10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

EM

+

PCA

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

PCA Readings:
Murphy 12
Bishop 12
HTF 14.5
Mitchell --

Matt Gormley
Lecture 17
March 22, 2017

Reminders

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

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: Complete Data Likelihood vs. Marginal Data Likelihood
  - Distinction #3: Latent Variables vs. Parameters
  - Objective Functions for EM
  - EM Algorithm
  - EM for GMM vs. K-Means
- **Properties of EM**
  - Nonconvexity / Local Optimization
  - Example: Grammar Induction
  - Variants of EM

Last Lecture

This Lecture

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:

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

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

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

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

Model:

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

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

(Marginal) Log-likelihood:

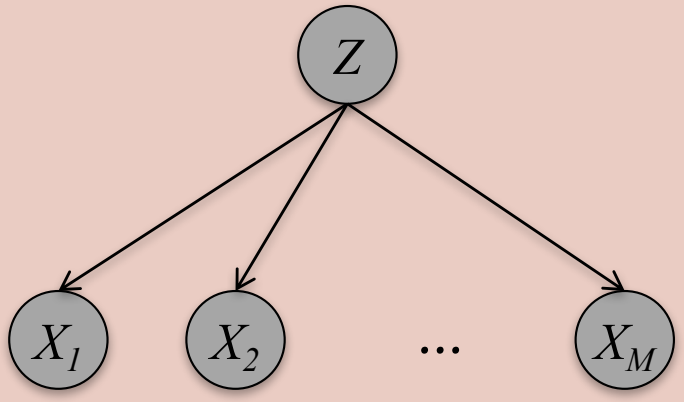
$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)})$$

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

Learning a Mixture Model

Supervised Learning: The parameters decouple!

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



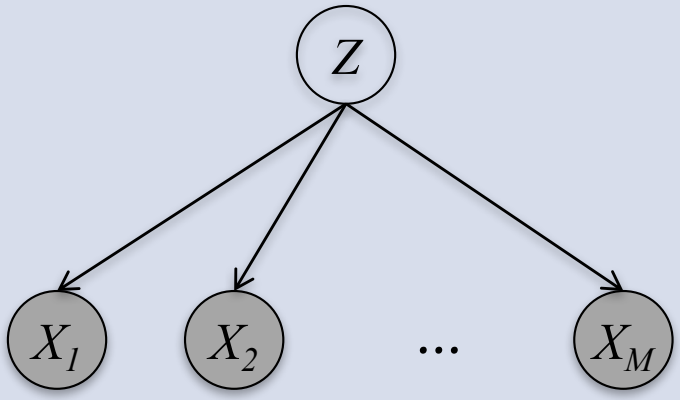
$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z^{(i)}) p_{\boldsymbol{\phi}}(z^{(i)})$$

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

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

Unsupervised Learning: Parameters are coupled by marginalization.

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



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

Learning a Mixture Model

Supervised Learning: The parameters decouple!

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

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

$$\theta$$

$$\phi$$

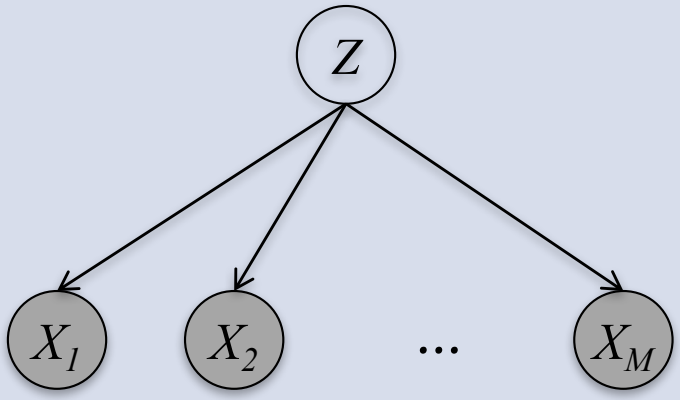
Training certainly isn't as simple as the supervised case.

