10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

K-Means

+

GMMs

Clustering Readings:
Murphy 25.5
Bishop 12.1, 12.3
HTF 14.3.0
Mitchell --

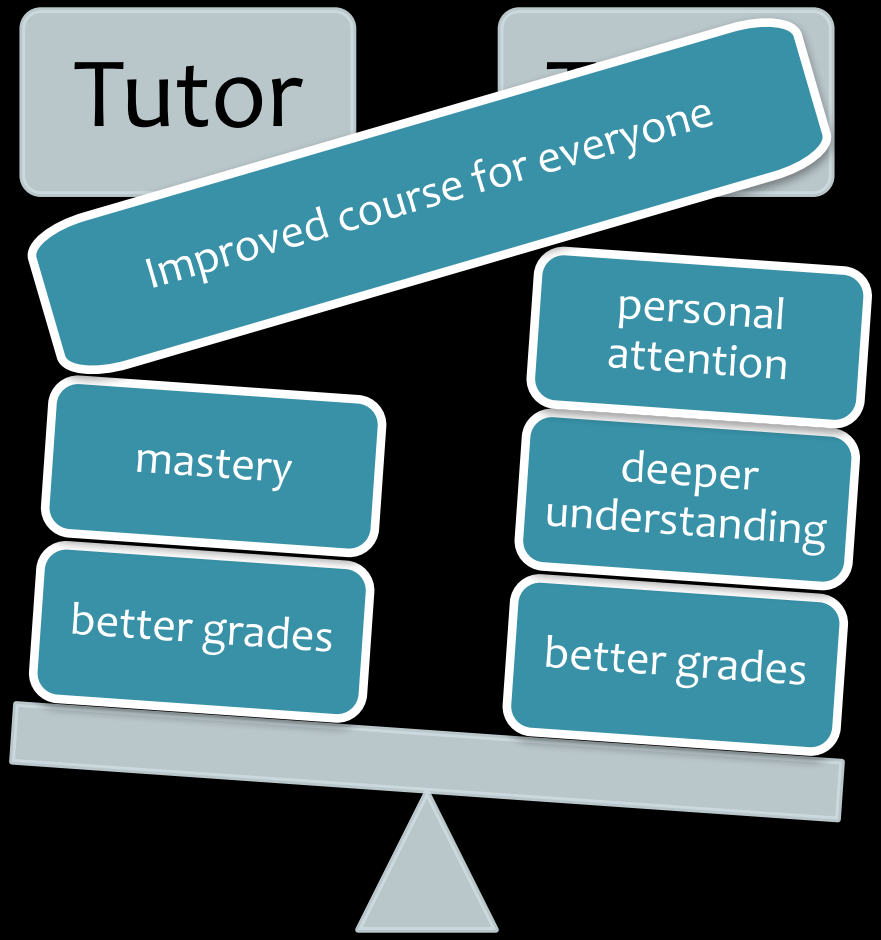
EM, GMM Readings:
Murphy 11.4.1, 11.4.2, 11.4.4
Bishop 9
HTF 8.5 - 8.5.3
Mitchell 6.12 - 6.12.2

Matt Gormley
Lecture 16
March 20, 2017

Reminders

- **Homework 5: Readings / Application of ML**
  - **Release: Wed, Mar. 08**
  - **Due: Wed, Mar. 22 at 11:59pm**

Peer Tutoring



K-MEANS

K-Means Outline

- **Clustering: Motivation / Applications**
- **Optimization Background**
  - Coordinate Descent
  - Block Coordinate Descent
- **Clustering**
  - Inputs and Outputs
  - Objective-based Clustering
- **K-Means**
  - K-Means Objective
  - Computational Complexity
  - K-Means Algorithm / Lloyd's Method
- **K-Means Initialization**
  - Random
  - Farthest Point
  - K-Means++

Last Lecture

This Lecture

Clustering, Informal Goals

Goal: Automatically partition unlabeled data into groups of similar datapoints.

Question: When and why would we want to do this?

Useful for:

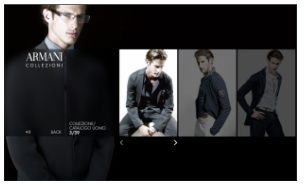
- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
    - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

Slide courtesy of Nina Balcan

Applications (Clustering comes up everywhere…)

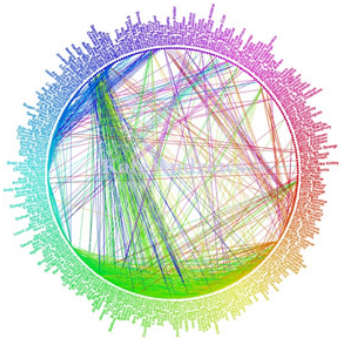
- Cluster news articles or web pages or search results by topic.



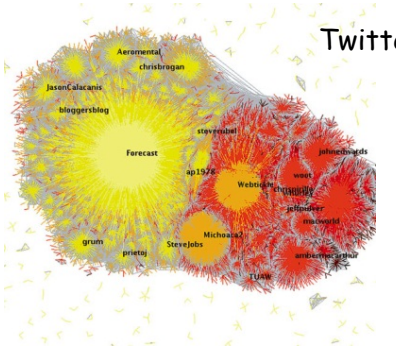


- Cluster protein sequences by function or genes according to expression profile.


- Cluster users of social networks by interest (community detection).



Facebook network

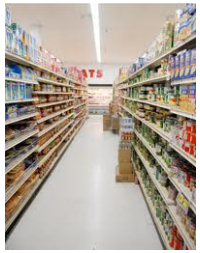


Twitter Network

Slide courtesy of Nina Balcan

Applications (Clustering comes up everywhere…)

- Cluster customers according to purchase history.



- Cluster galaxies or nearby stars (e.g. Sloan Digital Sky Survey)

- And many many more applications….

Slide courtesy of Nina Balcan

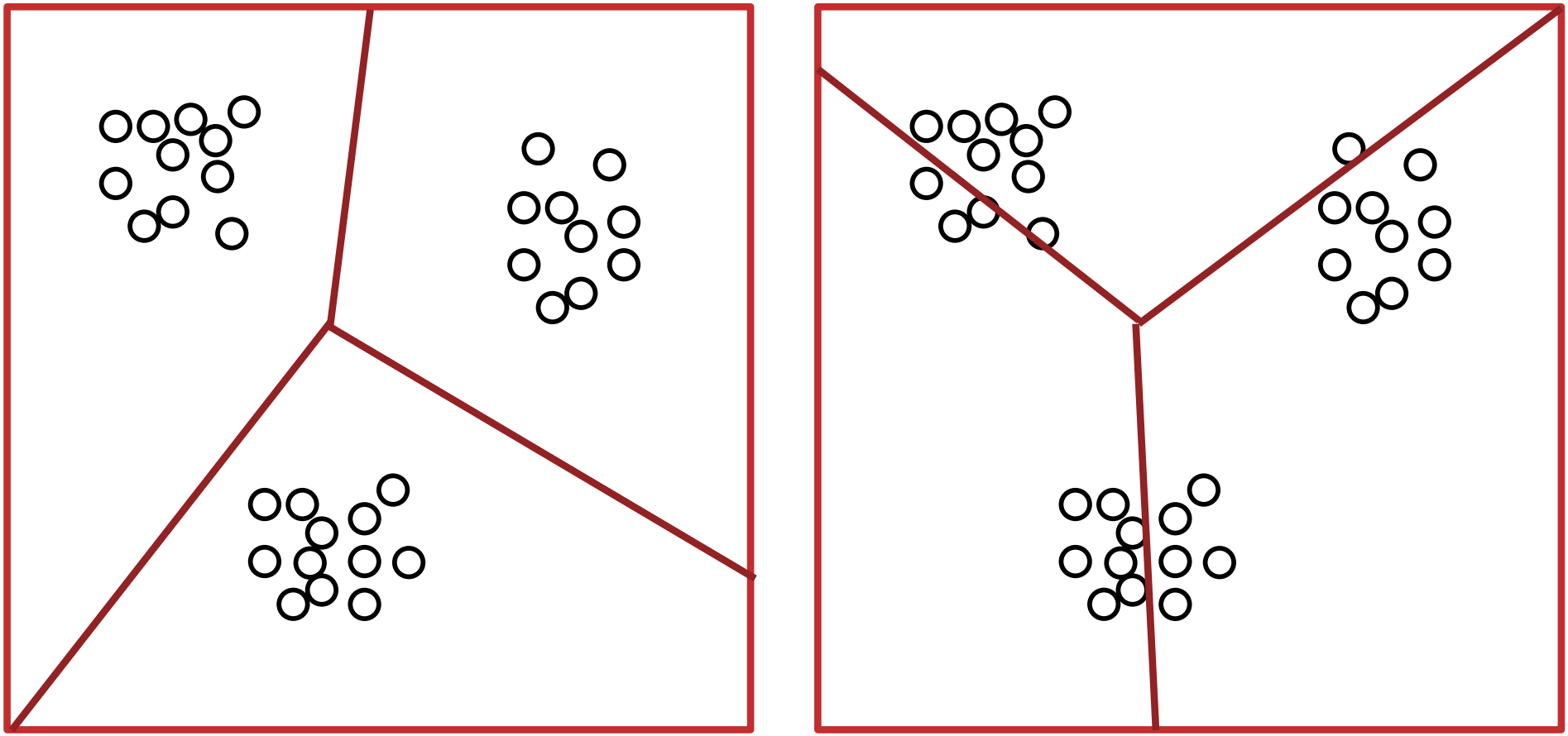
Optimization Background

Whiteboard:

- Coordinate Descent
- Block Coordinate Descent

Clustering

Question: Which of these partitions is “better”?



Clustering

Whiteboard:

- Inputs and Outputs
- Objective-based Clustering

K-Means

Whiteboard:

- K-Means Objective
- Computational Complexity
- K-Means Algorithm / Lloyd's Method

K-Means Initialization

Whiteboard:

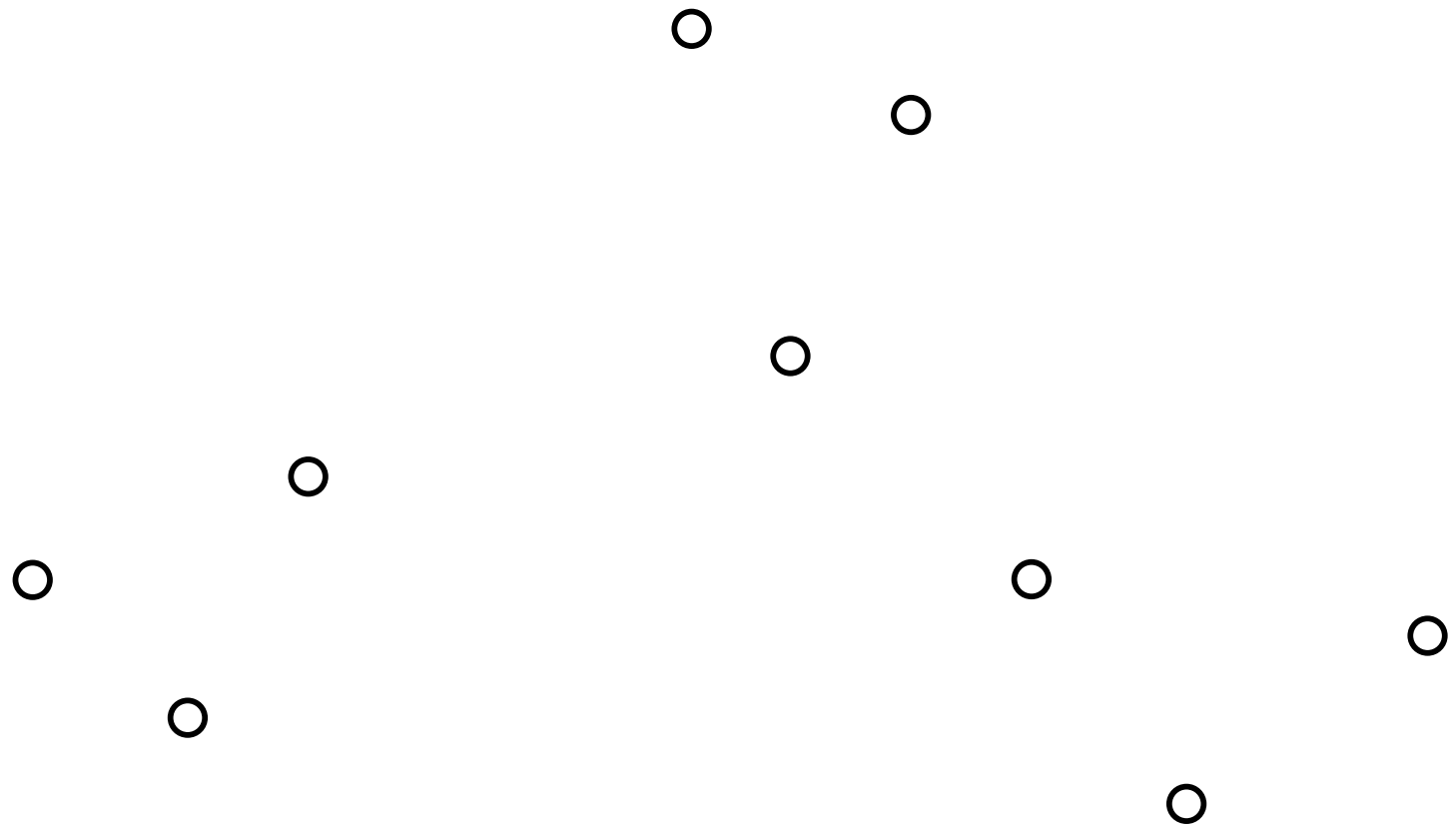
- Random
- Furthest Traversal
- K-Means++

