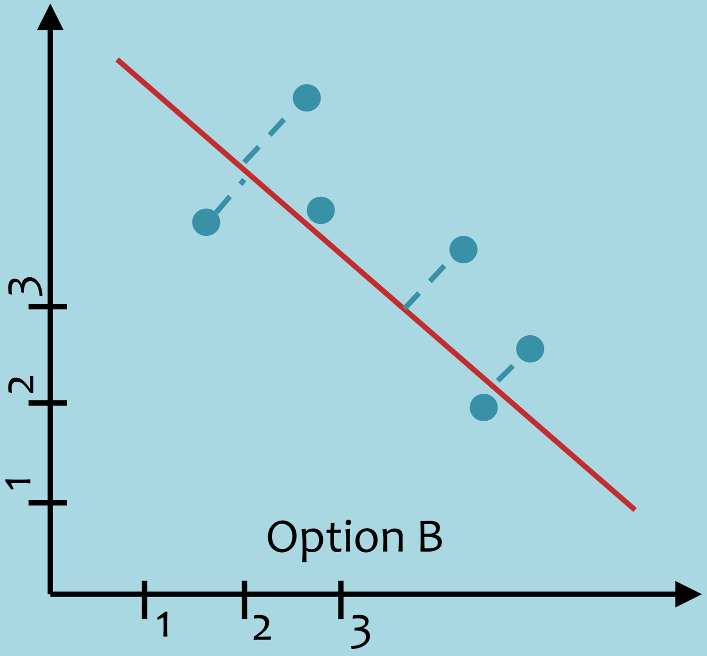
Projection Example

Below are two plots of the same dataset D. Consider the two projections shown.

1. **Poll Question 2:** Which maximizes the variance?
2. **Poll Question 3:** Which minimizes the reconstruction error?

Answer:





PCA Objective Functions

What is the first principal component $\mathbf{v}_1$ chosen by PCA?

Option 1: The vector that *minimizes* the **reconstruction error**

$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}^{(i)} - (\mathbf{v}^T \mathbf{x}^{(i)})\mathbf{v}||^2$$

Option 2: The vector that *maximizes* the **variance**

$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{v}^T \mathbf{x}^{(i)})^2$$

Equivalence of Maximizing Variance and Minimizing Reconstruction Error

PCA

Claim: Minimizing the reconstruction error is equivalent to maximizing the variance.

Proof: First, note that:

$$||\mathbf{x}^{(i)} - (\mathbf{v}^T\mathbf{x}^{(i)})\mathbf{v}||^2 = ||\mathbf{x}^{(i)}||^2 - (\mathbf{v}^T\mathbf{x}^{(i)})^2 \tag{1}$$

since $\mathbf{v}^T\mathbf{v} = ||\mathbf{v}||^2 = 1$.

Substituting into the minimization problem, and removing the extraneous terms, we obtain the maximization problem.

$$\mathbf{v}^* = \operatorname*{argmin}_{\mathbf{v}:||\mathbf{v}||^2=1} \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}^{(i)} - (\mathbf{v}^T\mathbf{x}^{(i)})\mathbf{v}||^2 \tag{2}$$

$$= \operatorname*{argmin}_{\mathbf{v}:||\mathbf{v}||^2=1} \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}^{(i)}||^2 - (\mathbf{v}^T\mathbf{x}^{(i)})^2 \tag{3}$$