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.

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



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

EXPECTATION MAXIMIZATION

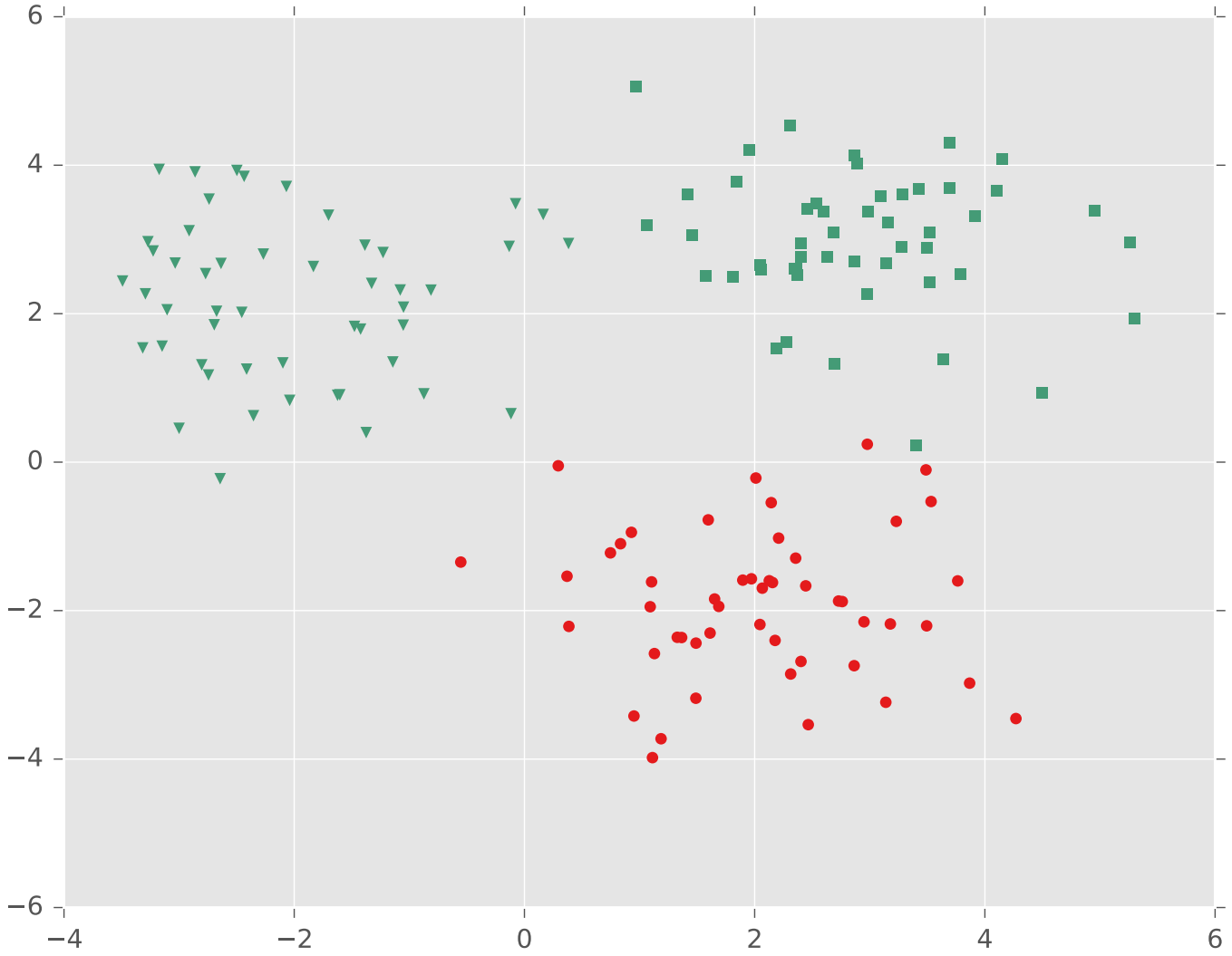
Expectation-Maximization