Lloyd's method: Random Initialization

Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

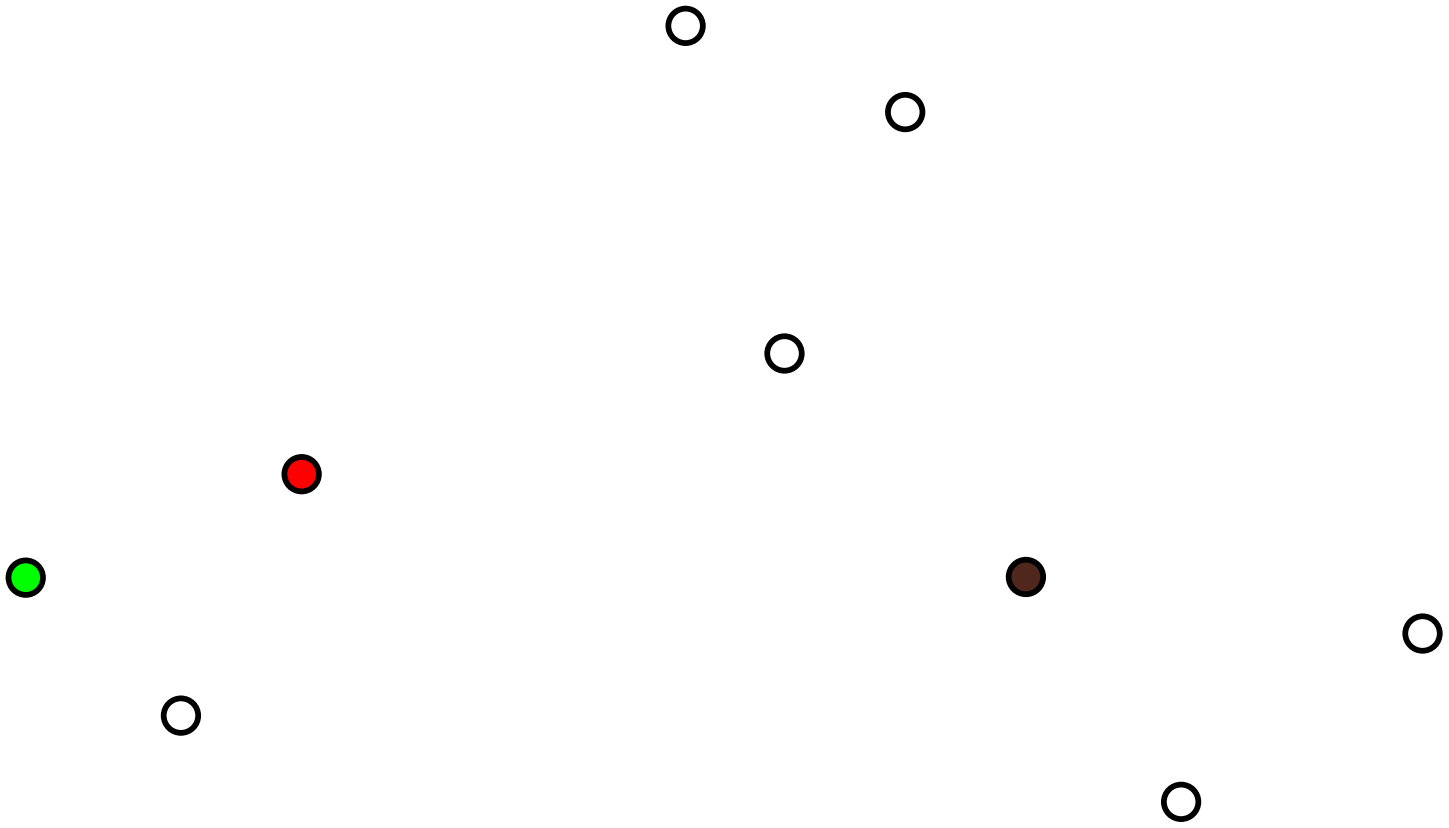
Example: Given a set of datapoints



Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

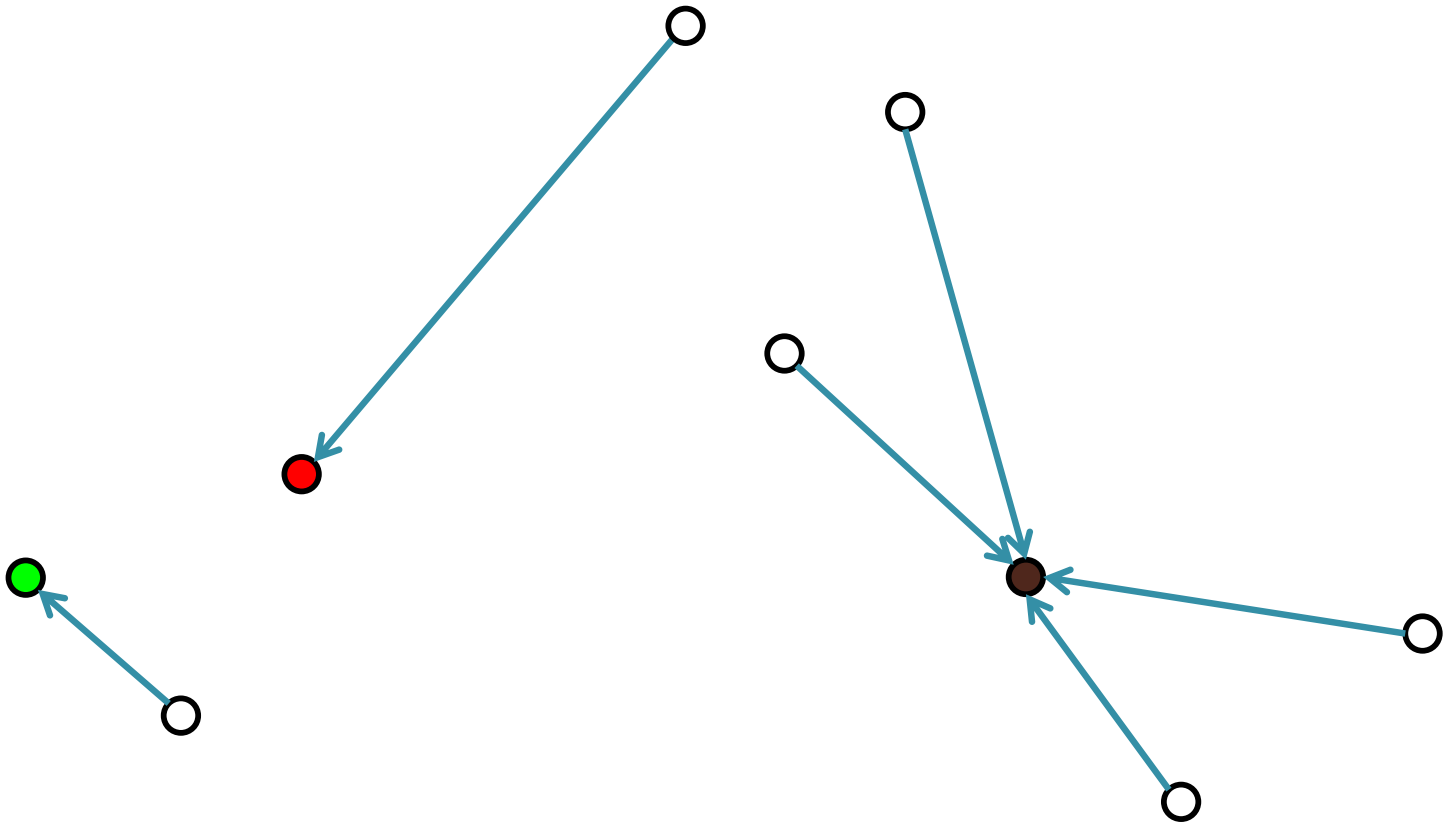
Select initial centers at random



Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

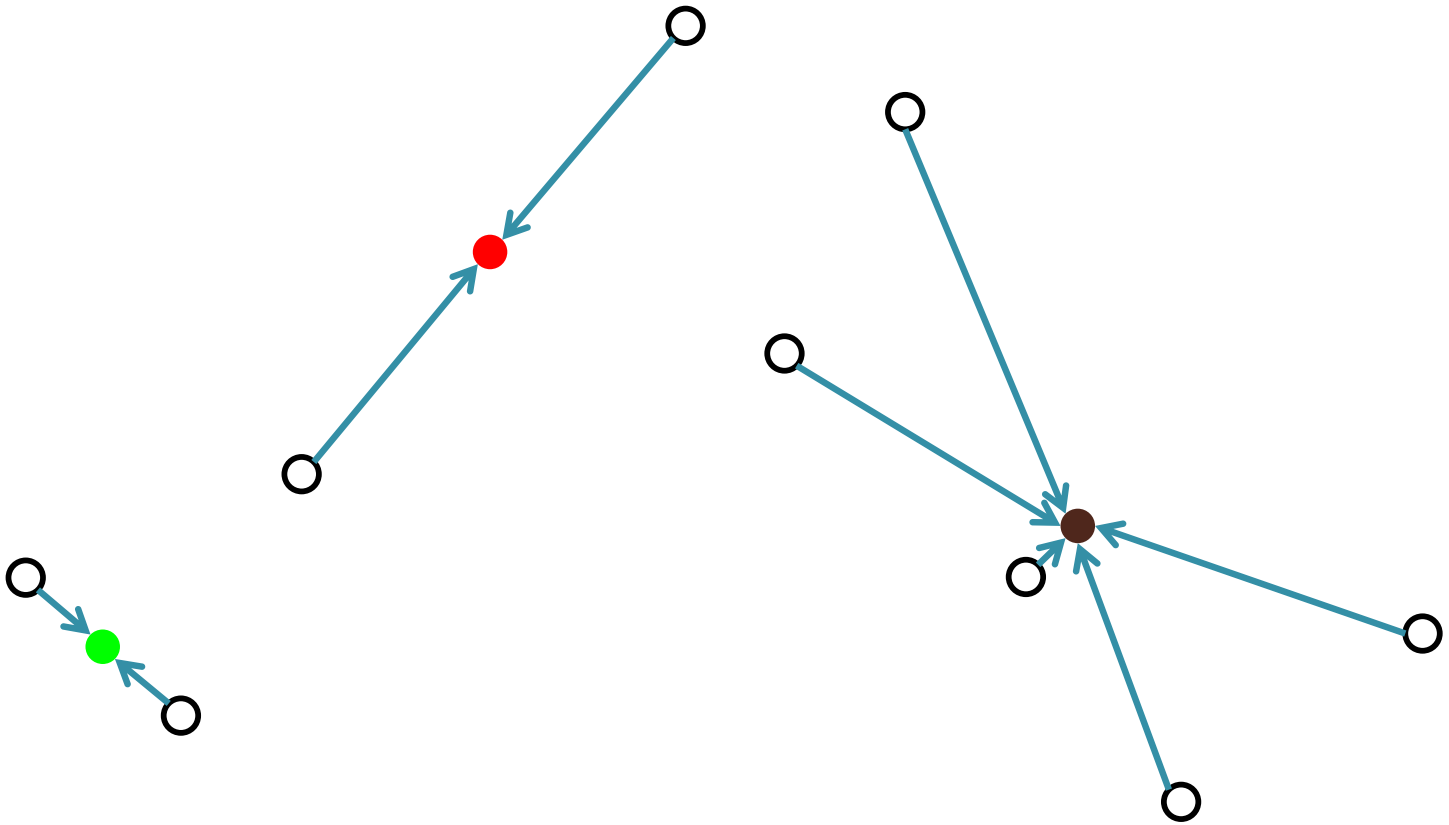
Assign each point to its nearest center



Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

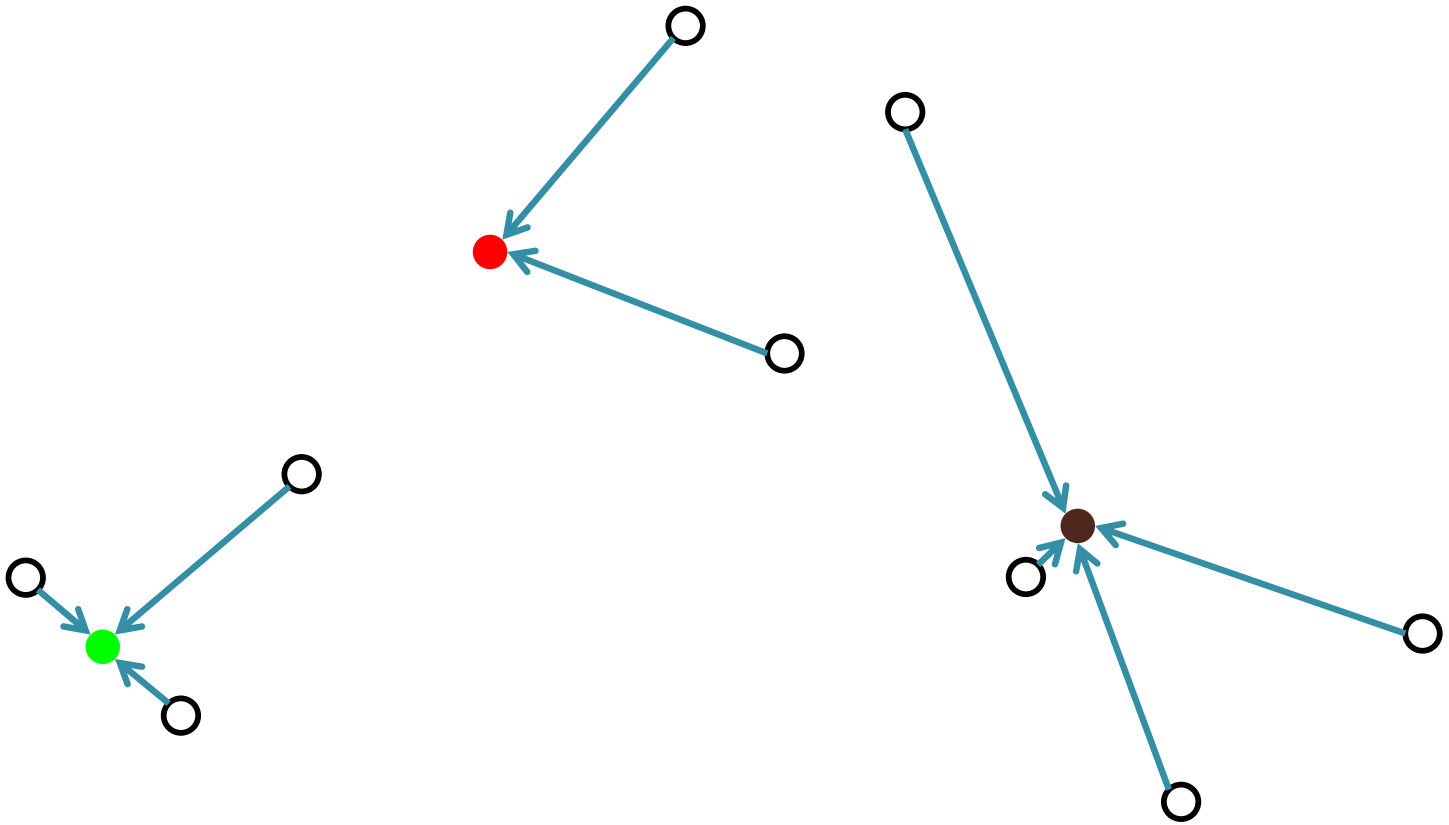
Recompute optimal centers given a fixed clustering



Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

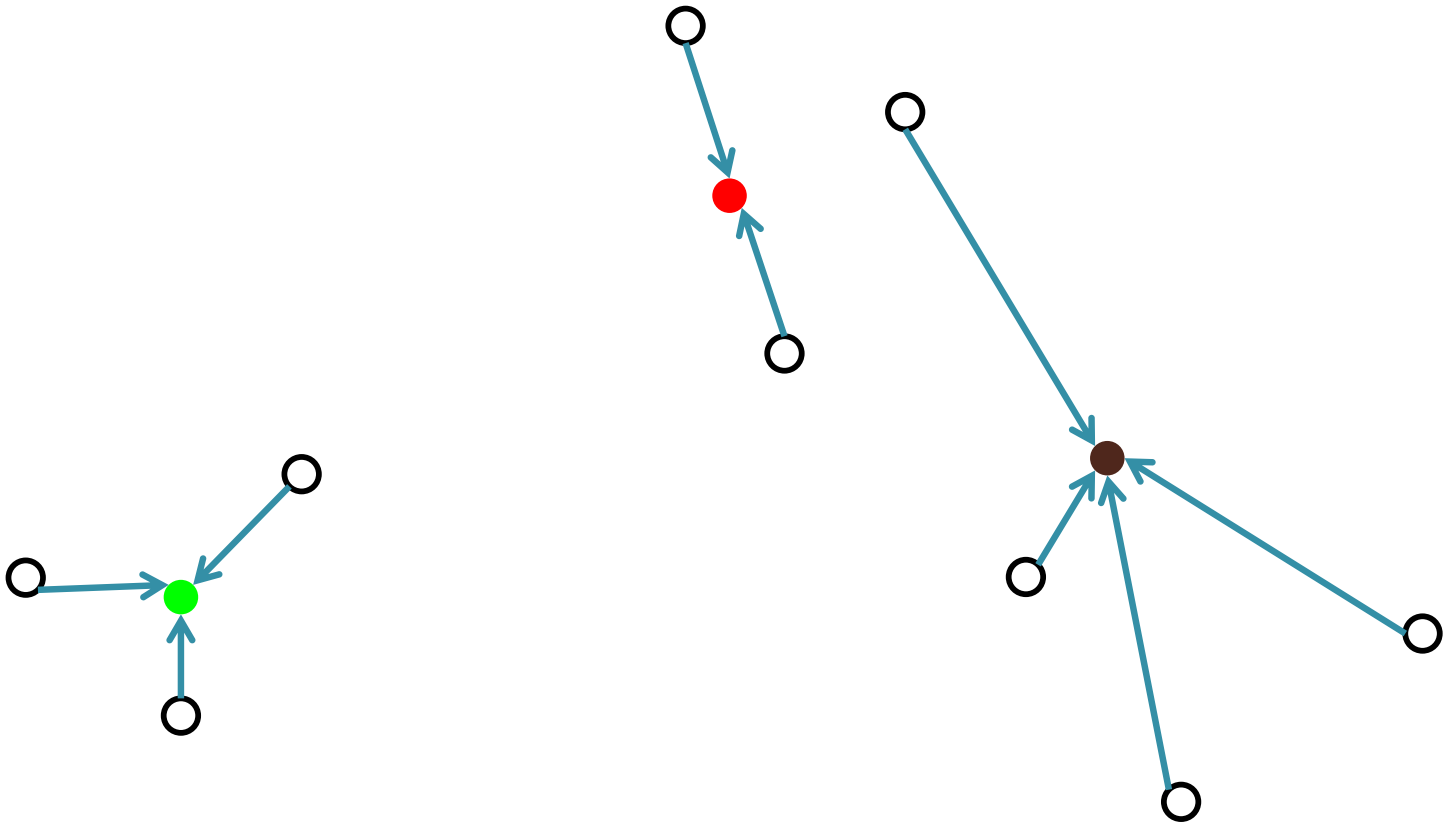
Assign each point to its nearest center



Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

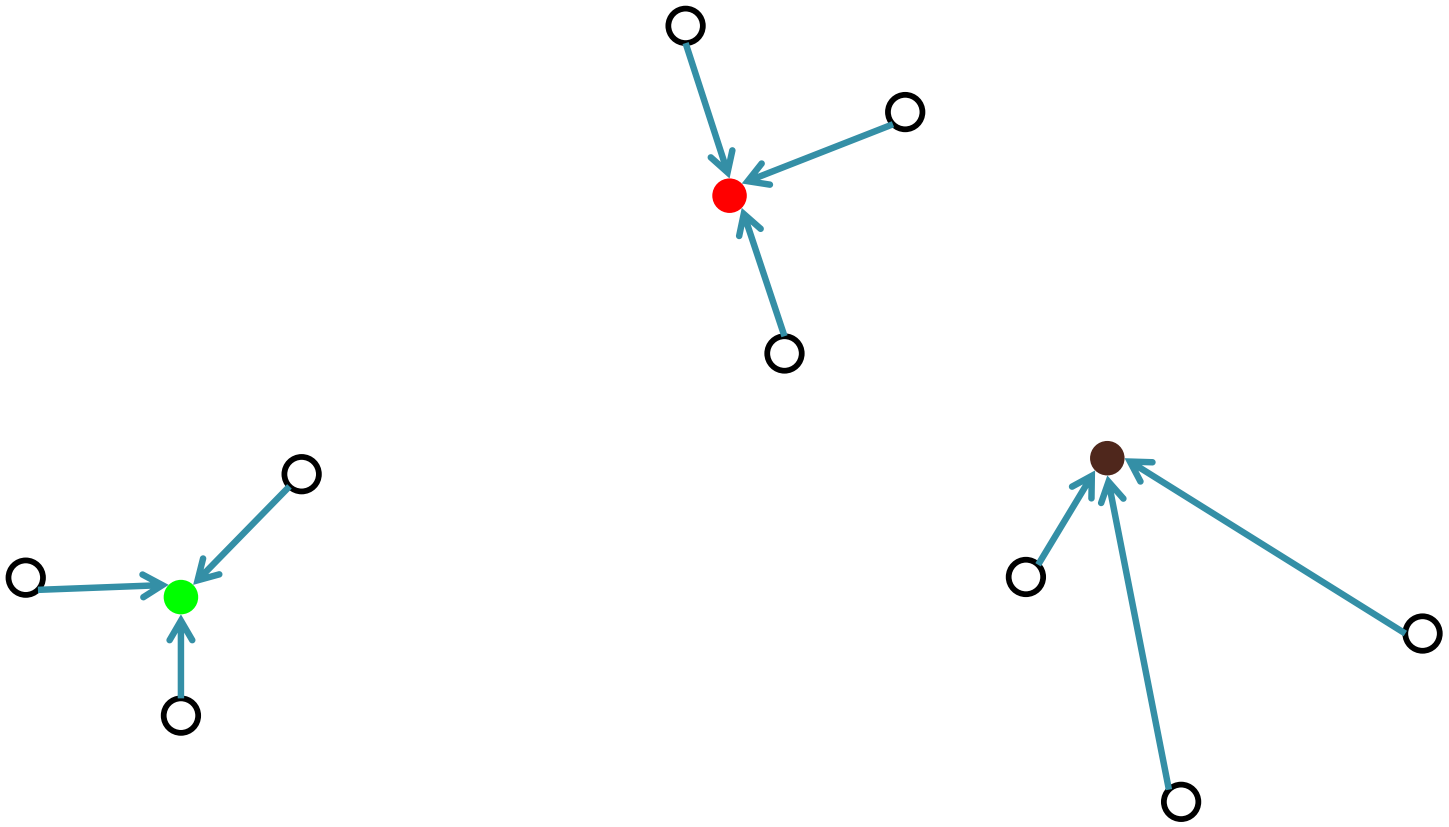
Recompute optimal centers given a fixed clustering



Slide courtesy of Nina Balcan

Lloyd's method: Random Initialization

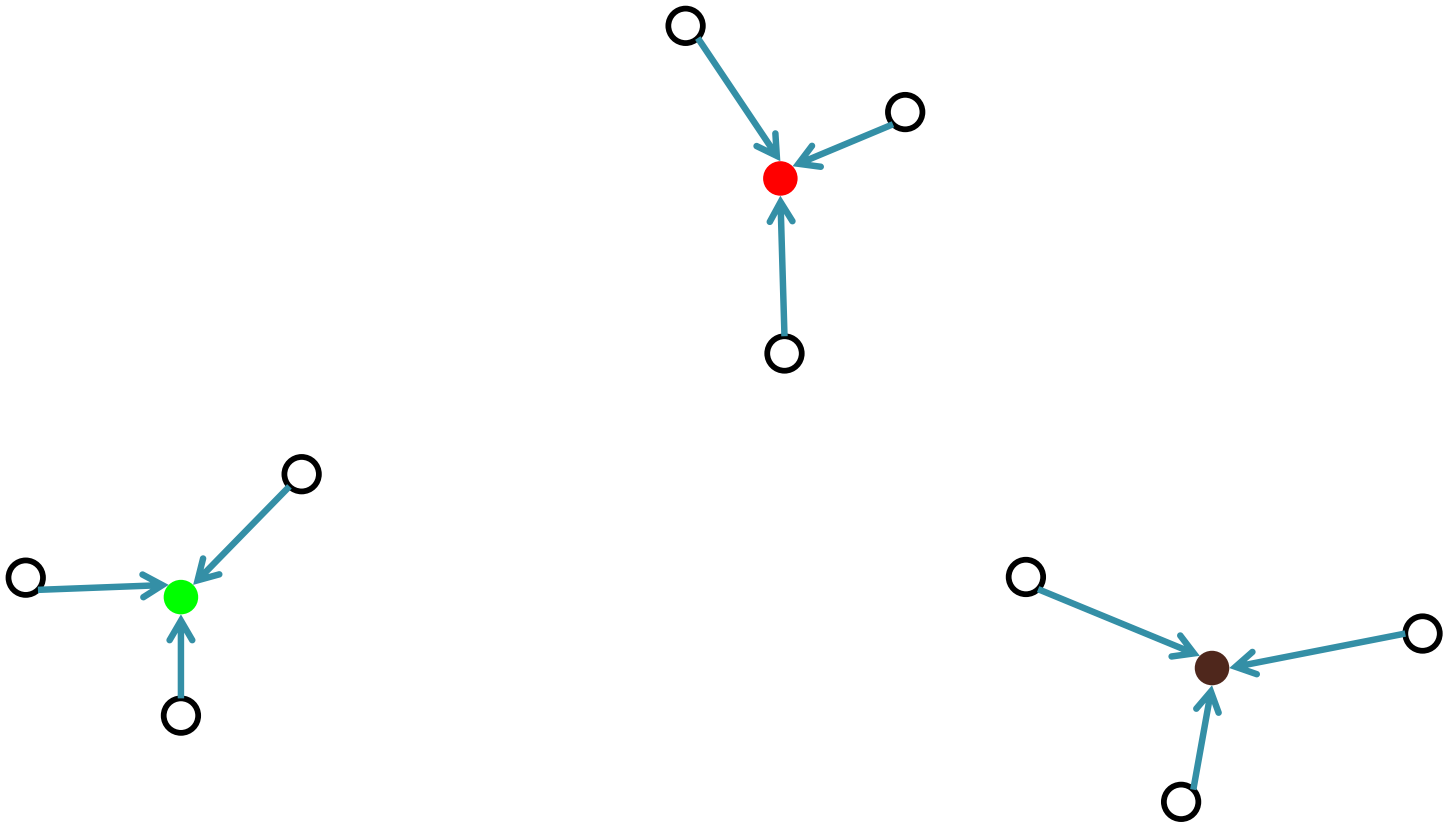
Assign each point to its nearest center



Slide courtesy of Nina Balcan

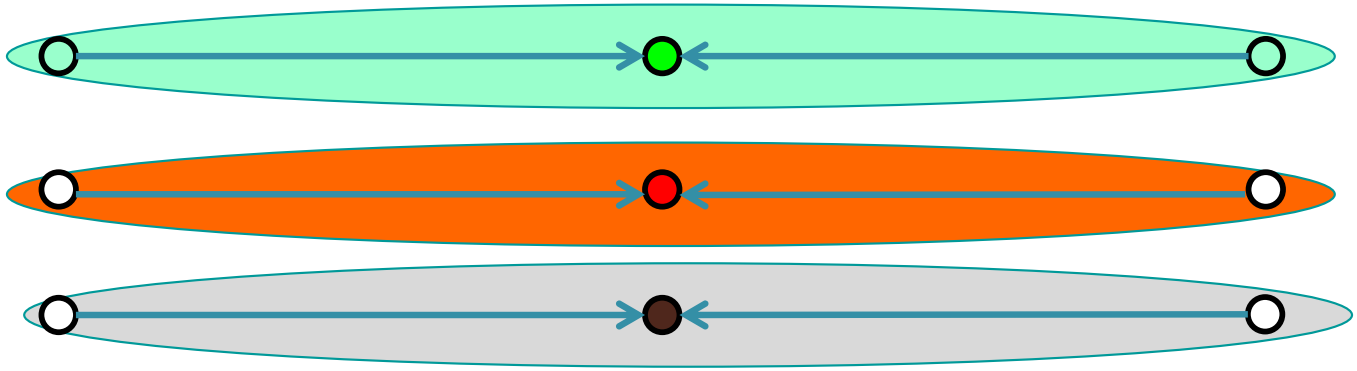
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



Get a good quality solution in this example.

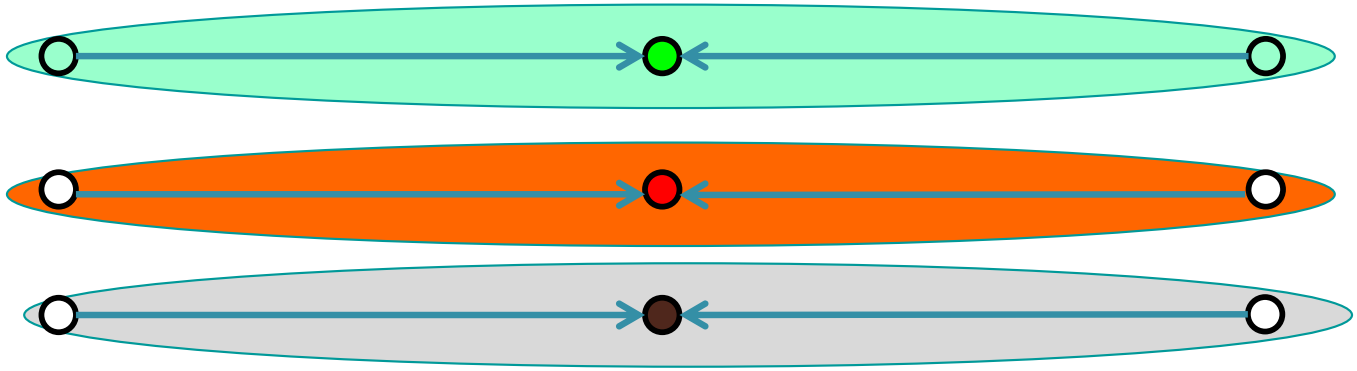
Lloyd's method: Performance



It always converges, but it may converge at a local optimum that is different from the global optimum, and in fact could be arbitrarily worse in terms of its score.