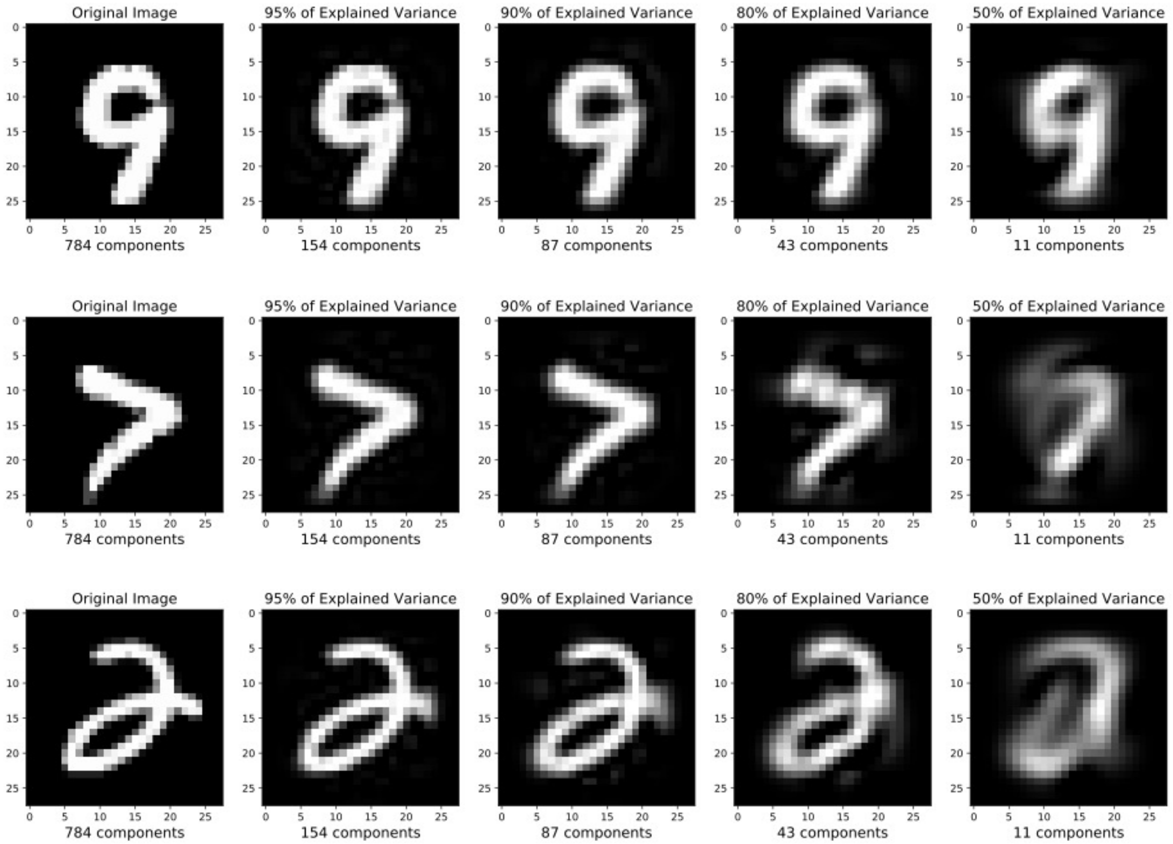
$$= \operatorname*{argmax}_{\mathbf{v}:||\mathbf{v}||^2=1} \frac{1}{N}\sum_{i=1}^{N} (\mathbf{v}^T\mathbf{x}^{(i)})^2 \tag{4}$$

PCA EXAMPLES

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Report percent of variance explained for K components
3. Then project back up to 28x28 image (i.e. a vector $\tilde{\mathbf{x}}^{(i)}$ of length 784) to visualize how much information was preserved



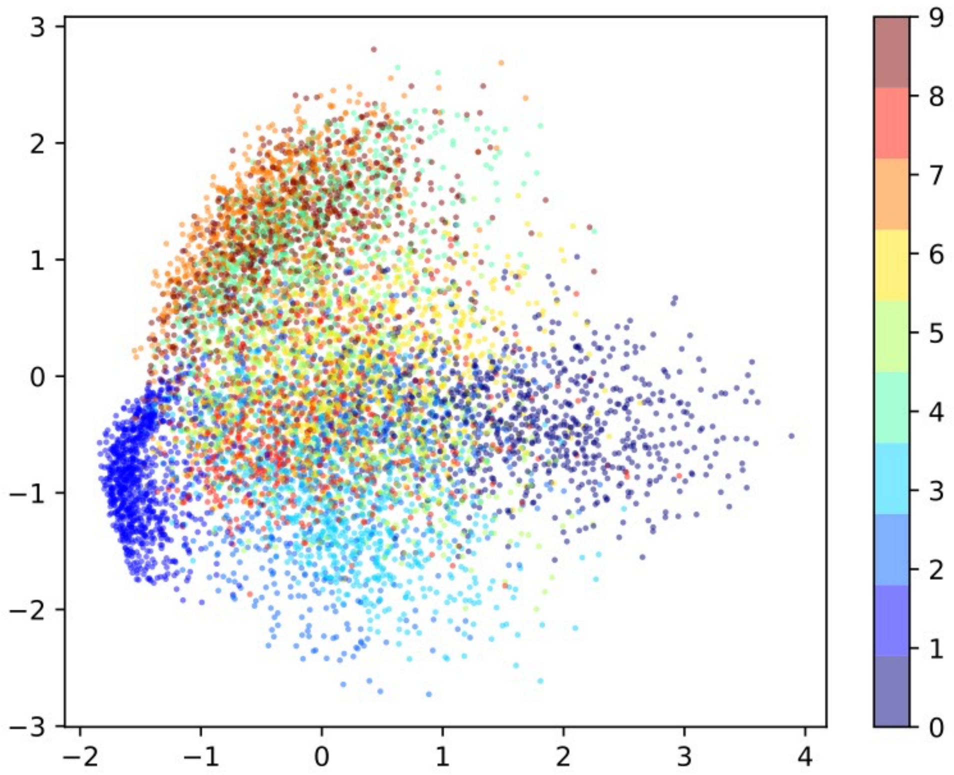
Takeaway: Using fewer principal components K leads to higher reconstruction error.