Whiteboard

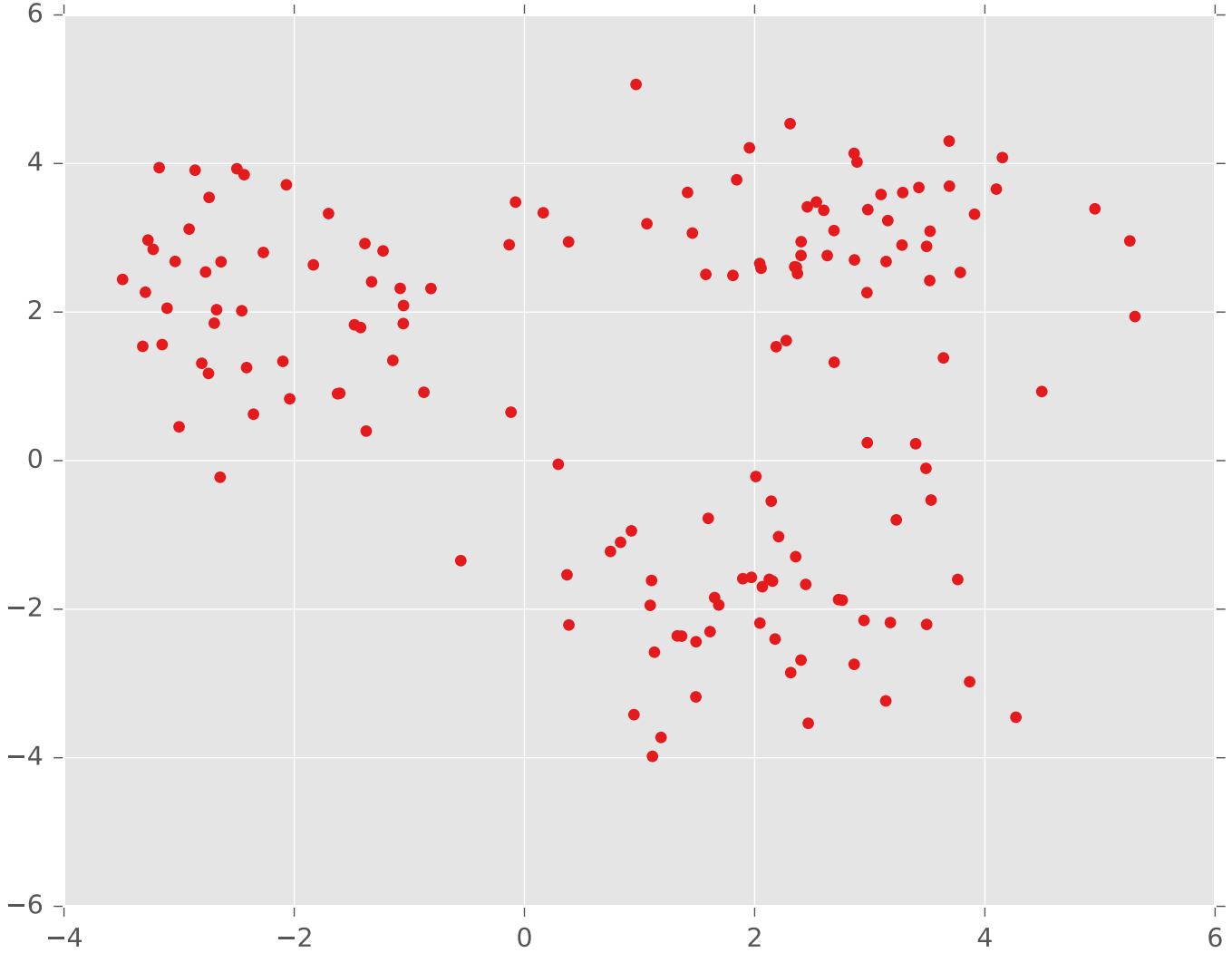
- Distinction #2: Complete Data Likelihood vs. Marginal Data Likelihood
- Distinction #3: Latent Variables vs. Parameters
- Objective Functions for EM
- EM Algorithm
- EM for GMM vs. K-Means