Slide courtesy of Nina Balcan

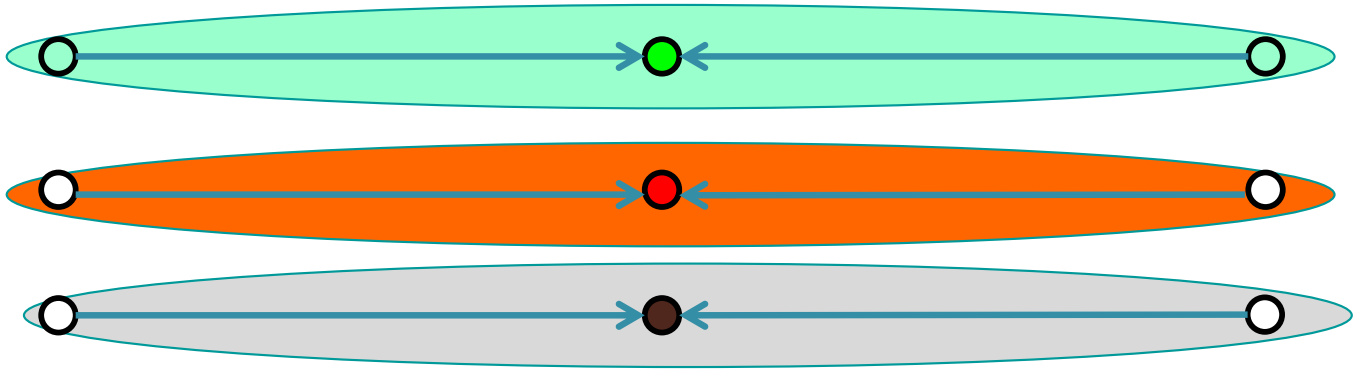
Lloyd's method: Performance



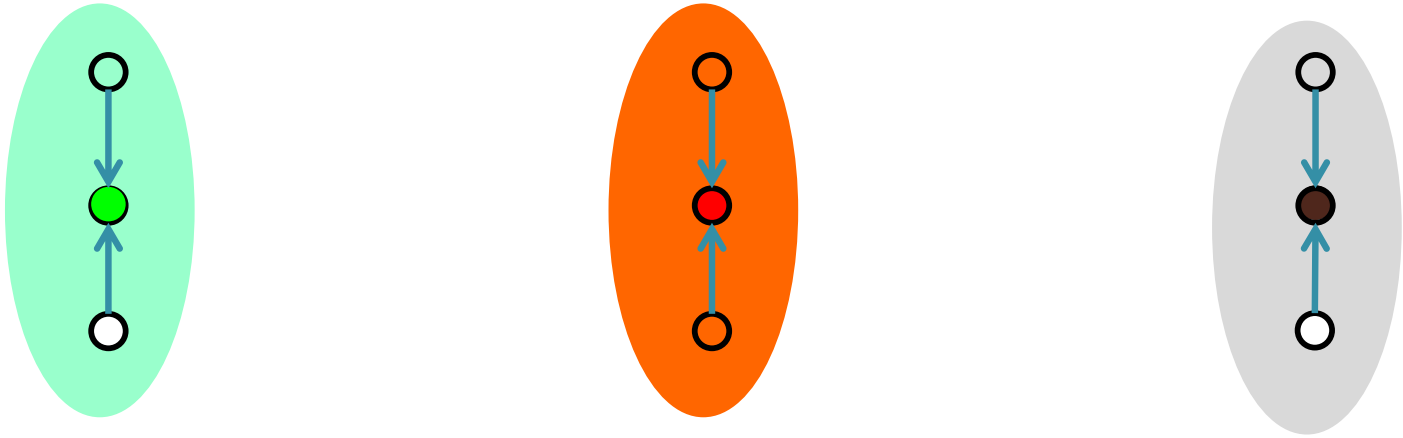
Local optimum: every point is assigned to its nearest center and every center is the mean value of its points.

Slide courtesy of Nina Balcan

Lloyd's method: Performance

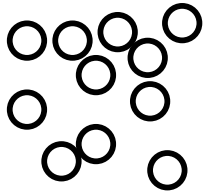


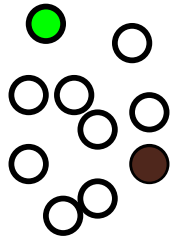
.It is arbitrarily worse than optimum solution….

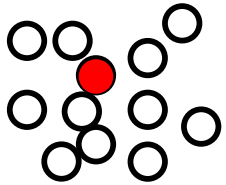


Slide courtesy of Nina Balcan

Lloyd's method: Performance



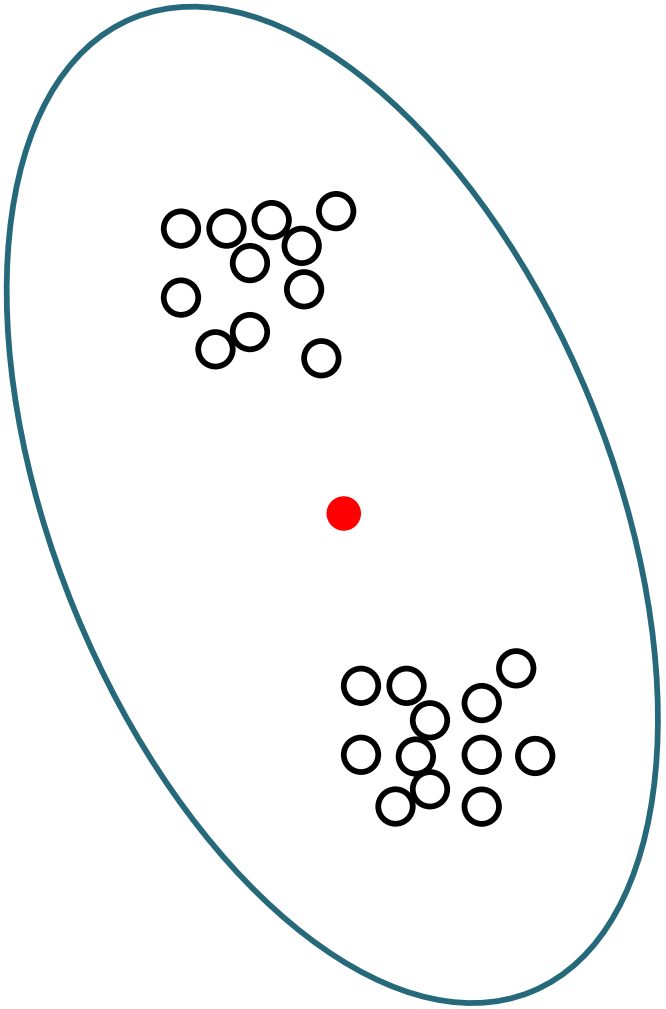


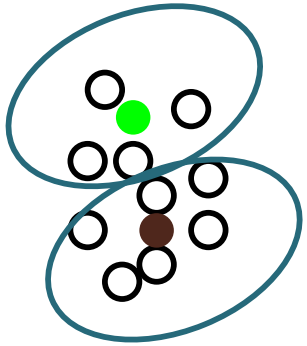


This bad performance, can happen even with well separated Gaussian clusters.

Slide courtesy of Nina Balcan

Lloyd's method: Performance





This bad performance, can happen even with well separated Gaussian clusters.

Some Gaussian are combined…..

Slide courtesy of Nina Balcan

Lloyd's method: Performance

- 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 Lloyd's method will output a bad solution).

  - For k equal-sized Gaussians, Pr[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

Another Initialization Idea: Furthest Point Heuristic

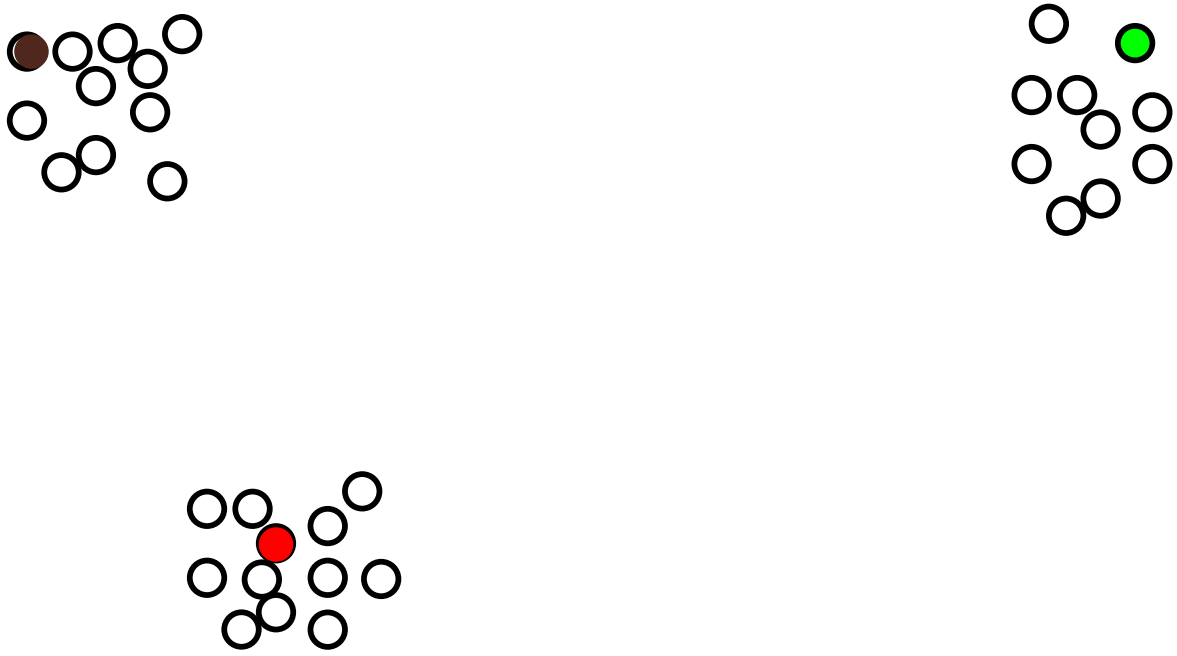
Choose $\mathbf{c_1}$ arbitrarily (or at random).

- For $\mathrm{j} = 2, \ldots, \mathrm{k}$
  - Pick $\mathbf{c_j}$ among datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ that is farthest from previously chosen $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_{j-1}}$

Fixes the Gaussian problem. But it can be thrown off by outliers….

Slide courtesy of Nina Balcan

Furthest point heuristic does well on previous example



Slide courtesy of Nina Balcan

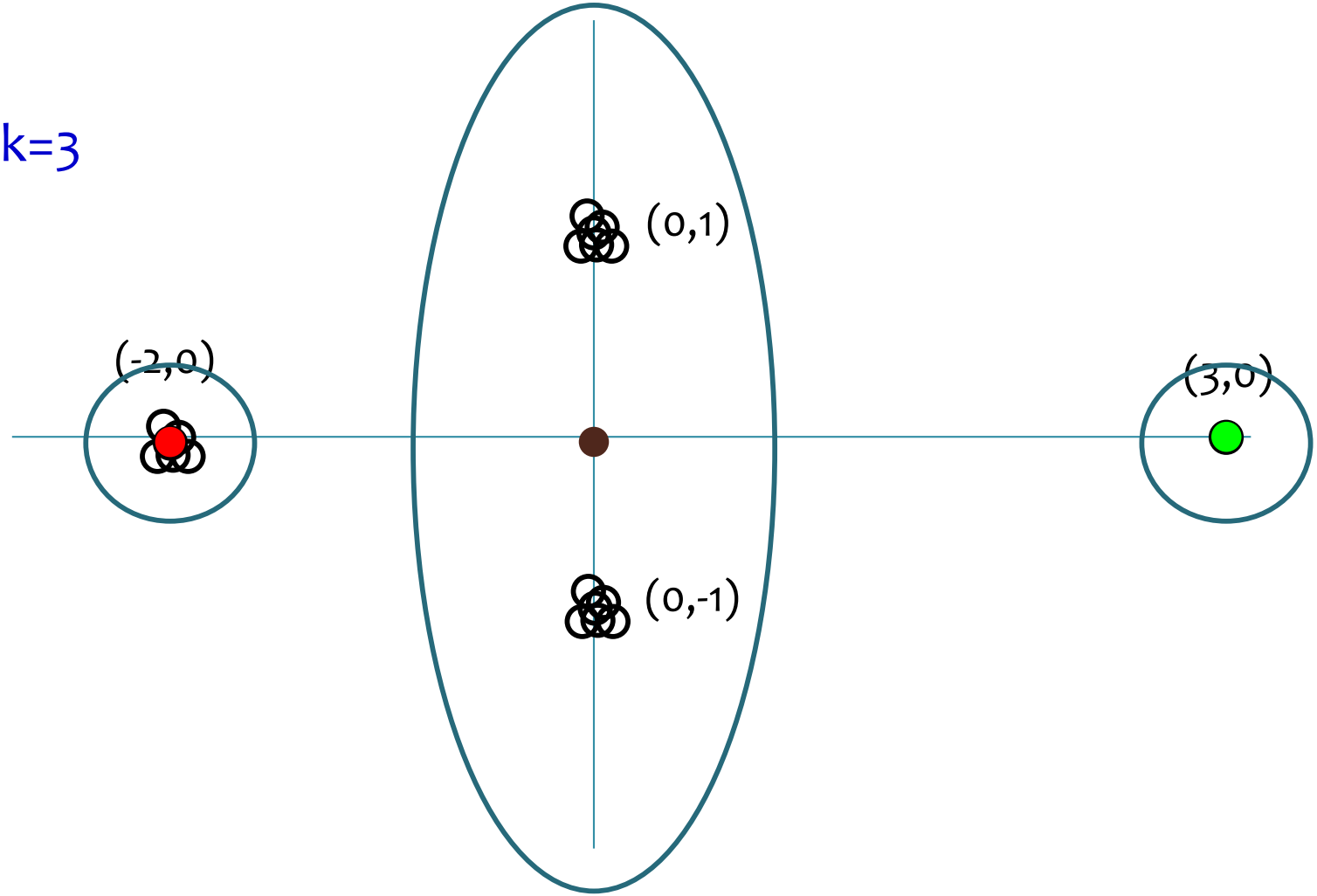
Furthest point initialization heuristic sensitive to outliers

Assume k=3

Slide courtesy of Nina Balcan

Furthest point initialization heuristic sensitive to outliers

Assume k=3



Slide courtesy of Nina Balcan

K-means++ Initialization: D^2 sampling [AV07]

- Interpolate between random and furthest point initialization
- 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

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

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

Running Lloyd's can only further improve the cost.

Slide courtesy of Nina Balcan

K-means++ Idea: D^2 sampling

- Interpolate between random and furthest point initialization
- Let $D(\mathbf{x})$ be the distance between a point x and its nearest center.

  Chose the next center proportional to $D^{\alpha}(\mathbf{x}) = \mathbf{min}_{\mathbf{j'<j}} \left|\left|\mathbf{x^i} - \mathbf{c_{j'}}\right|\right|^{\alpha}$.

  - $\alpha = 0$, random sampling
  - $\alpha = \infty$, furthest point (Side note: it actually works well for k-center)
  - $\alpha = 2$, k-means++

Side note: $\alpha = 1$, works well for k-median

Slide adapted from Nina Balcan

K-means ++ Fix

Slide courtesy of Nina Balcan

K-means++/ Lloyd's Running Time

- K-means ++ initialization: O(nd) and one pass over data to select next center. So O(nkd) time in total.
- Lloyd's method

Repeat until there is no change in the cost.

- For each j: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c_j}\}$
- For each j: $\mathbf{c_j} \leftarrow$ mean of C_j

Each round takes time O(nkd).

- Exponential # of rounds in the worst case [AV07].
- Expected polynomial time in the smoothed analysis (non worst-case) model!

Slide courtesy of Nina Balcan

K-means++/ Lloyd's Summary

- Running Lloyd's can only further improve the cost.
    - Exponential # of rounds in the worst case [AV07].
    - Expected polynomial time in the smoothed analysis model!
    - Does well in practice.
- K-means++ always attains an O(log k) approximation to optimal k-means solution in expectation.

Slide courtesy of Nina Balcan

What value of k?

- Heuristic: Find large gap between (k - 1)-means cost and k-means cost.
- Hold-out validation/cross-validation on auxiliary task (e.g., supervised learning task).
- Try hierarchical clustering.

Slide courtesy of Nina Balcan

EM AND GMMS

Expectation-Maximization Outline

- **Background**
  - Multivariate Gaussian Distribution
  - Marginal Probabilities
- **Building up to GMMs**
  - Distinction #1: Model vs. Objective Function
  - Gaussian Naïve Bayes (GNB)
  - Gaussian Discriminant Analysis
  - Gaussian Mixture Model (GMM)
- **Expectation-Maximization**
  - Distinction #2: Data Likelihood vs. Marginal Data Likelihood
  - Distinction #3: Latent Variables vs. Parameters
  - Objective Functions for EM
  - Hard (Viterbi) EM Algorithm
    - Example: K-Means as Hard EM
  - Soft (Standard) EM Algorithm
    - Example: Soft EM for GMM
- **Properties of EM**
  - Nonconvexity / Local Optimization
  - Example: Grammar Induction
  - Variants of EM

Background

Whiteboard

- Multivariate Gaussian Distribution
- Marginal Probabilities

GAUSSIAN MIXTURE MODEL (GMM)

Building up to GMMs

Whiteboard

- Distinction #1: Model vs. Objective Function
- Gaussian Naïve Bayes (GNB)
- Gaussian Discriminant Analysis
- Gaussian Mixture Model (GMM)

Gaussian Discriminant Analysis

Data: $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$ and $z^{(i)} \in \{1, \ldots, K\}$

Generative Story:

$$z \sim \text{Categorical}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

Model:

Joint: $p(\mathbf{x}, z; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

Log-likelihood:

$$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z^{(i)}; \boldsymbol{\phi})$$

Gaussian Discriminant Analysis

Data: $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$ and $z^{(i)} \in \{1, \ldots, K\}$

Log-likelihood: $$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} | z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z^{(i)}; \boldsymbol{\phi})$$

Maximum Likelihood Estimates:

Take the derivative of the Lagrangian, set it equal to zero and solve.

$$\phi_k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k), \forall k$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k) \mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)}, \forall k$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)}, \forall k$$

Implementation: Just counting

Gaussian Mixture-Model

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

Generative Story:

$$z \sim \text{Categorical}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

Model:

Joint: $p(\mathbf{x}, z; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

Marginal: $p(\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{z=1}^{K} p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

(Marginal) Log-likelihood:

$$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p(\mathbf{x}^{(i)}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$$

Mixture-Model

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

Generative Story:

$$z \sim \text{Categorical}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\cdot|z)$$

Model:

Joint: $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, z) = p_{\boldsymbol{\theta}}(\mathbf{x}|z)p_{\boldsymbol{\phi}}(z)$$

Marginal: $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}|z)p_{\boldsymbol{\phi}}(z)$$

(Marginal) Log-likelihood:

$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z)p_{\boldsymbol{\phi}}(z)$$

Mixture-Model

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

Generative Story:

$$z \sim \text{Categorical}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\cdot|z)$$

This could be any arbitrary distribution parameterized by **θ**.

Today we're thinking about the case where it is a Multivariate Gaussian.

Model:

Joint: $p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, z) = p_{\boldsymbol{\theta}}(\mathbf{x}|z)p_{\boldsymbol{\phi}}(z)$

Marginal: $p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}|z)p_{\boldsymbol{\phi}}(z)$

(Marginal) Log-likelihood:

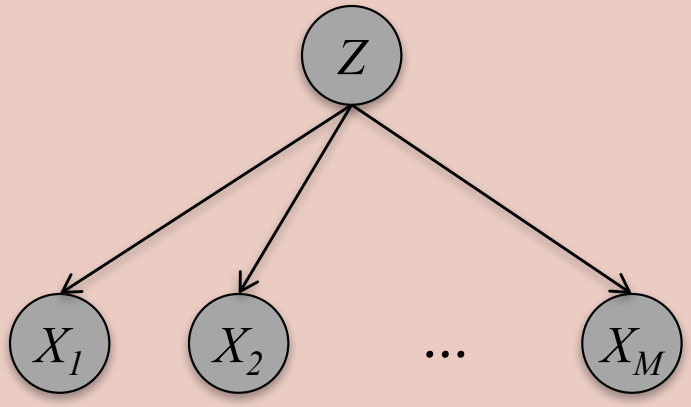
$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z)p_{\boldsymbol{\phi}}(z)$$

Learning a Mixture Model

Supervised Learning: The parameters decouple!

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



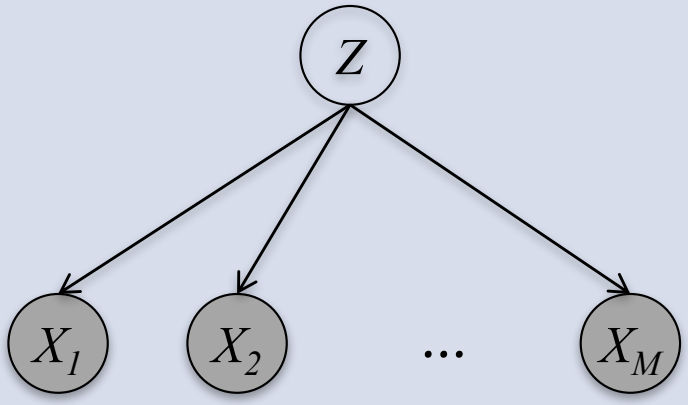
$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \operatorname*{argmax}_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z^{(i)}) p_{\boldsymbol{\phi}}(z^{(i)})$$

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z^{(i)})$$

$$\boldsymbol{\phi}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_{\boldsymbol{\phi}}(z^{(i)})$$

Unsupervised Learning: Parameters are coupled by marginalization.

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



$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \operatorname*{argmax}_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z) p_{\boldsymbol{\phi}}(z)$$

Learning a Mixture Model

Supervised Learning: The parameters decouple!

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

$\boldsymbol{\theta}^*, \boldsymbol{\phi}^*$

$\boldsymbol{\theta}$

$\boldsymbol{\phi}$

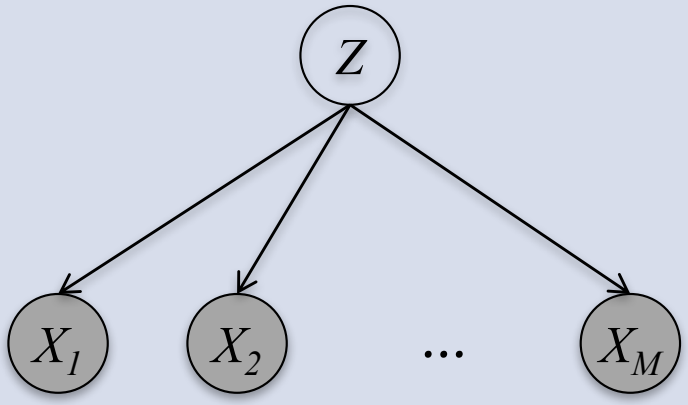
Training certainly isn't as simple as the supervised case.

In many cases, we could still use some black-box optimization method (e.g. Newton-Raphson) to solve this *coupled* optimization problem.

This lecture is about a more problem-specific method: EM.

Unsupervised Learning: Parameters are coupled by marginalization.