But even a small number (say 43) still preserves a lot of information about the original image.

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K=2 components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Plot the 2 dimensional points $\mathbf{u}^{(i)}$ and label with the (unknown to PCA) label $y^{(i)}$ as the color
3. Here we look at all ten digits 0 - 9

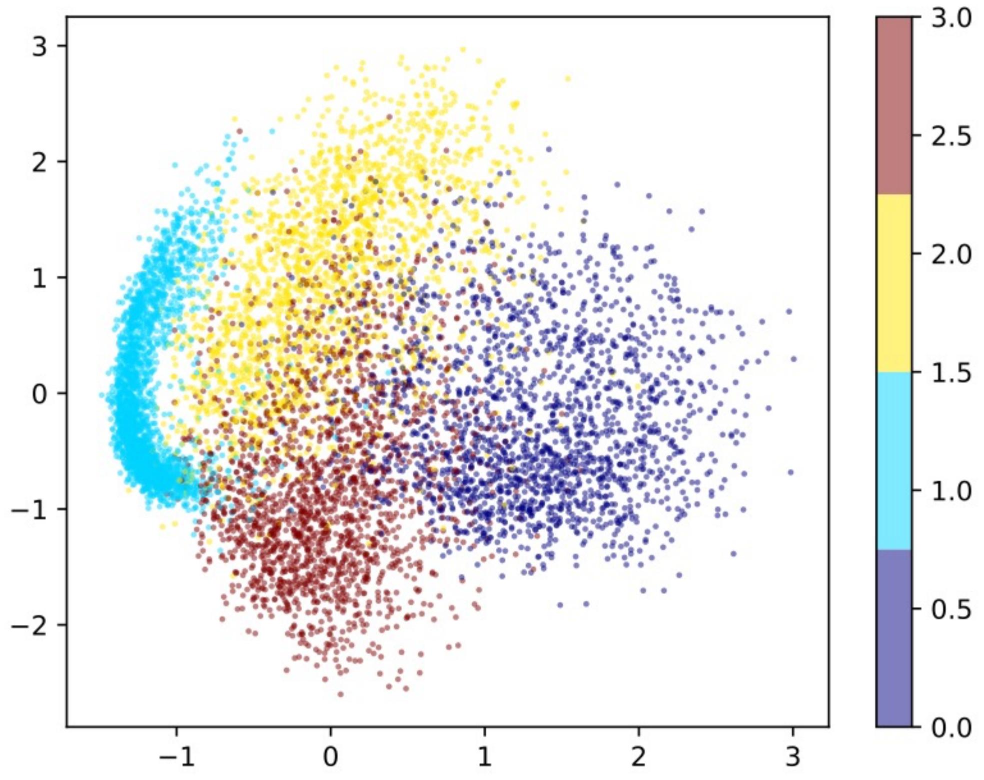


Takeaway: Even with a tiny number of principal components K=2, PCA learns a representation that captures the *latent* information about the type of digit

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K=2 components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Plot the 2 dimensional points $\mathbf{u}^{(i)}$ and label with the (unknown to PCA) label $y^{(i)}$ as the color
3. Here we look at just four digits 0, 1, 2, 3



Takeaway: Even with a tiny number of principal components K=2, PCA learns a representation that captures the *latent* information about the type of digit

Learning Objectives

Dimensionality Reduction / PCA

You should be able to…

1. Define the sample mean, sample variance, and sample covariance of a vector-valued dataset
2. Identify examples of high dimensional data and common use cases for dimensionality reduction
3. Draw the principal components of a given toy dataset
4. Establish the equivalence of minimization of reconstruction error with maximization of variance
5. Given a set of principal components, project from high to low dimensional space and do the reverse to produce a reconstruction
6. Explain the connection between PCA, eigenvectors, eigenvalues, and covariance matrix
7. Use common methods in linear algebra to obtain the principal components