EXAMPLE: K-MEANS VS GMM

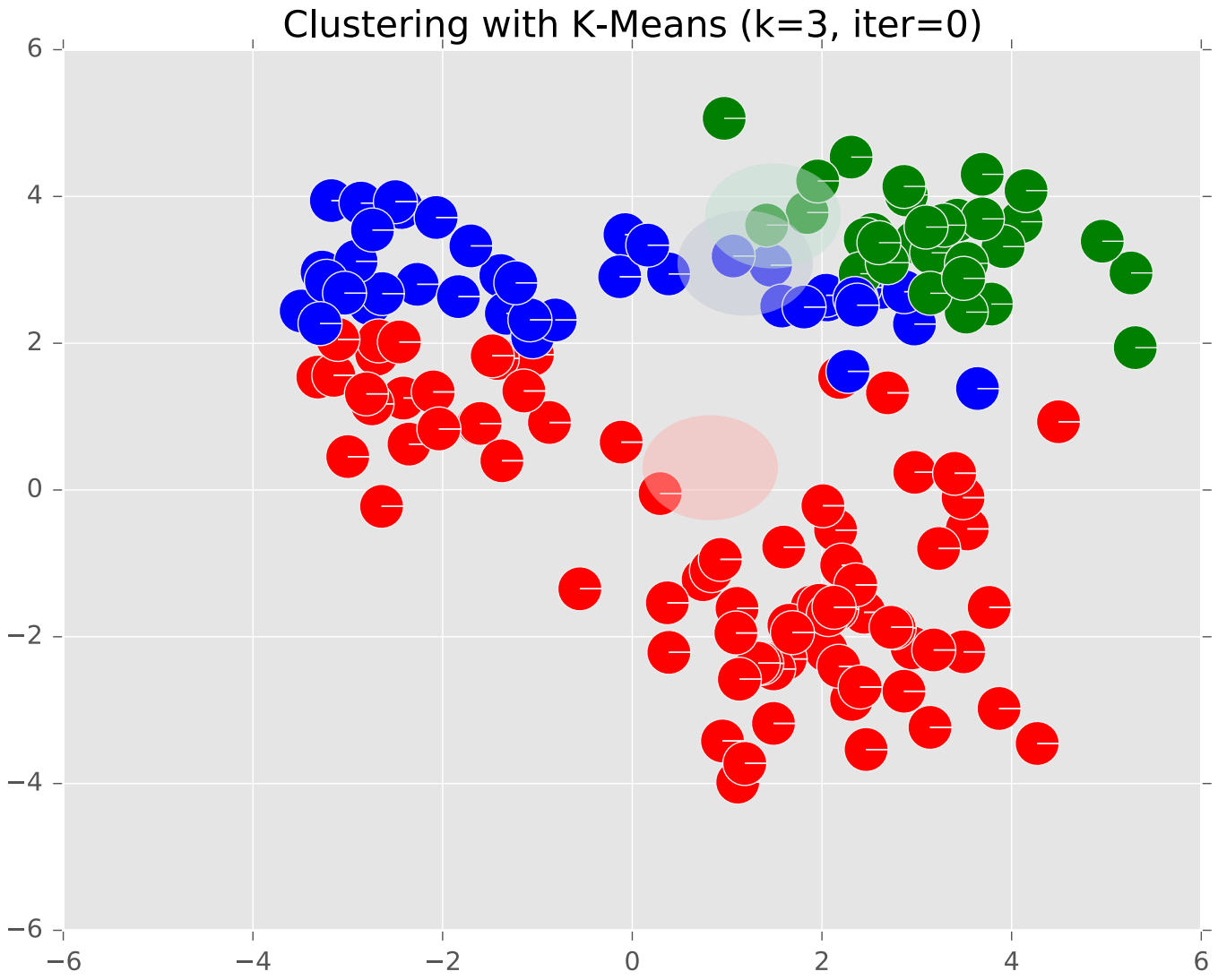