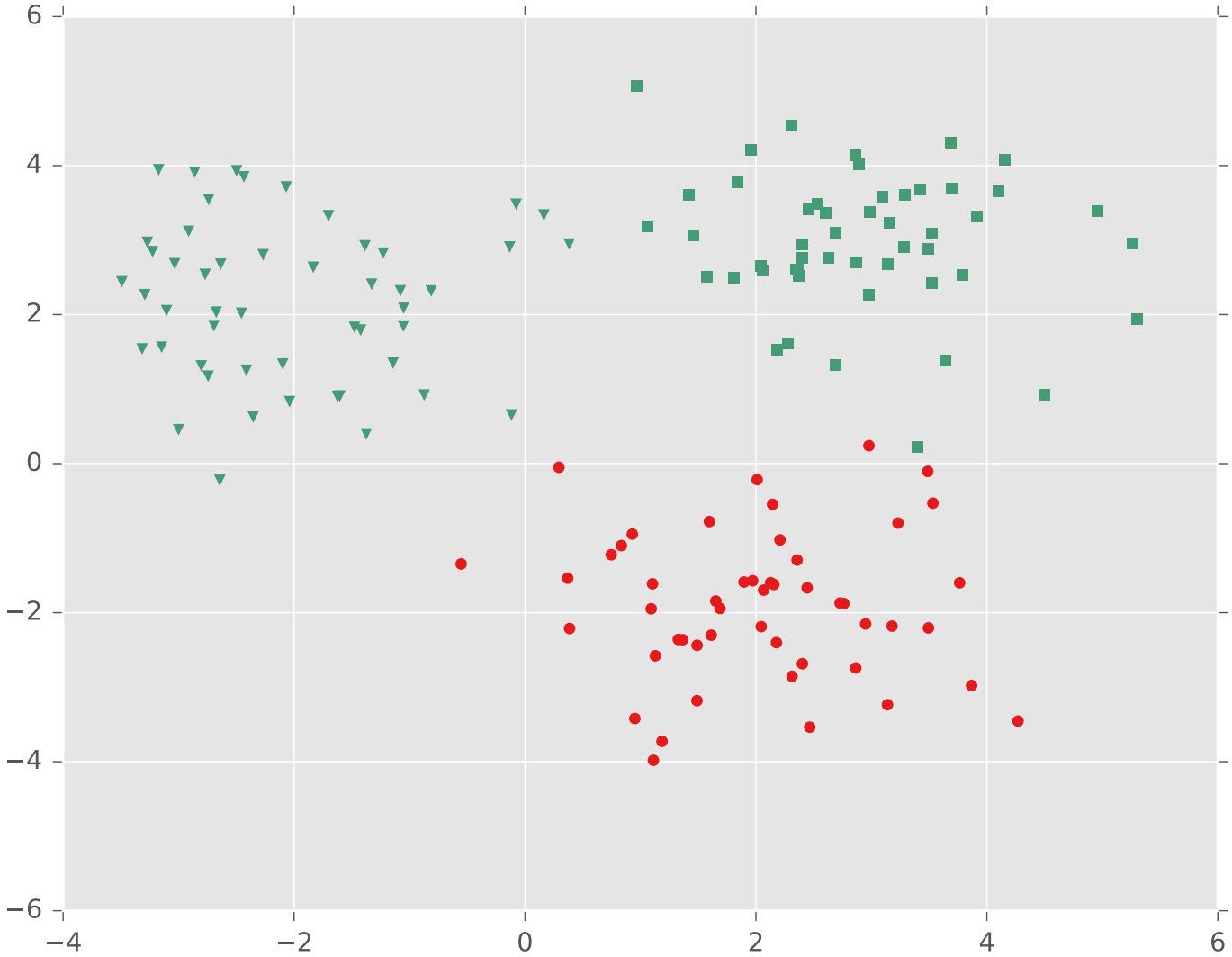
$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$



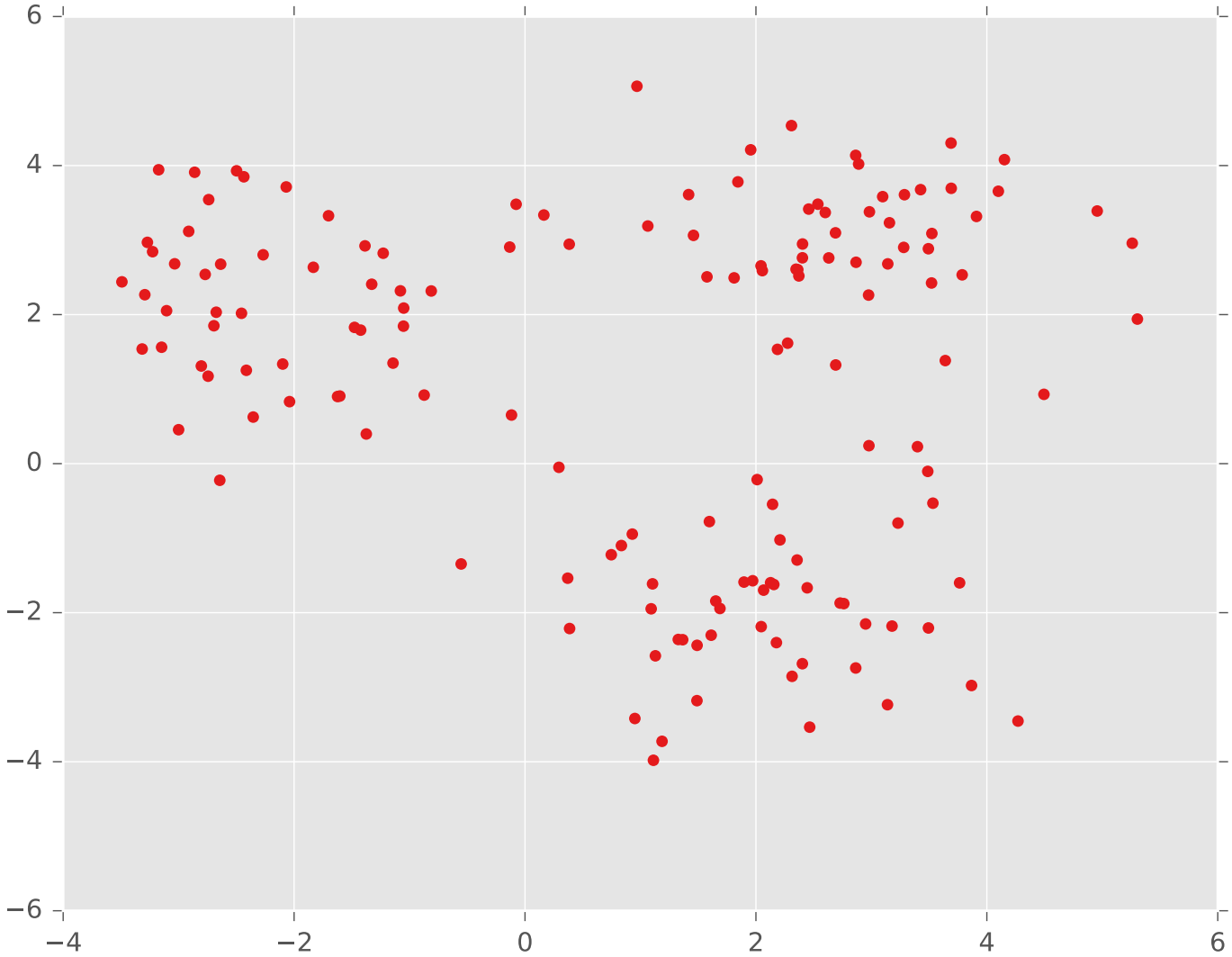
$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta}, \boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z) p_{\boldsymbol{\phi}}(z)$$

EXAMPLE: K-MEANS VS GMM

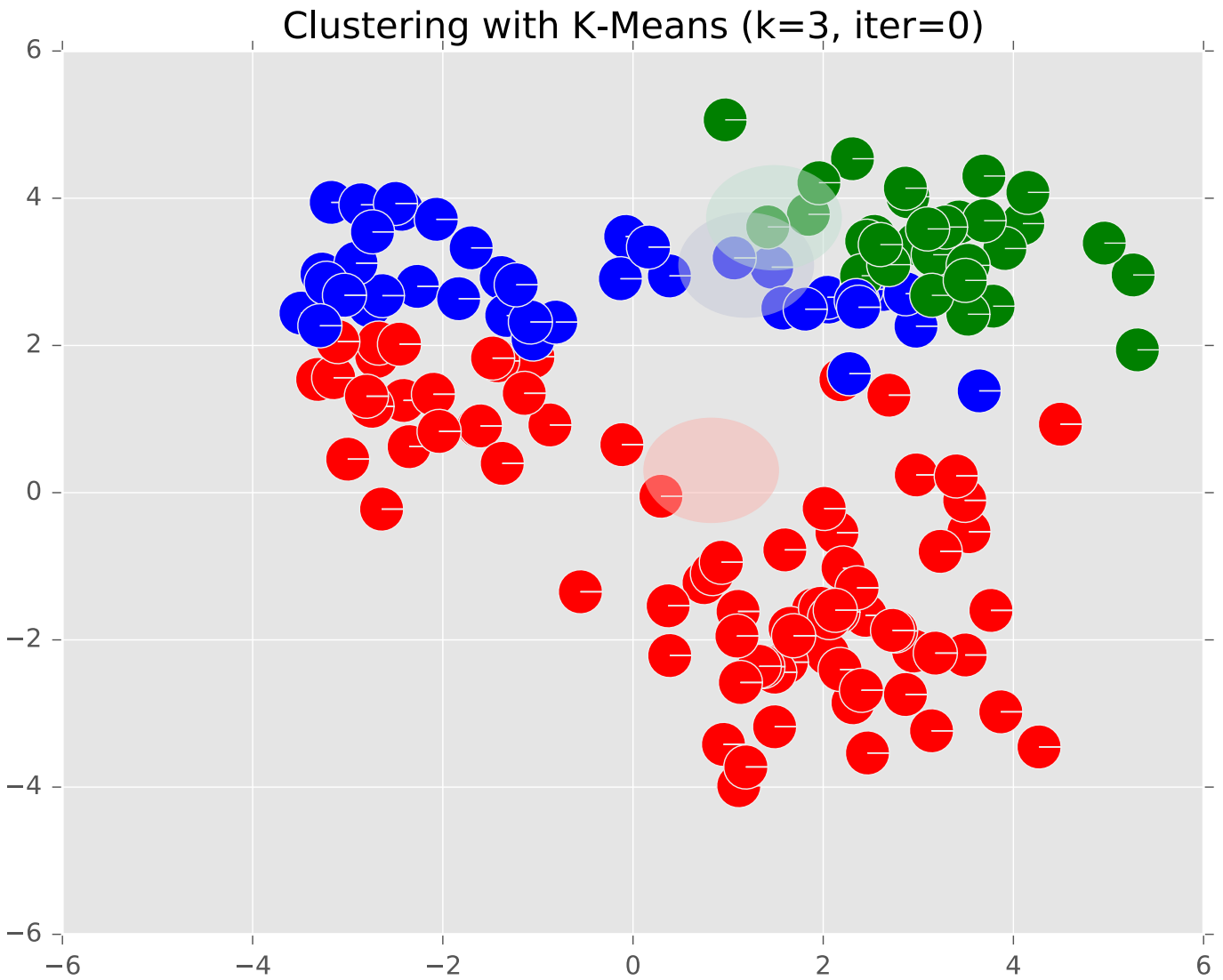
Example: K-Means



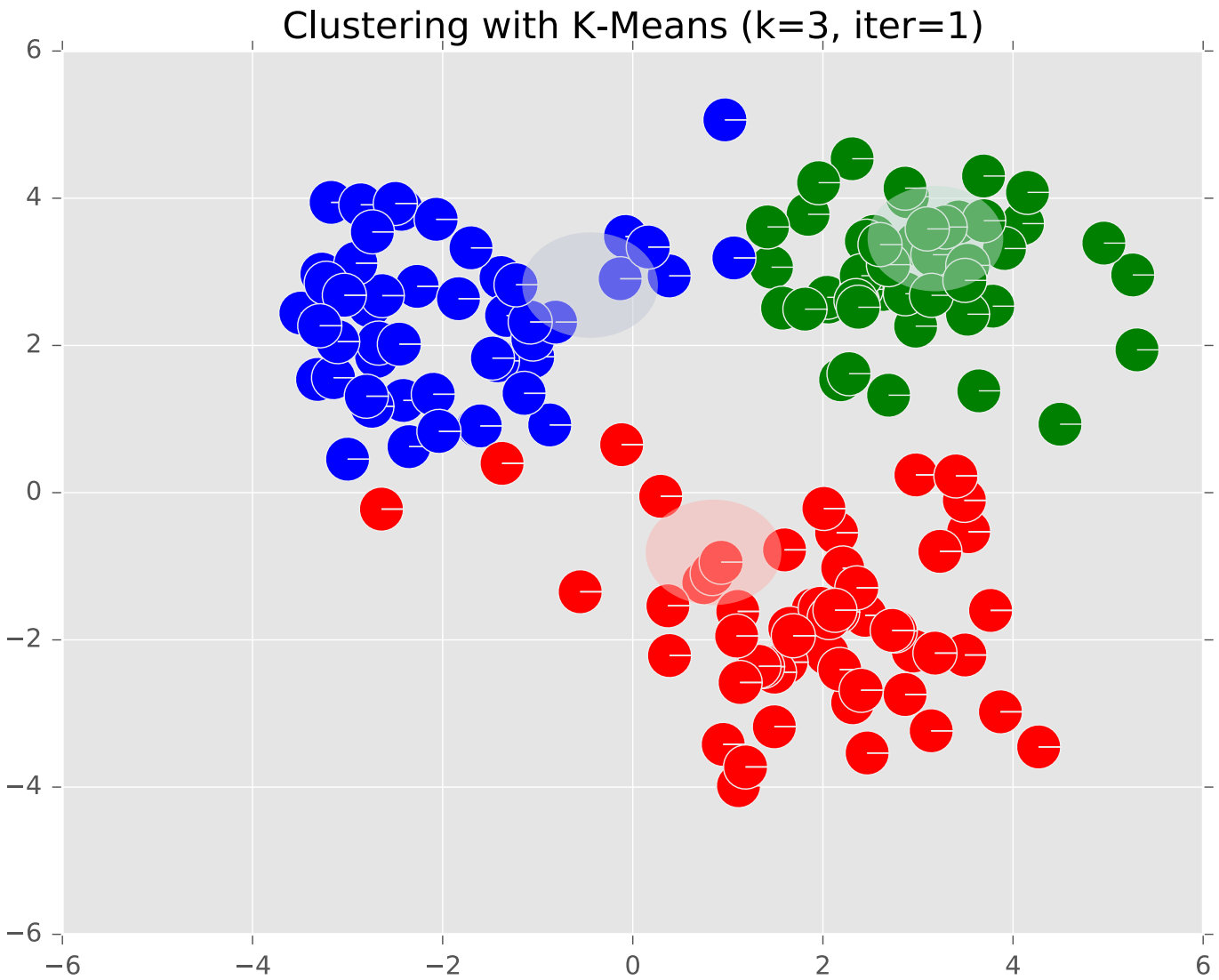
Example: K-Means



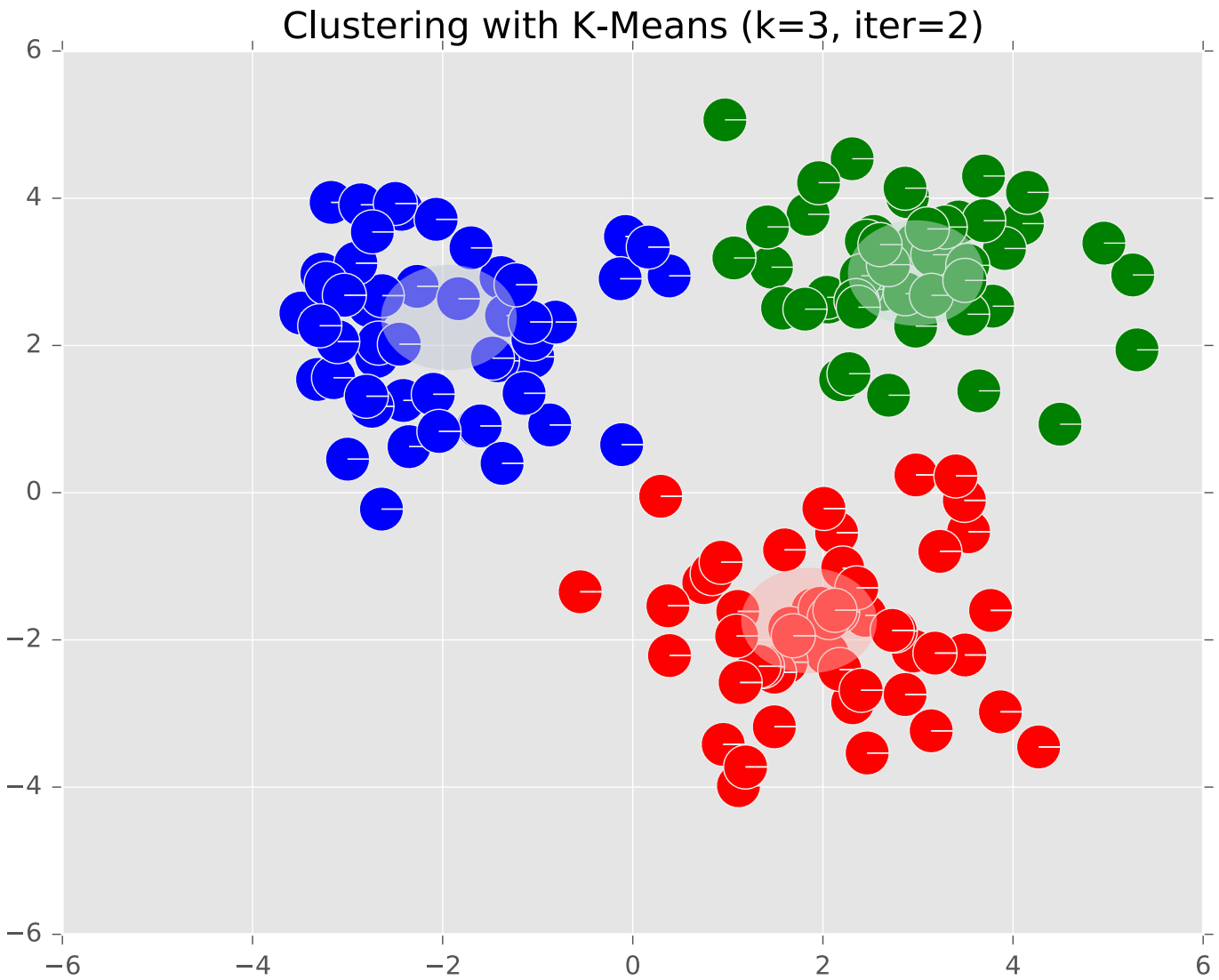
Example: K-Means



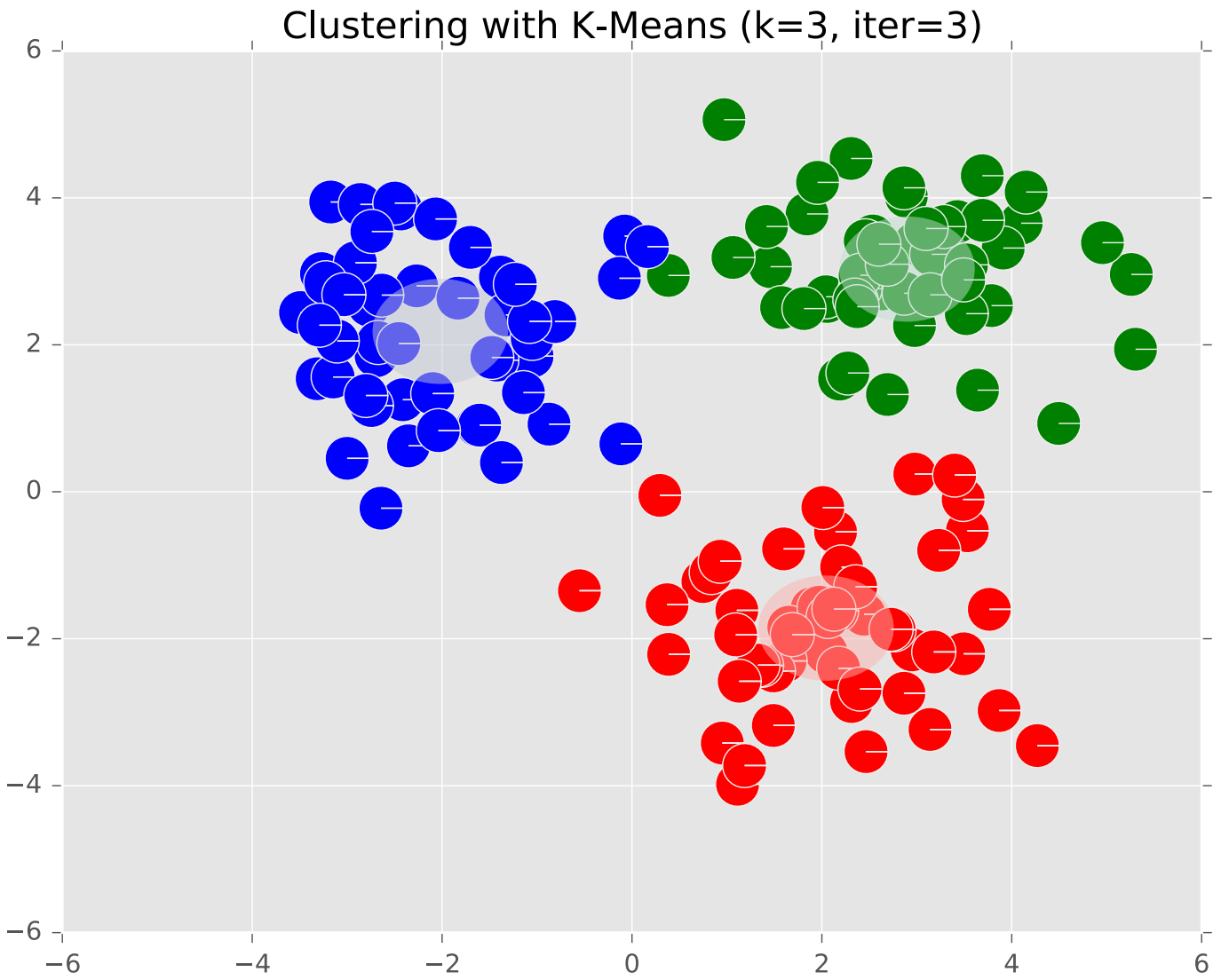
Example: K-Means



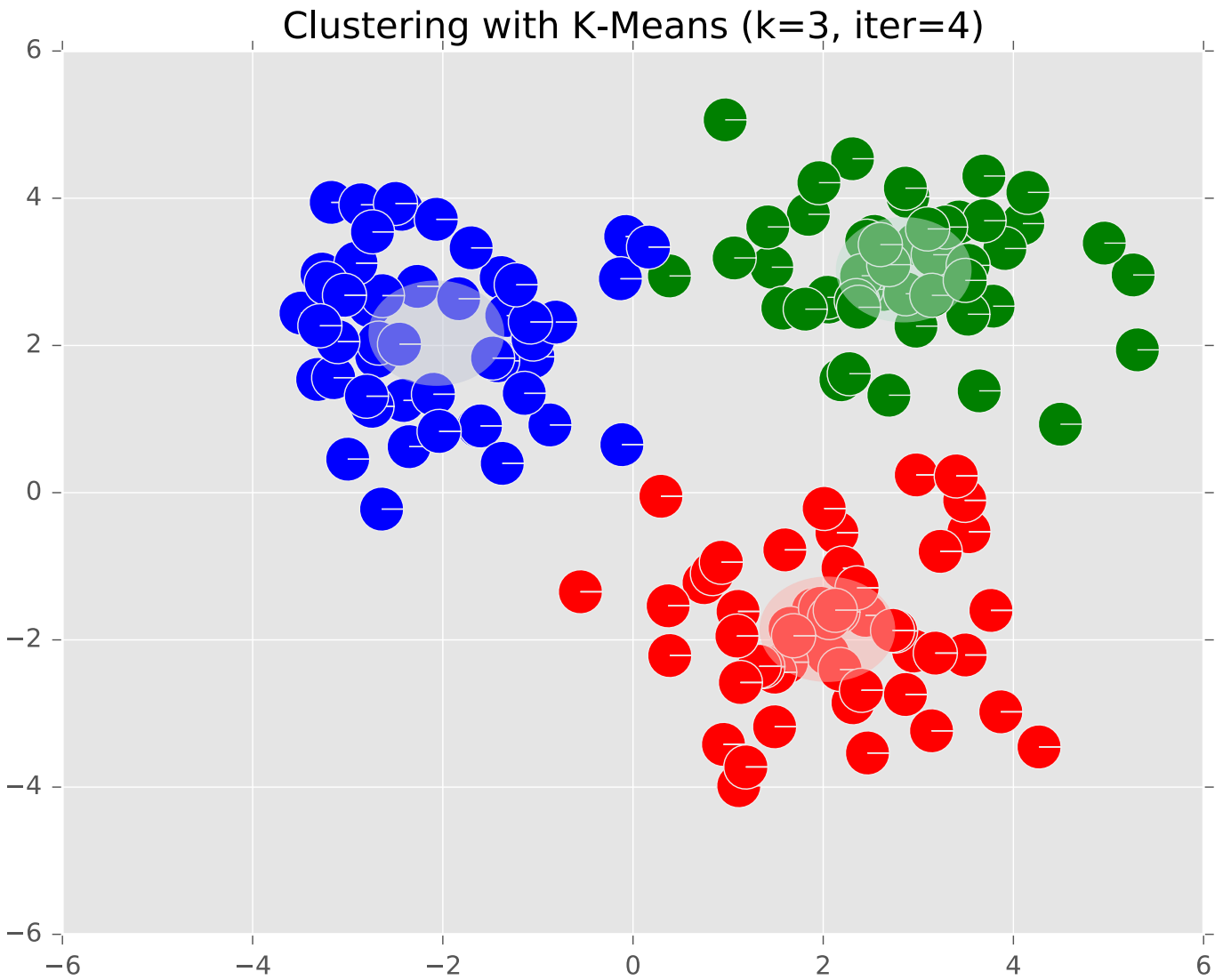
Example: K-Means



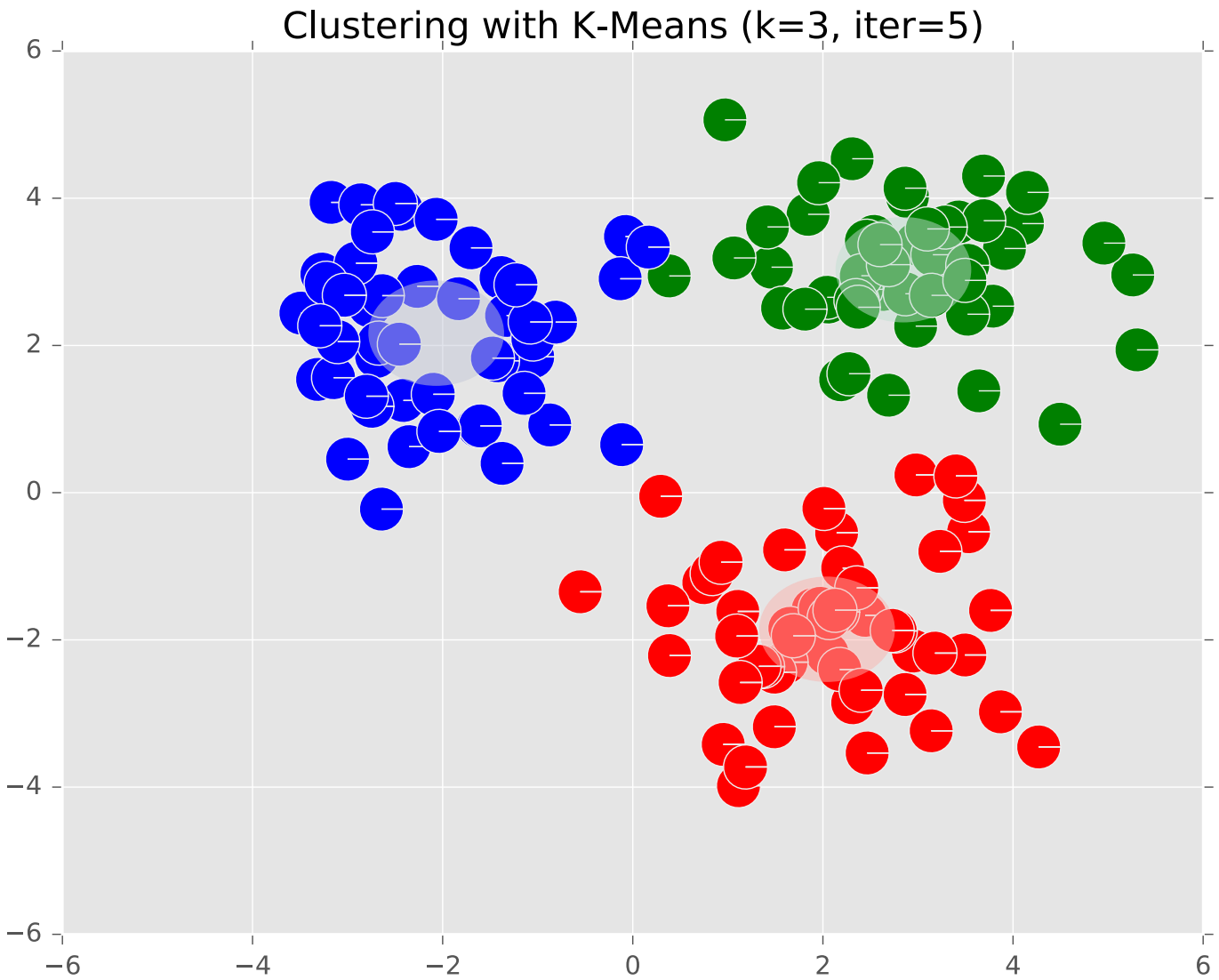
Example: K-Means



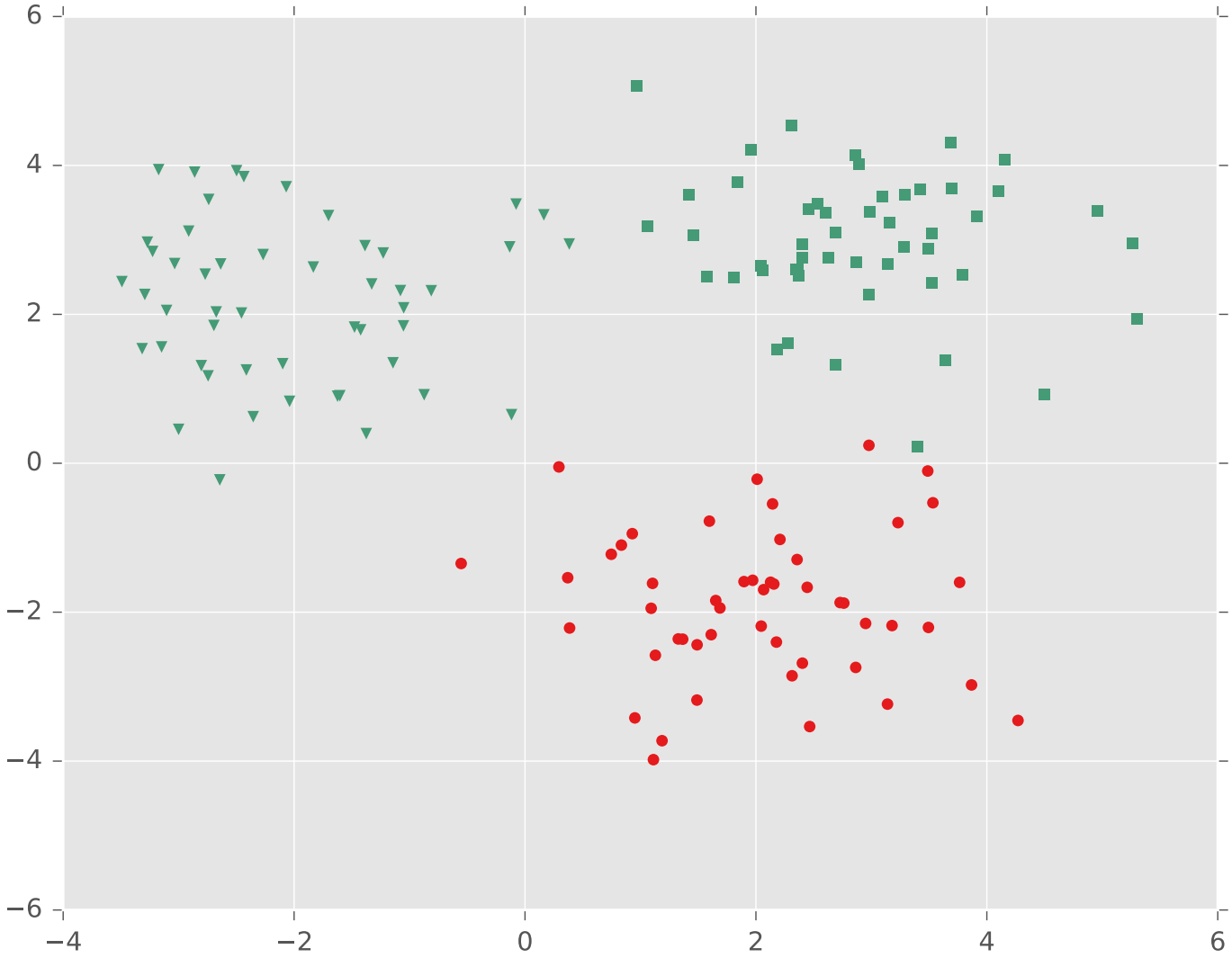
Example: K-Means



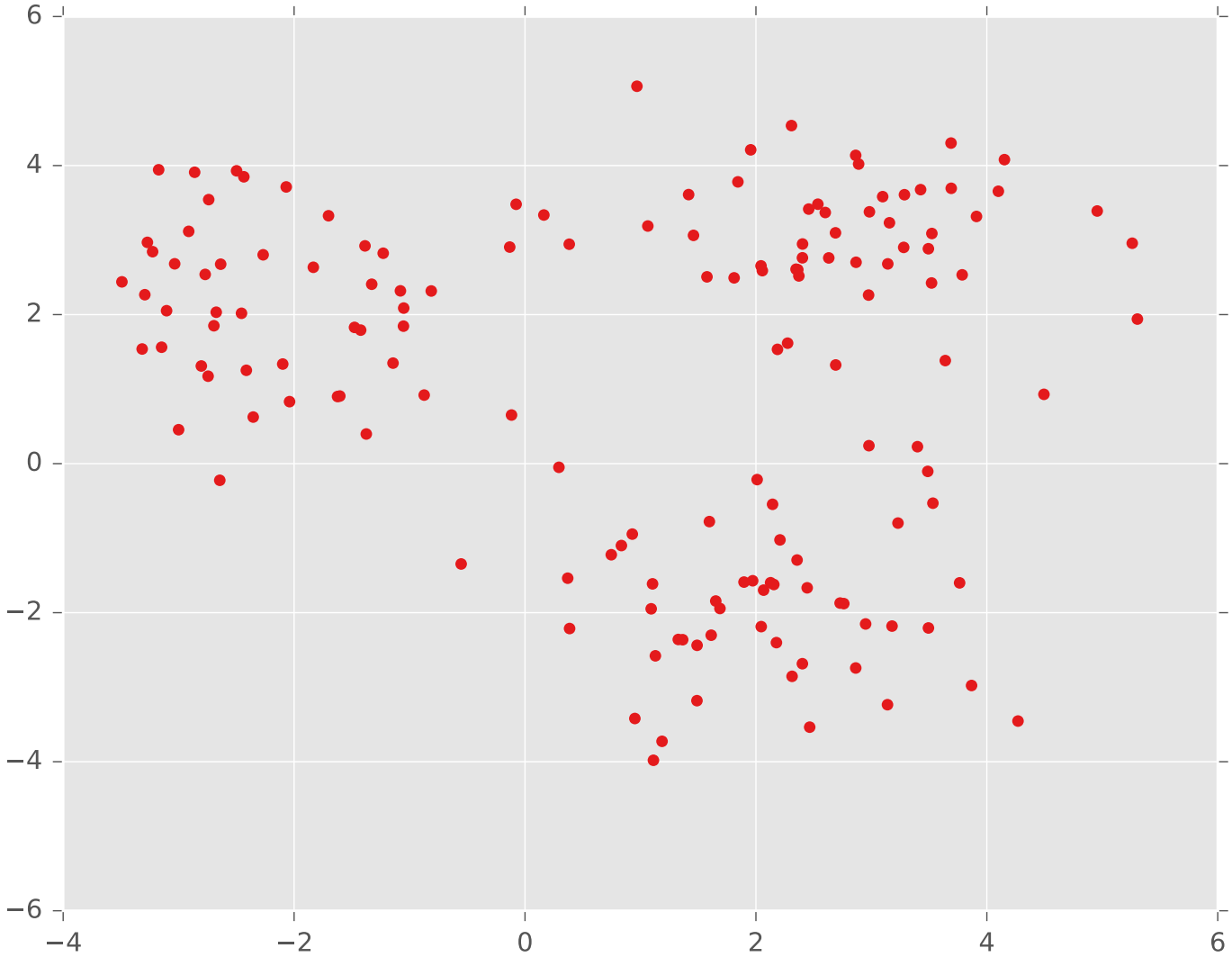
Example: K-Means



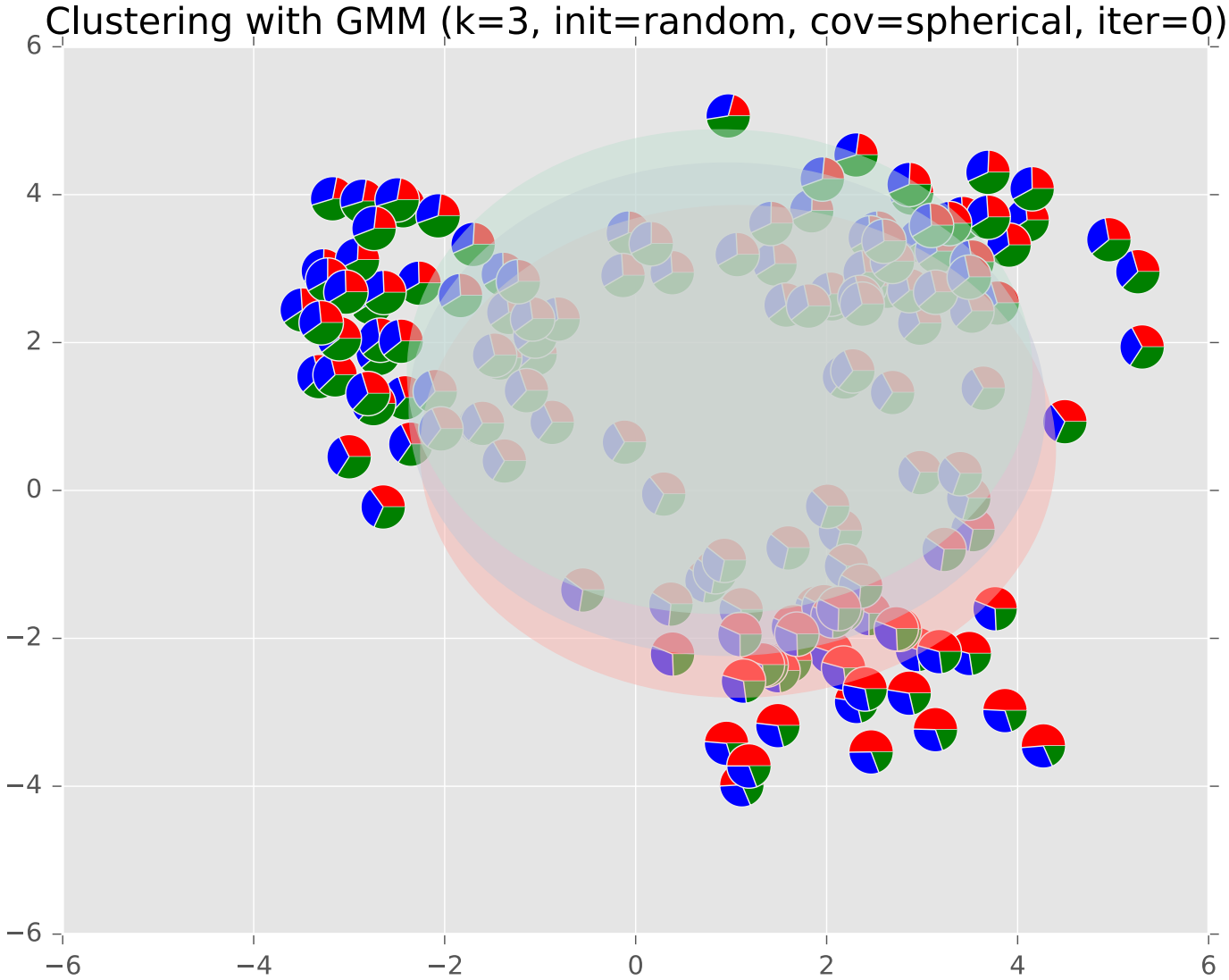
Example: GMM



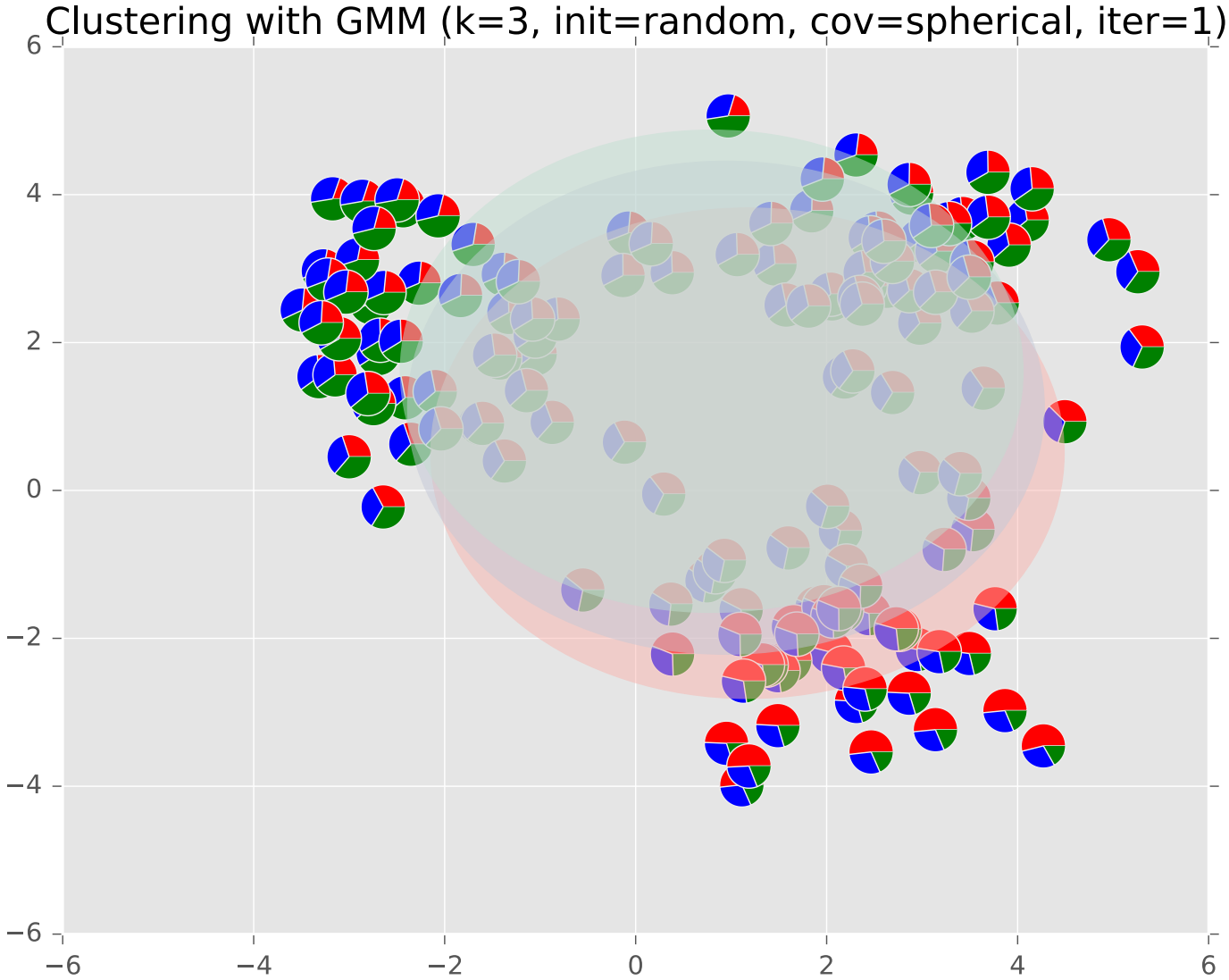
Example: GMM



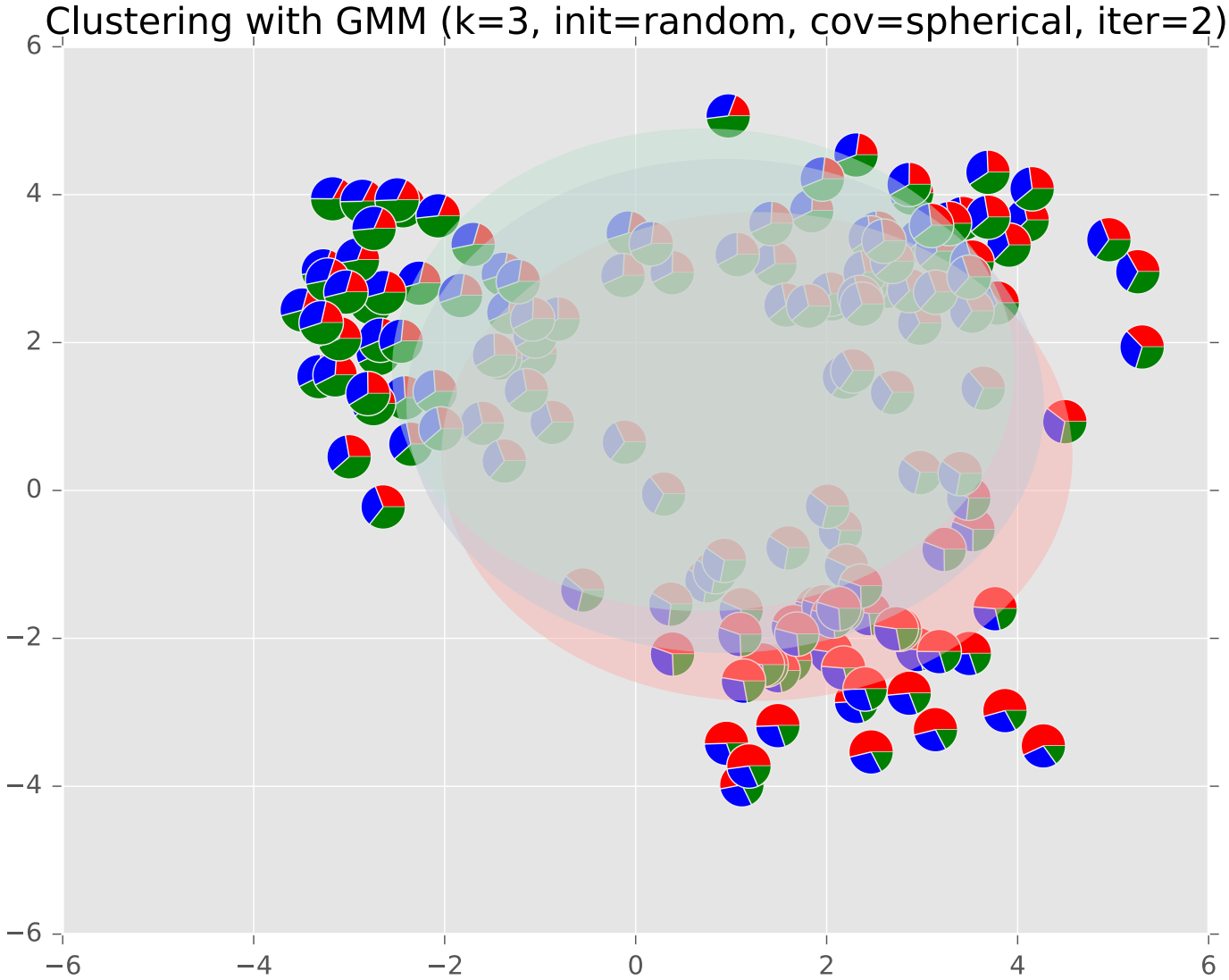
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=0)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