Example: K-Means



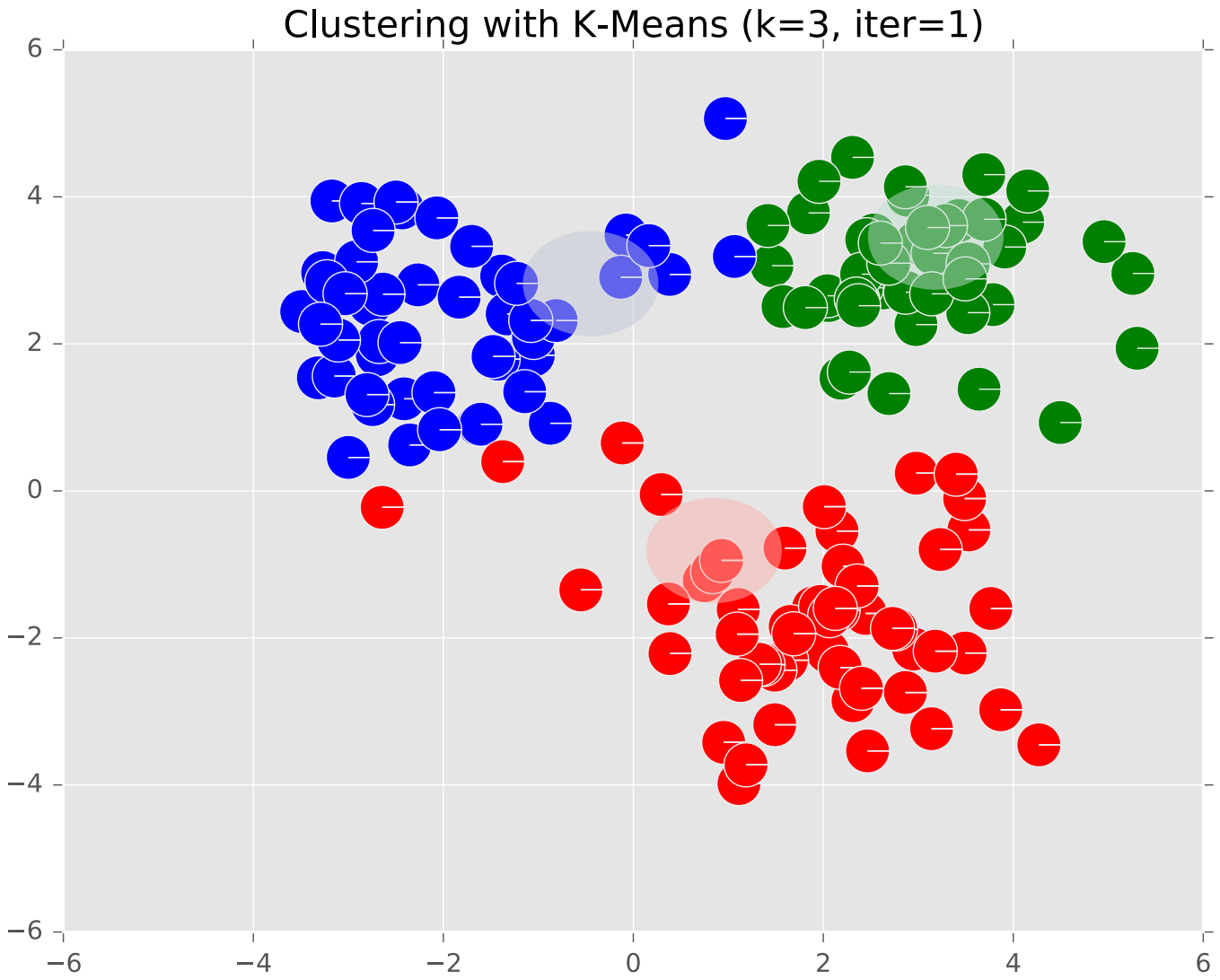