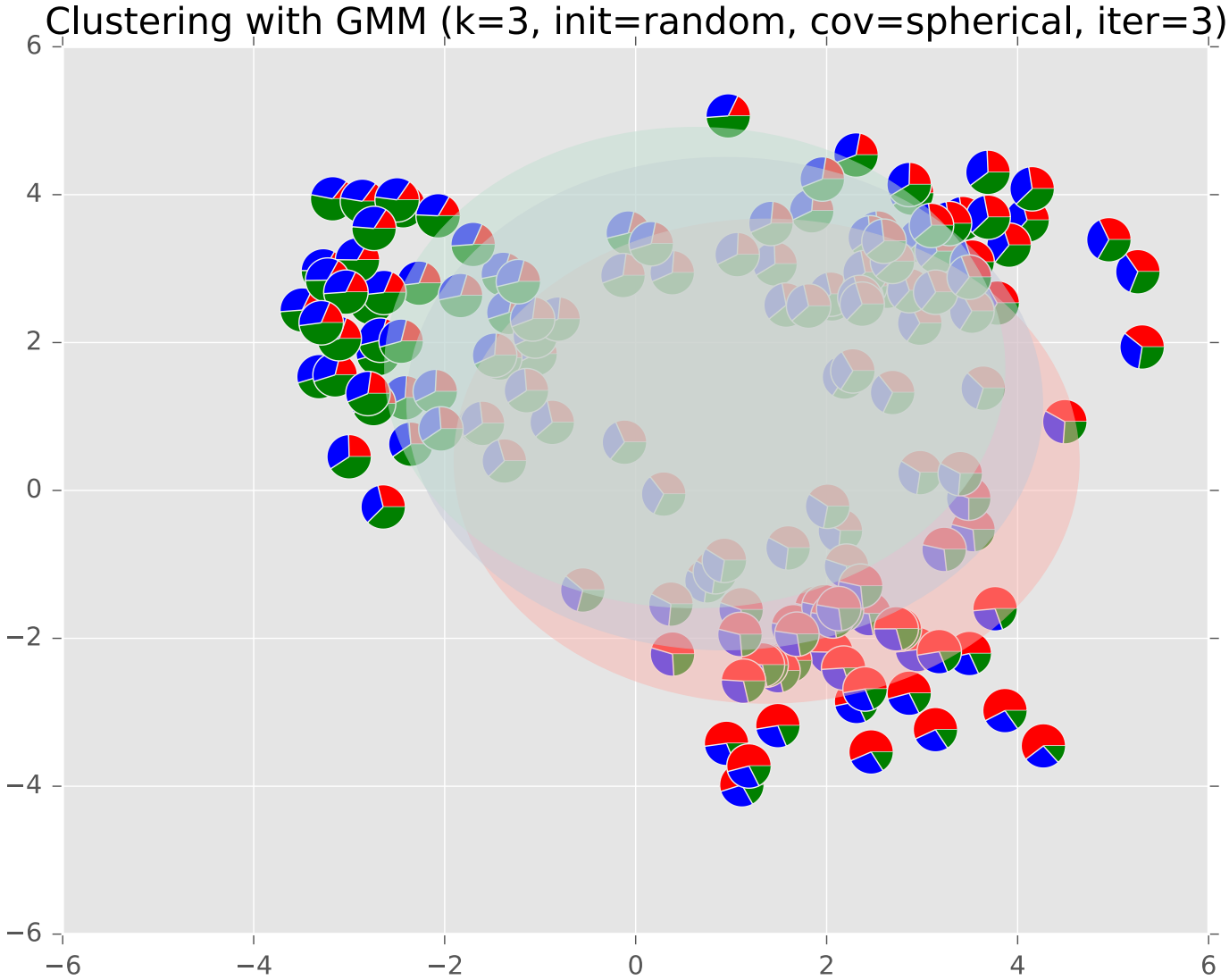
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=1)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

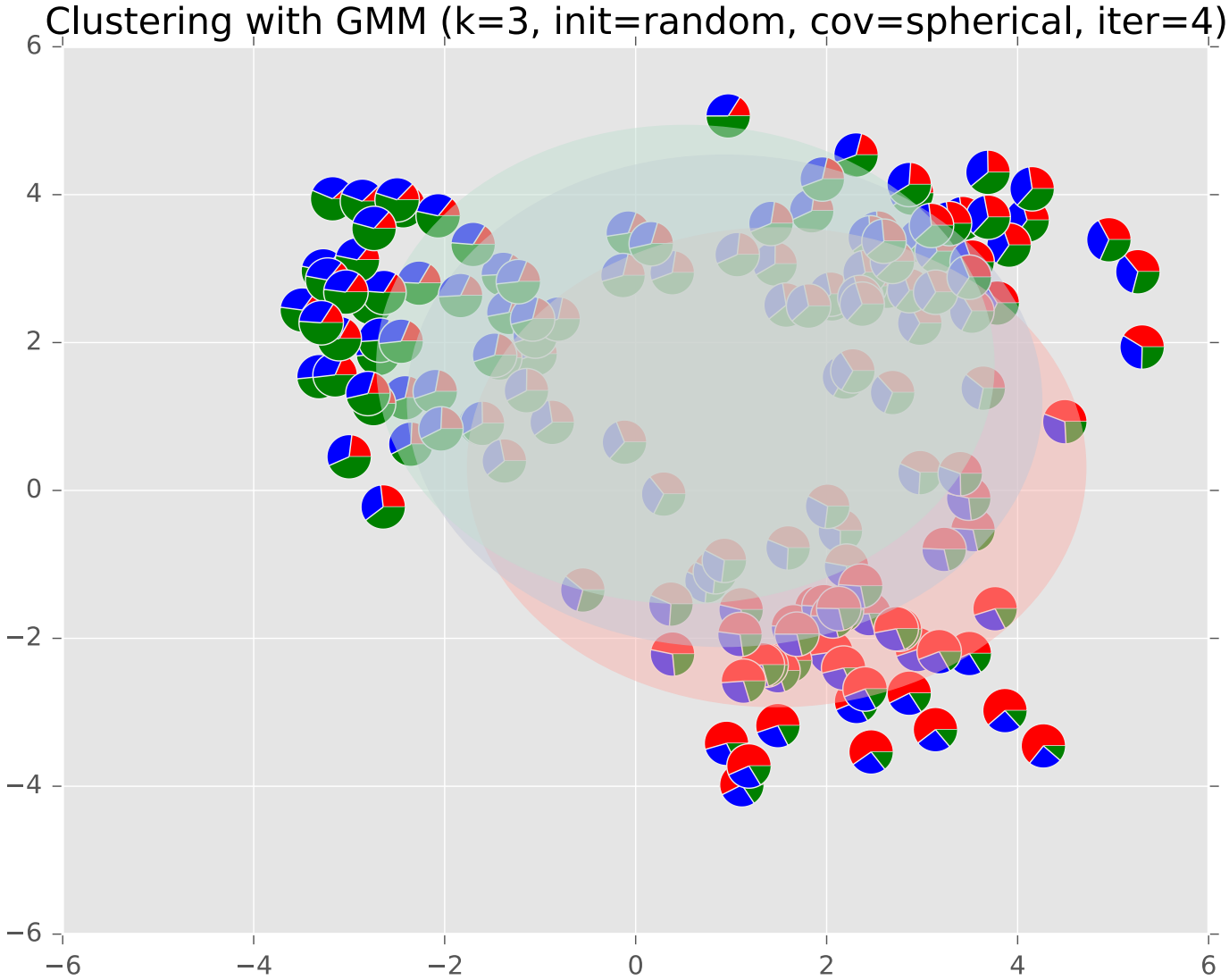
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=2)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

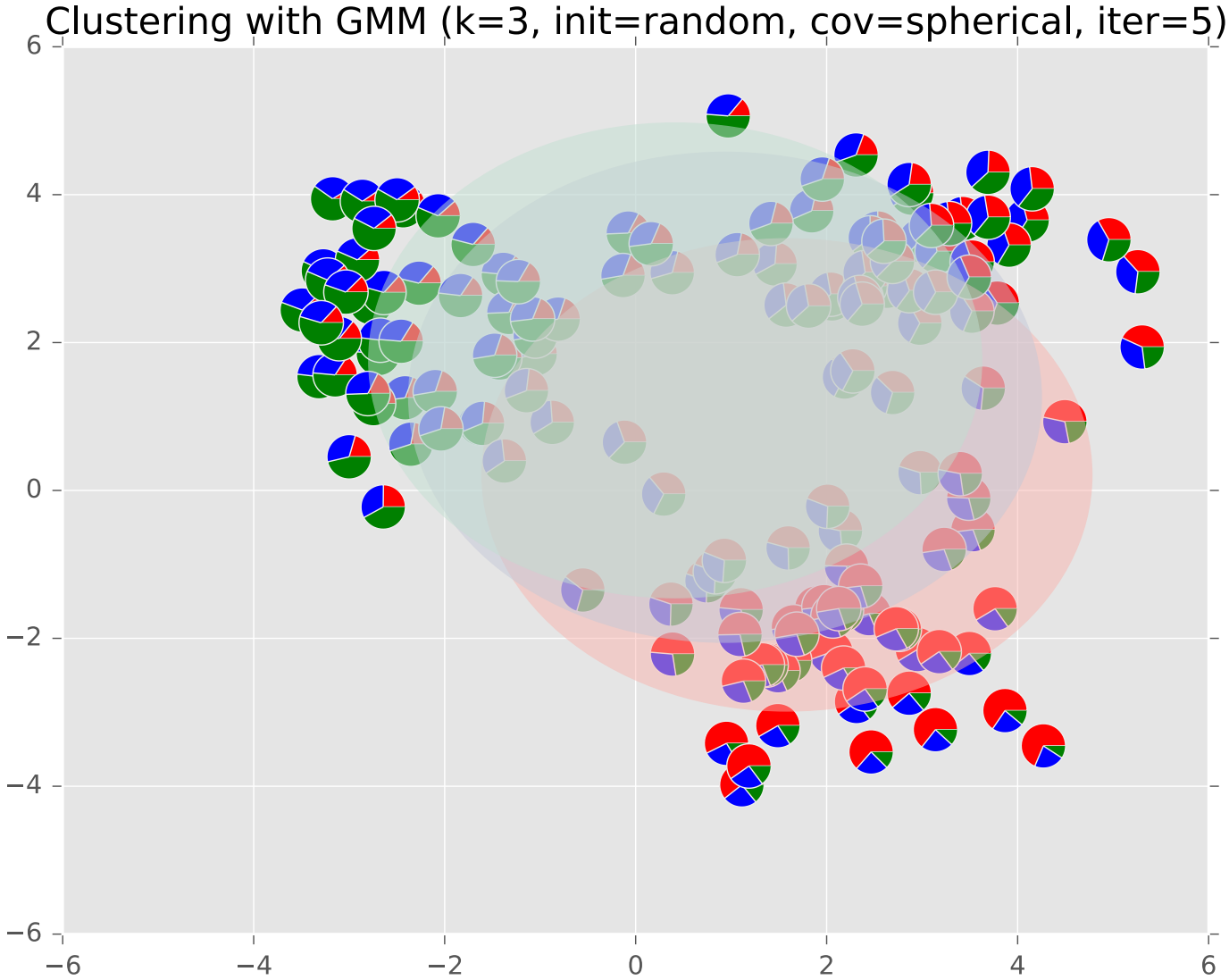
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=3)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

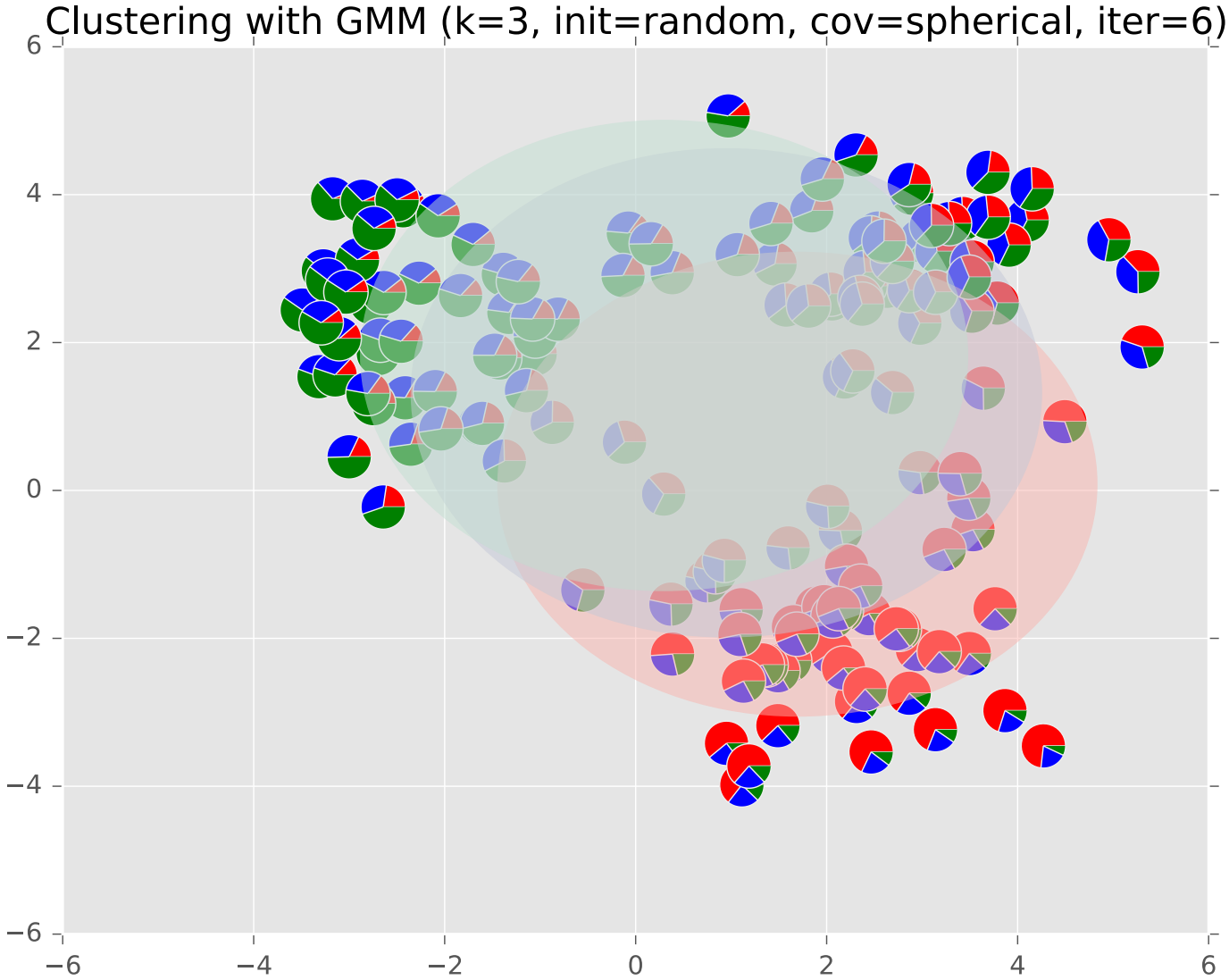
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=4)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

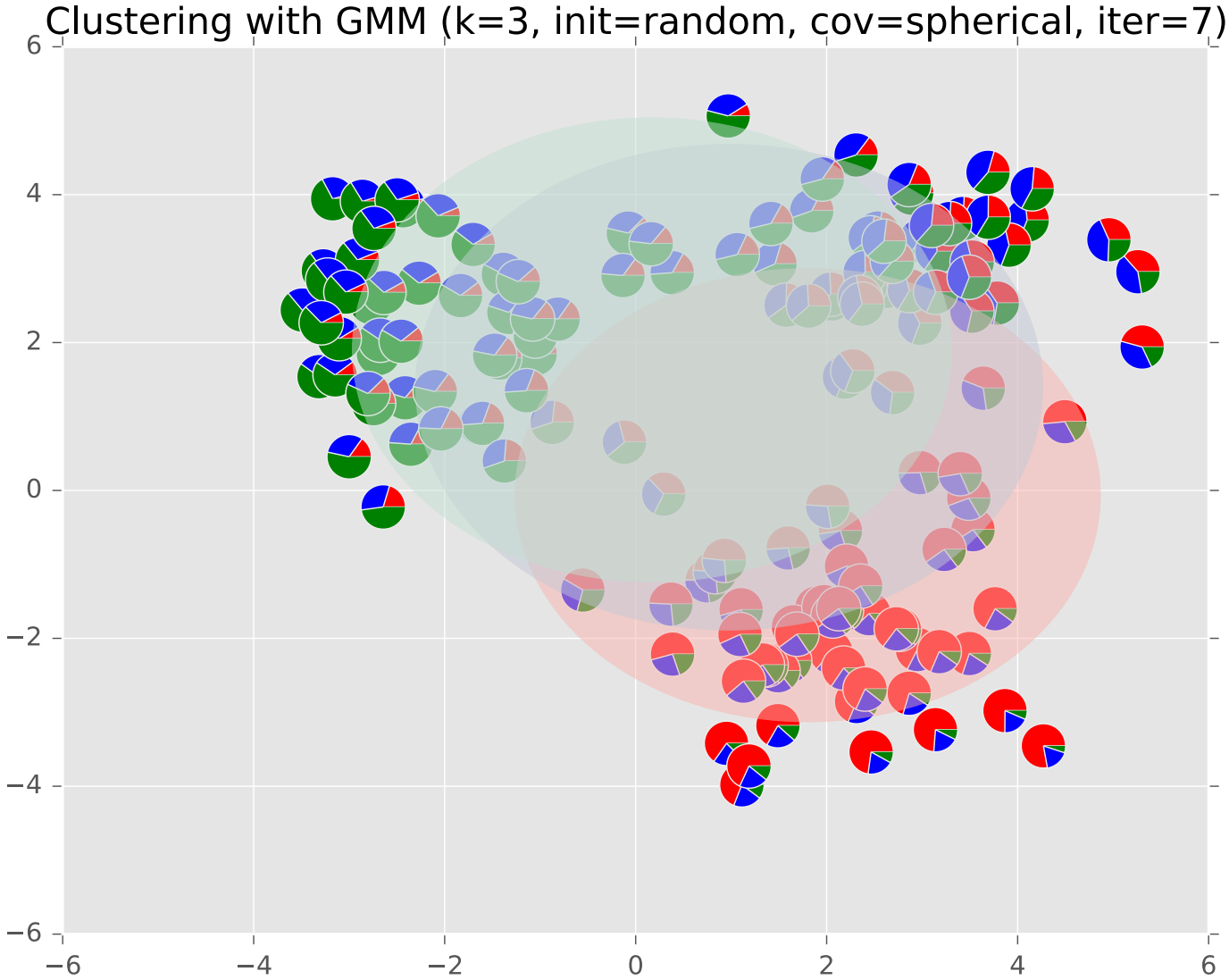
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=5)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

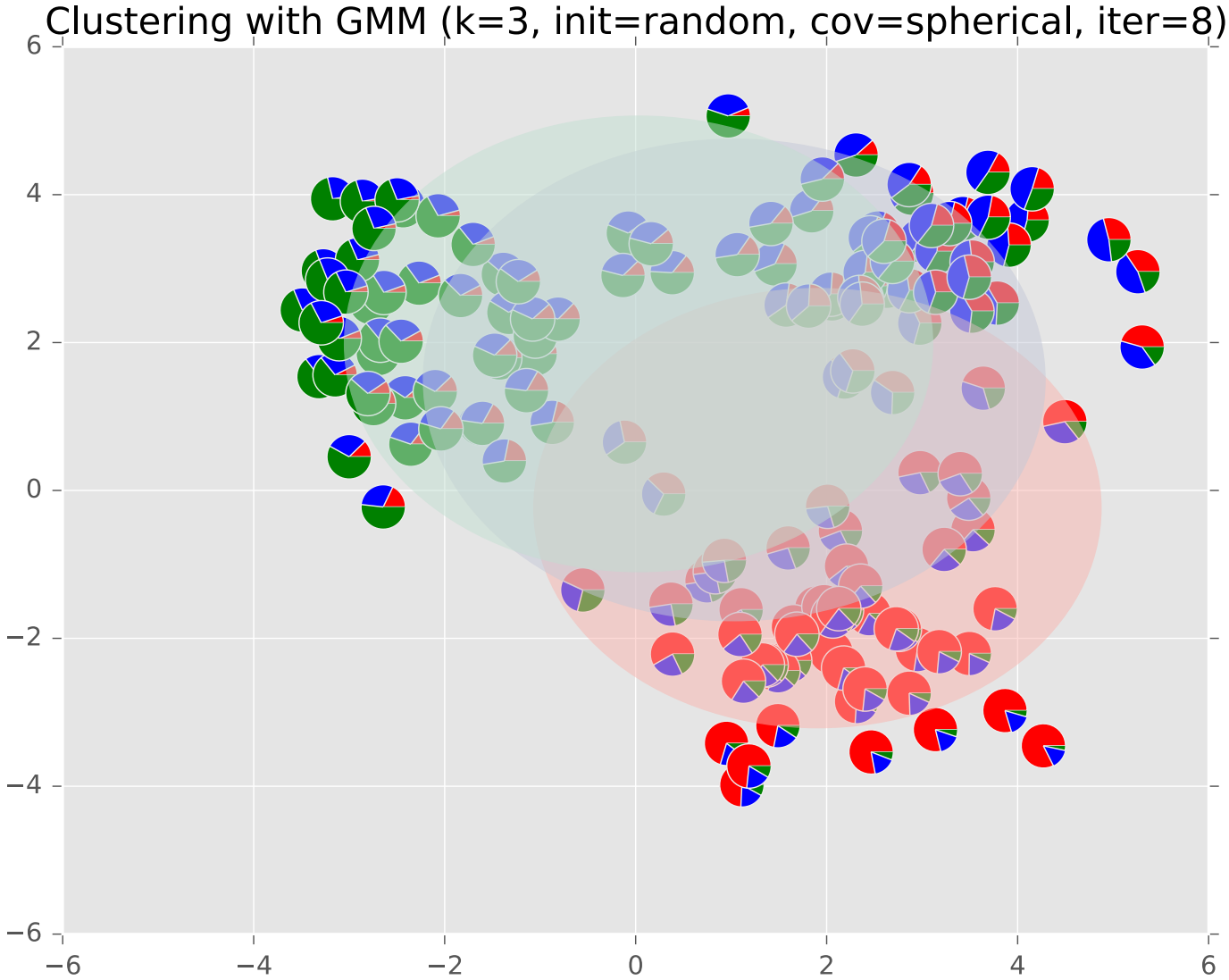
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=6)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

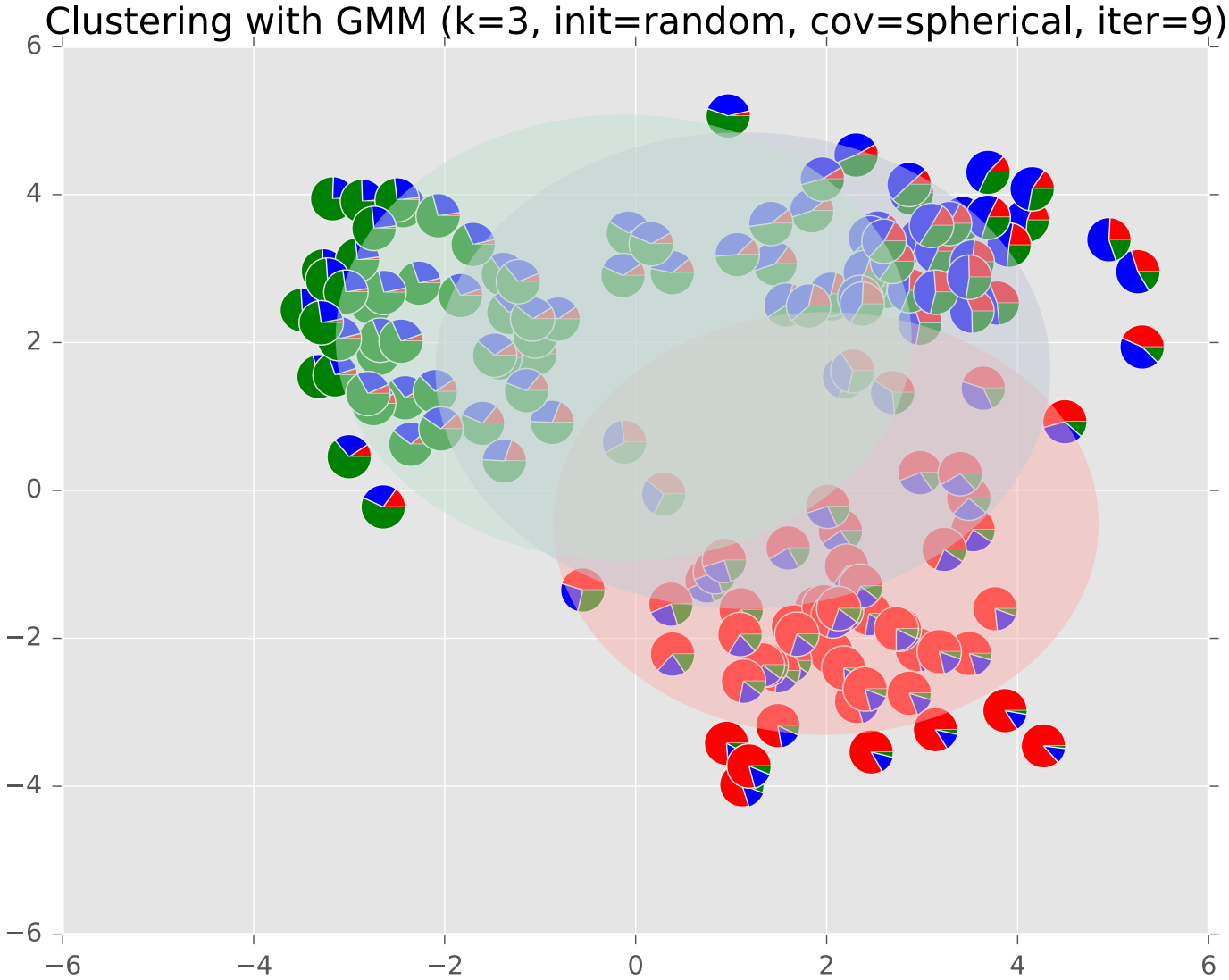
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=7)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

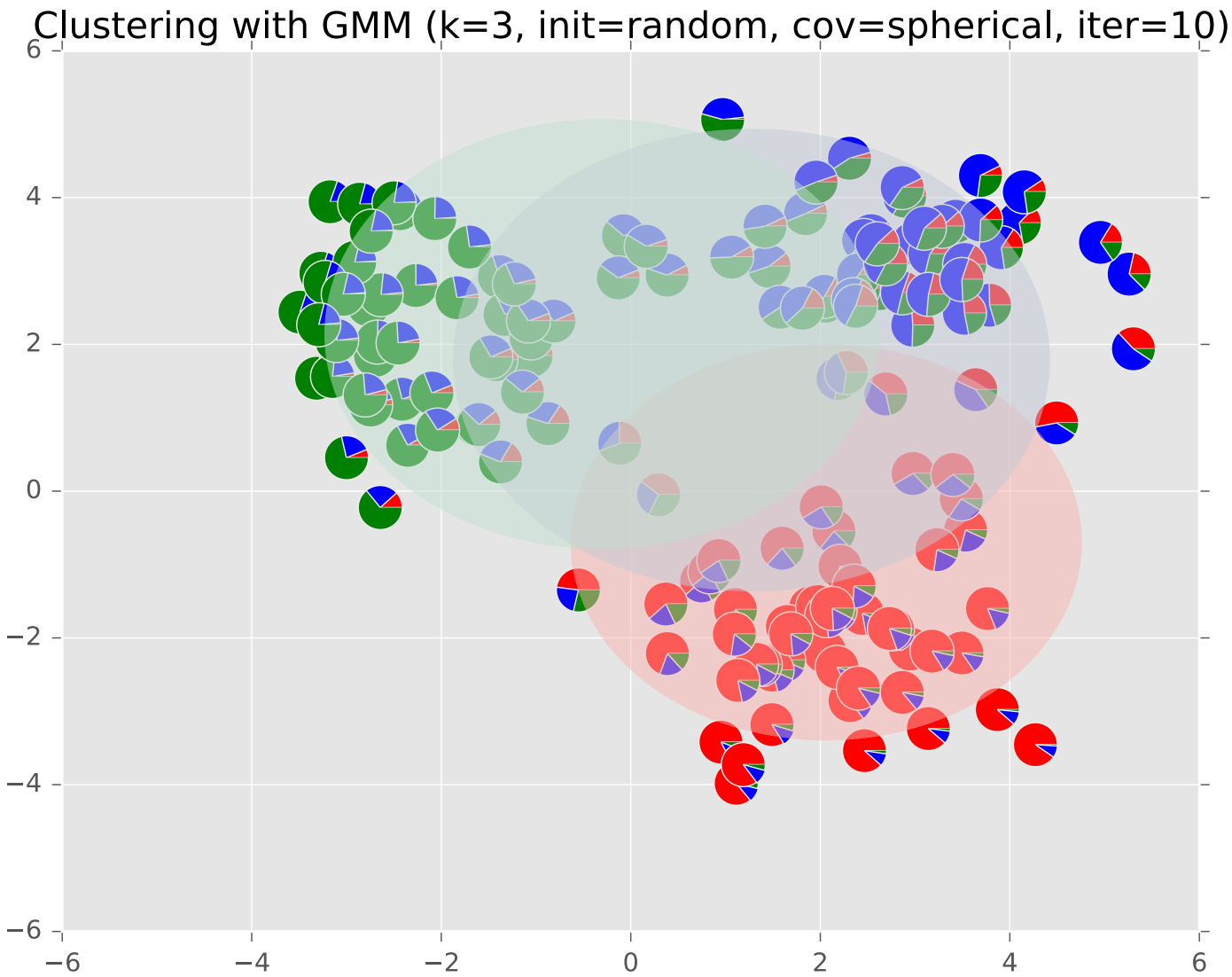
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=8)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

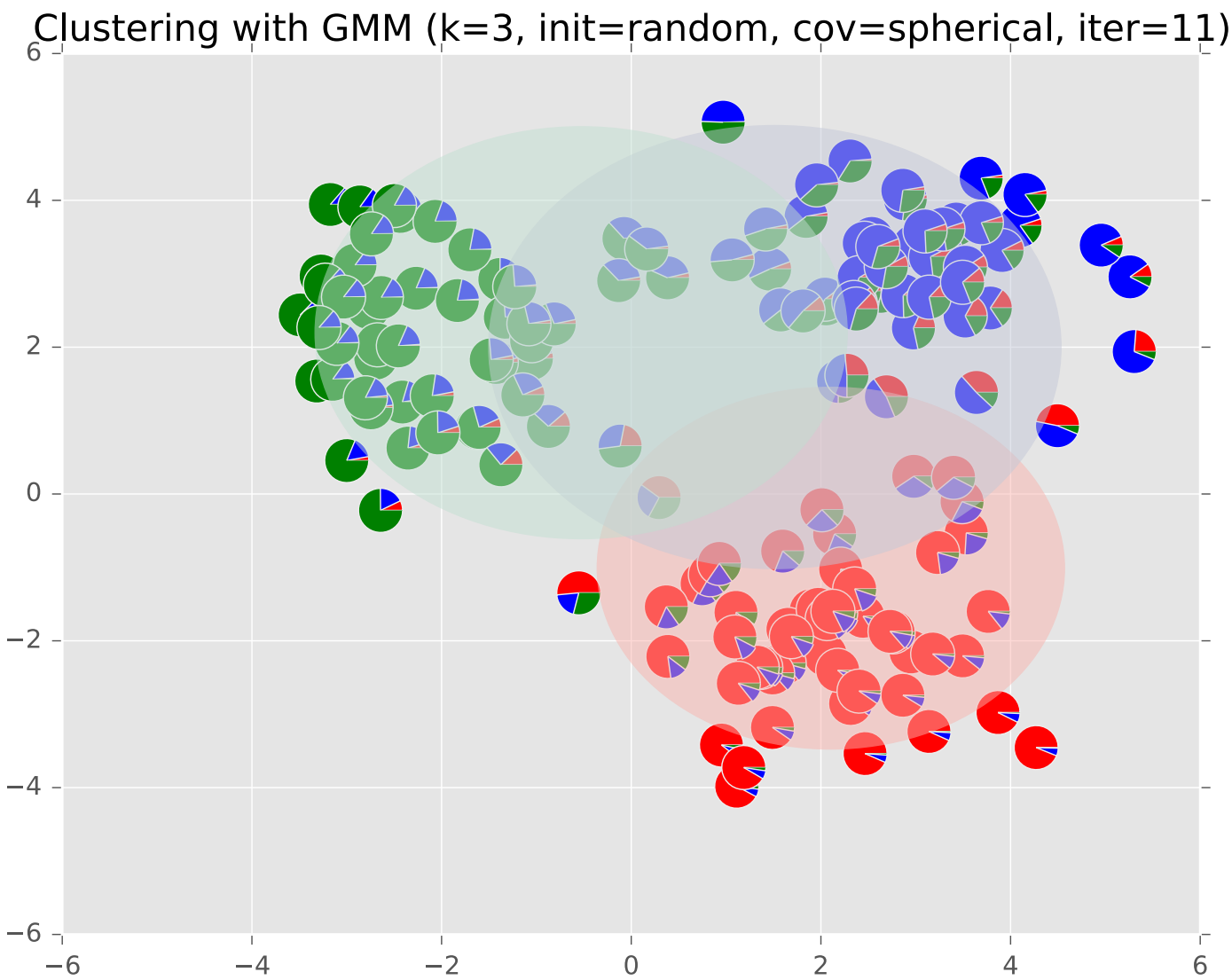
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=9)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

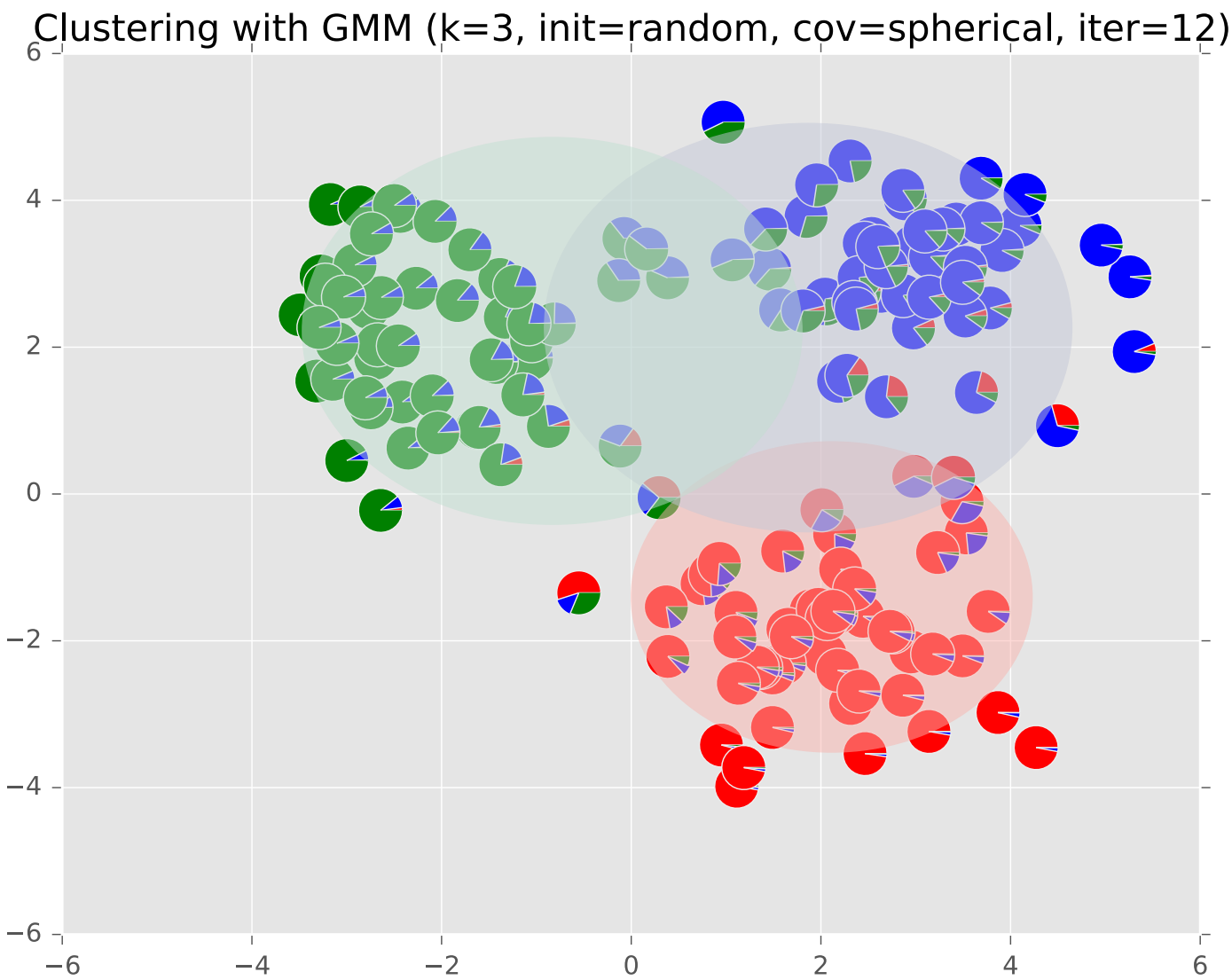
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=10)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

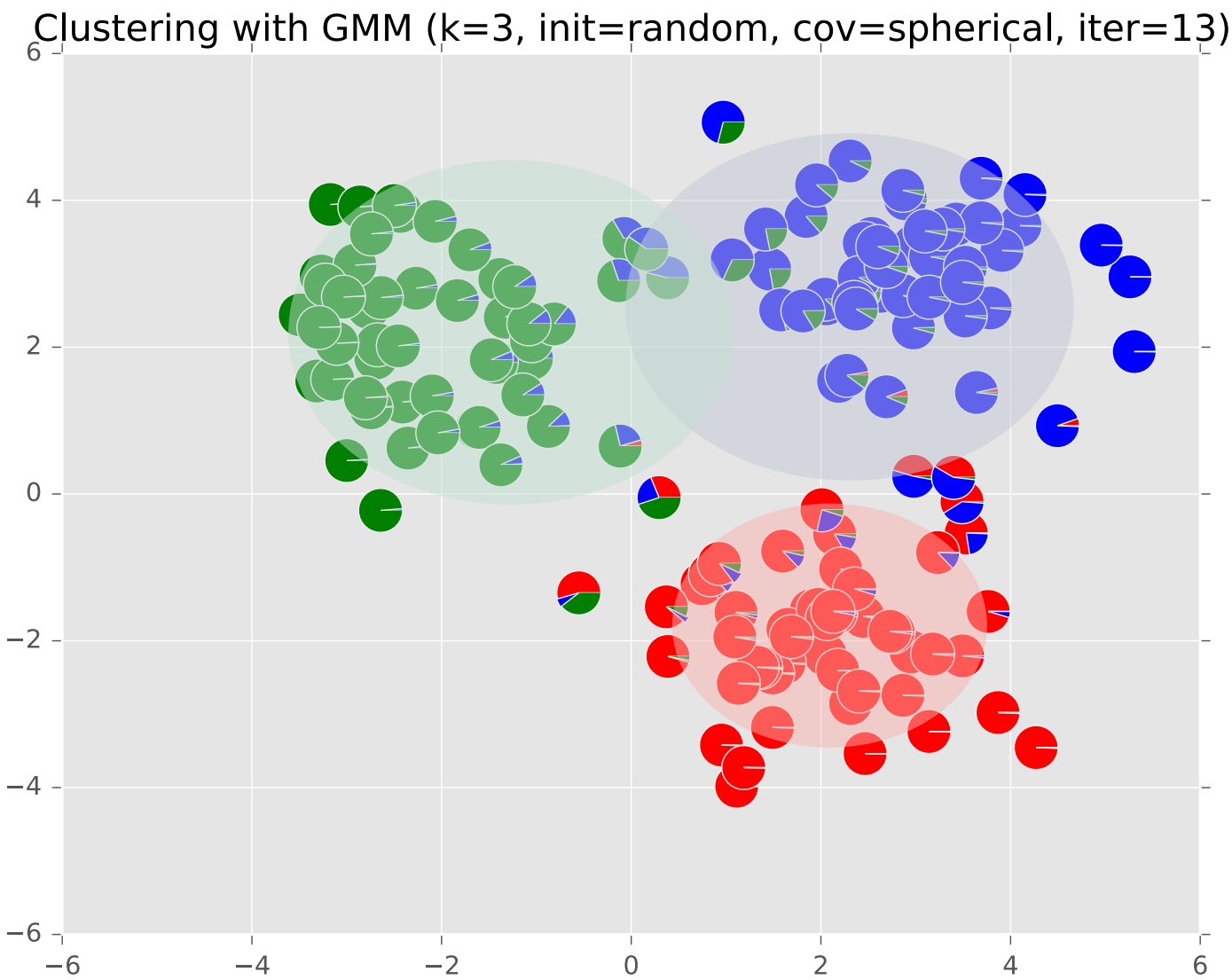
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=11)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