Example: K-Means



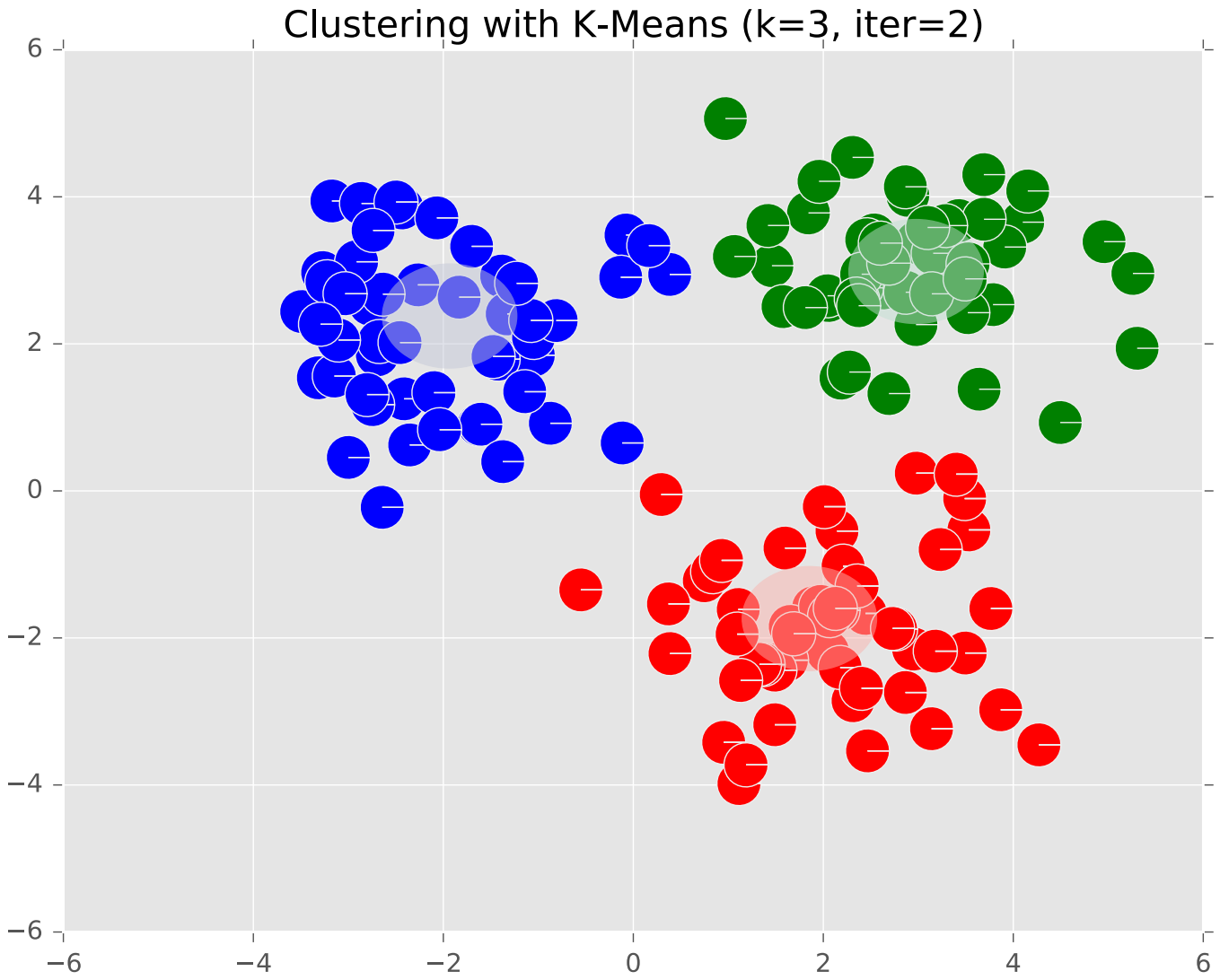
Example: K-Means



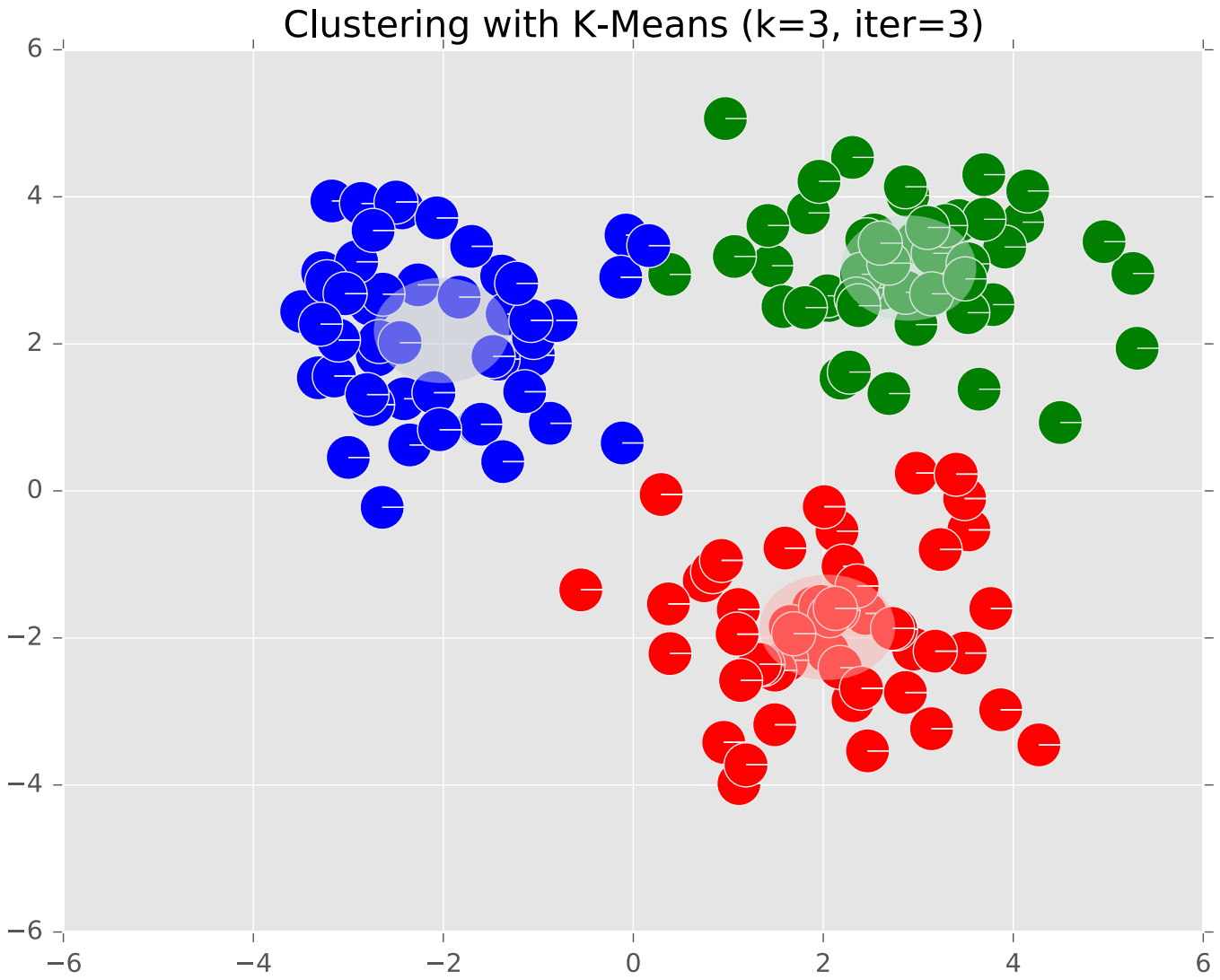
Example: K-Means