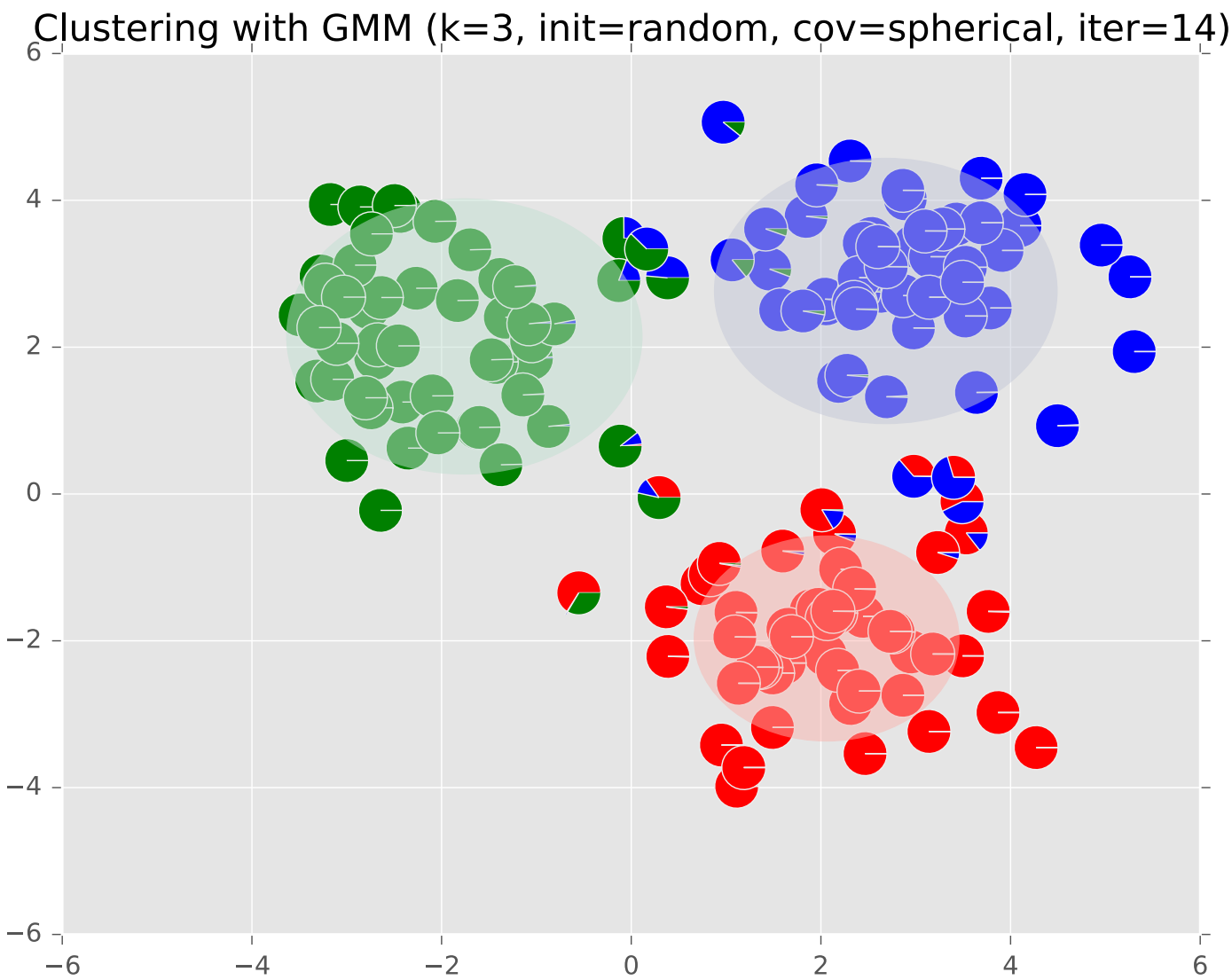
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=12)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

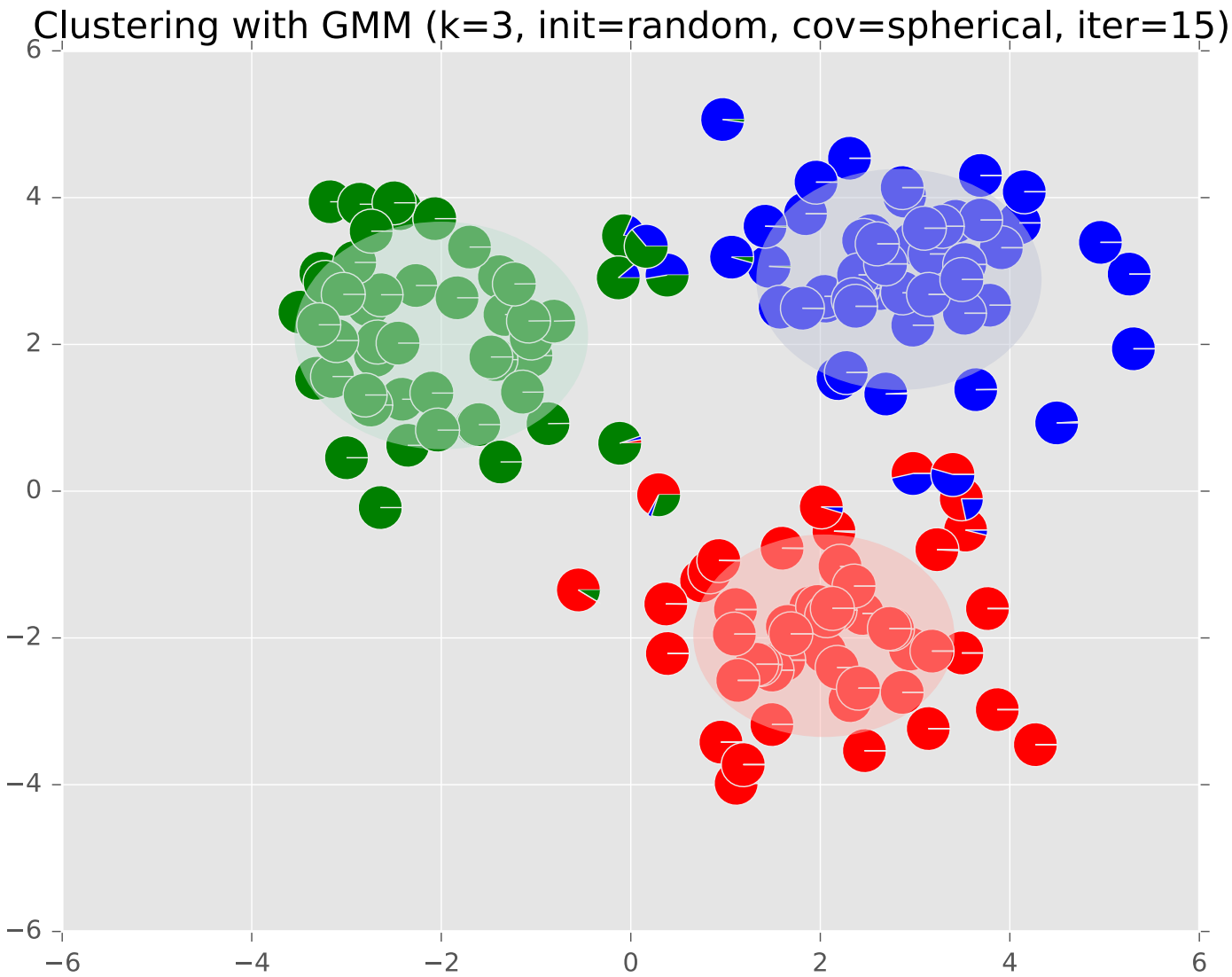
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=13)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

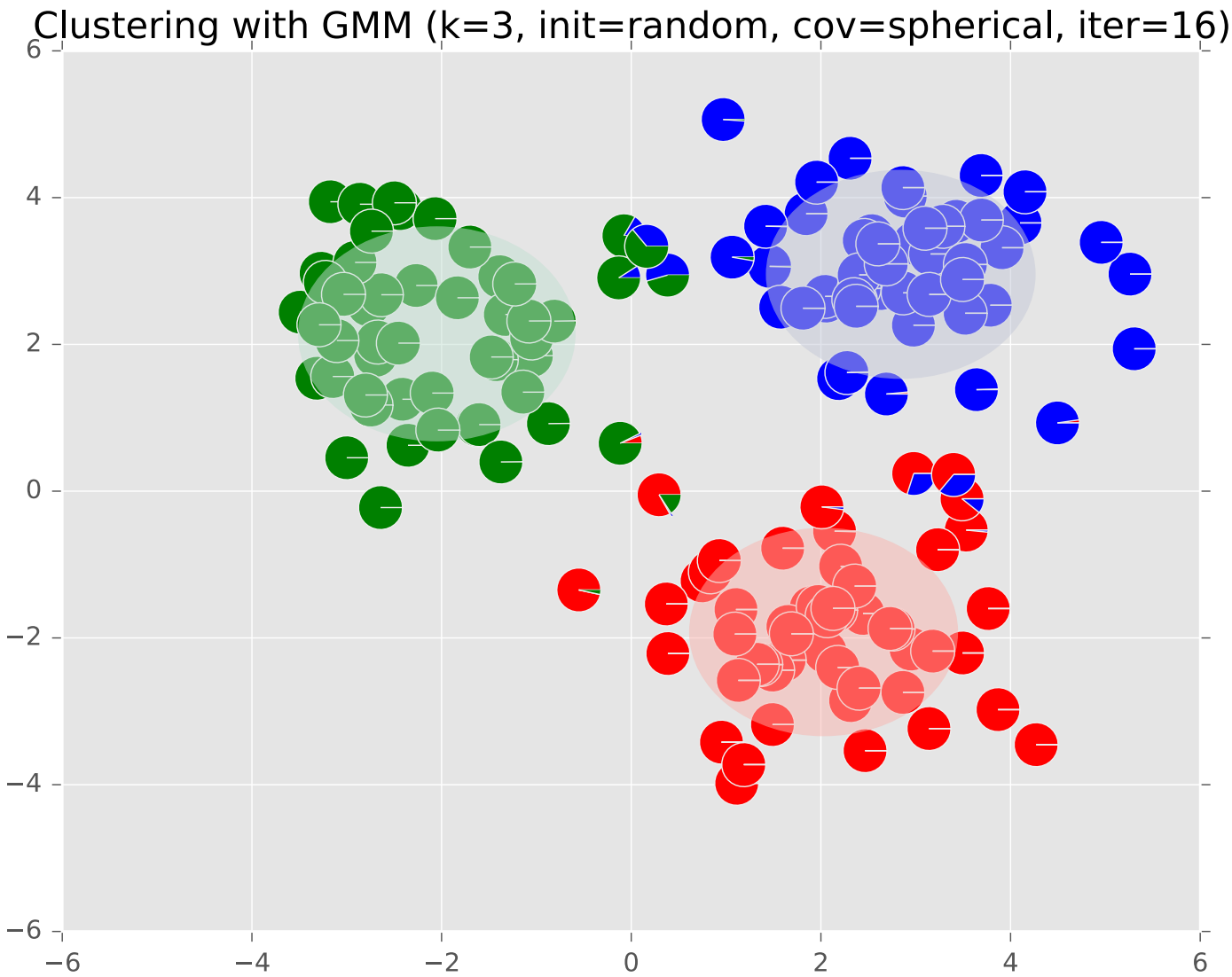
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=14)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

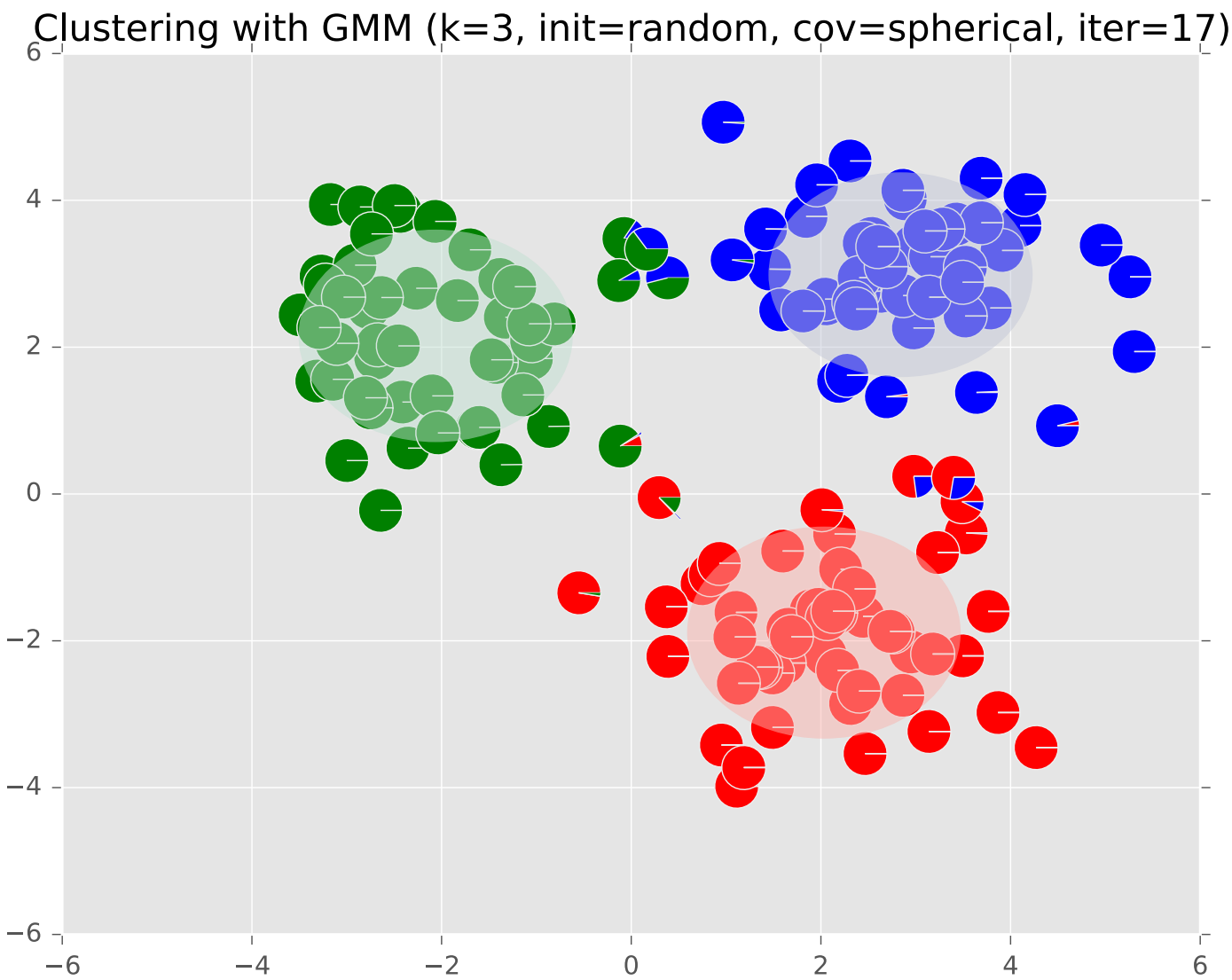
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=15)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

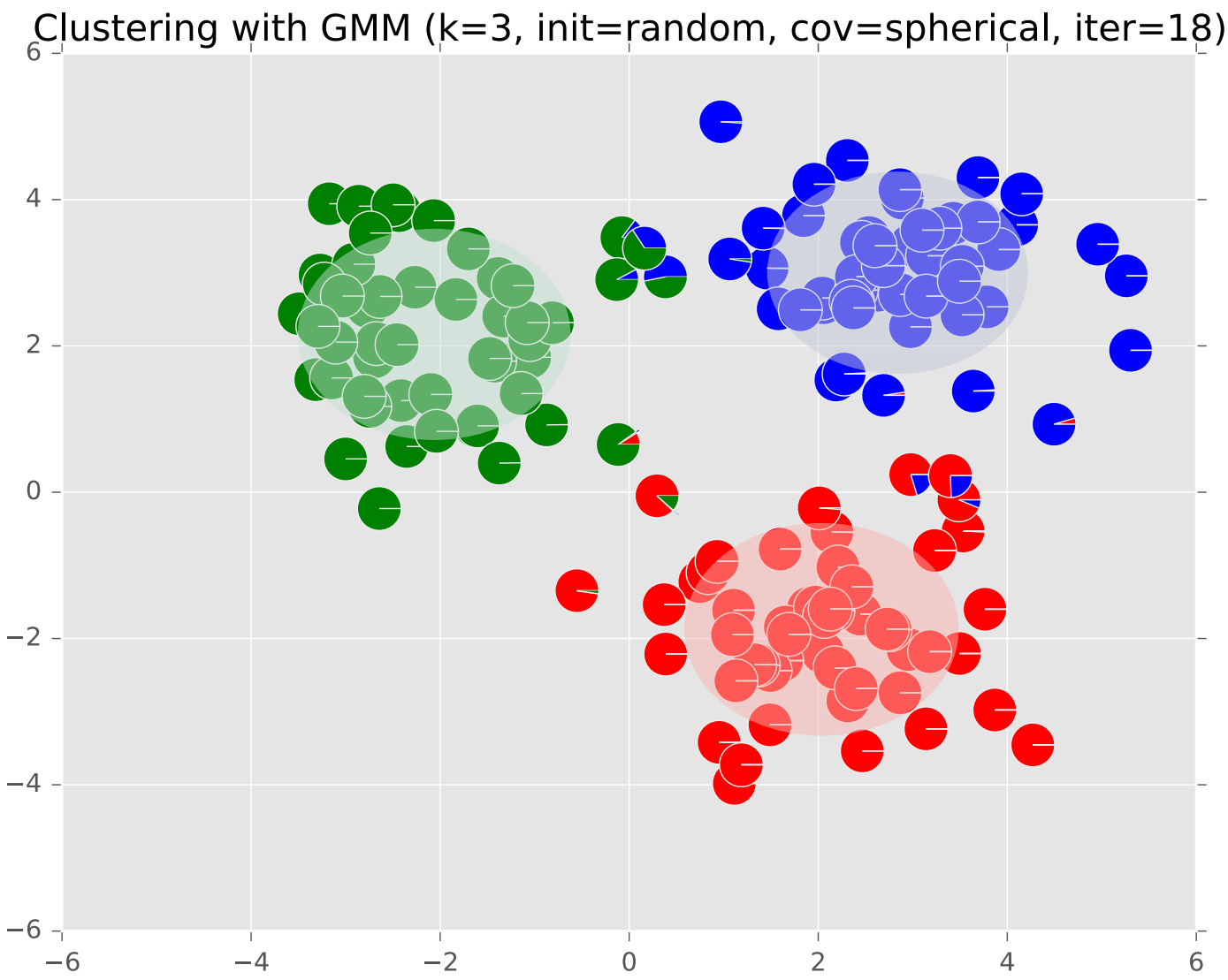
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=16)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

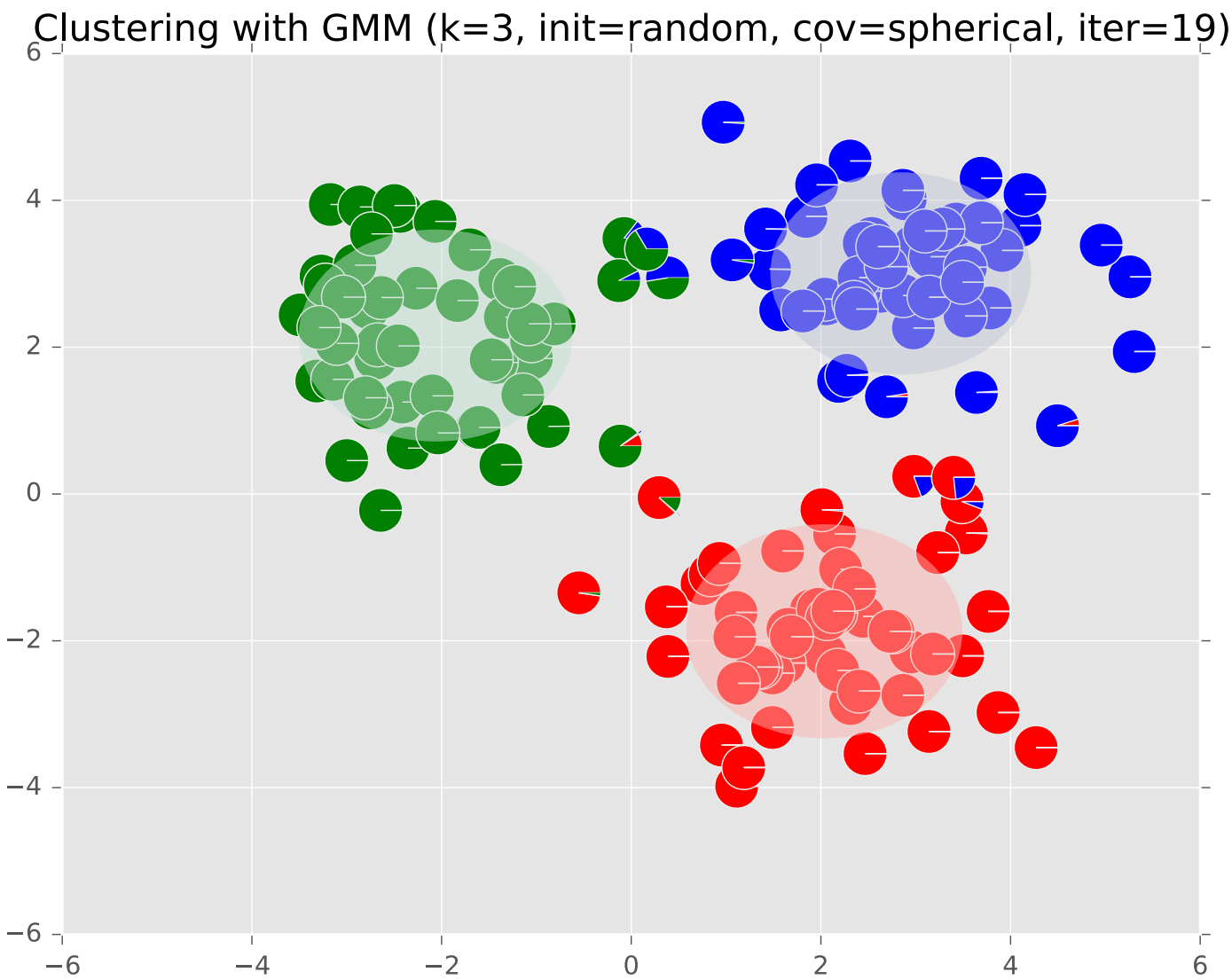
Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=17)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=18)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM


Clustering with GMM (k=3, init=random, cov=spherical, iter=19)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

K-Means vs. GMM

Convergence:

K-Means tends to **converge** much faster than a **GMM**

Speed:

Each iteration of **K-Means** is **computationally less intensive** than each iteration of a **GMM**

Initialization:

To **initialize** a **GMM**, we typically first run **K-Means** and use the resulting cluster centers as the means of the Gaussian components

Output:

A **GMM** yields a **probability distribution** over the cluster assignment for each point; whereas **K-Means** gives a single **hard assignment**