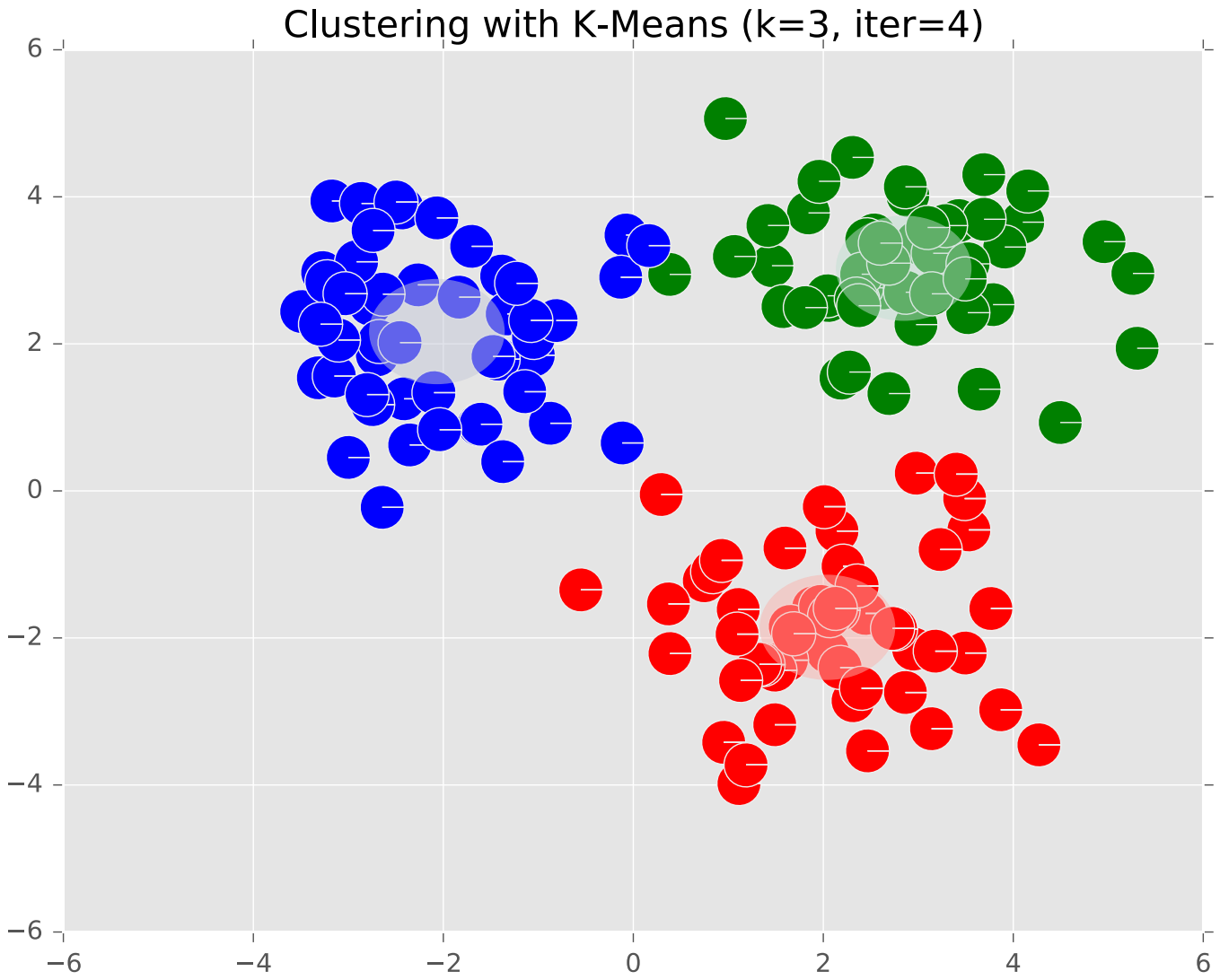
Example: K-Means



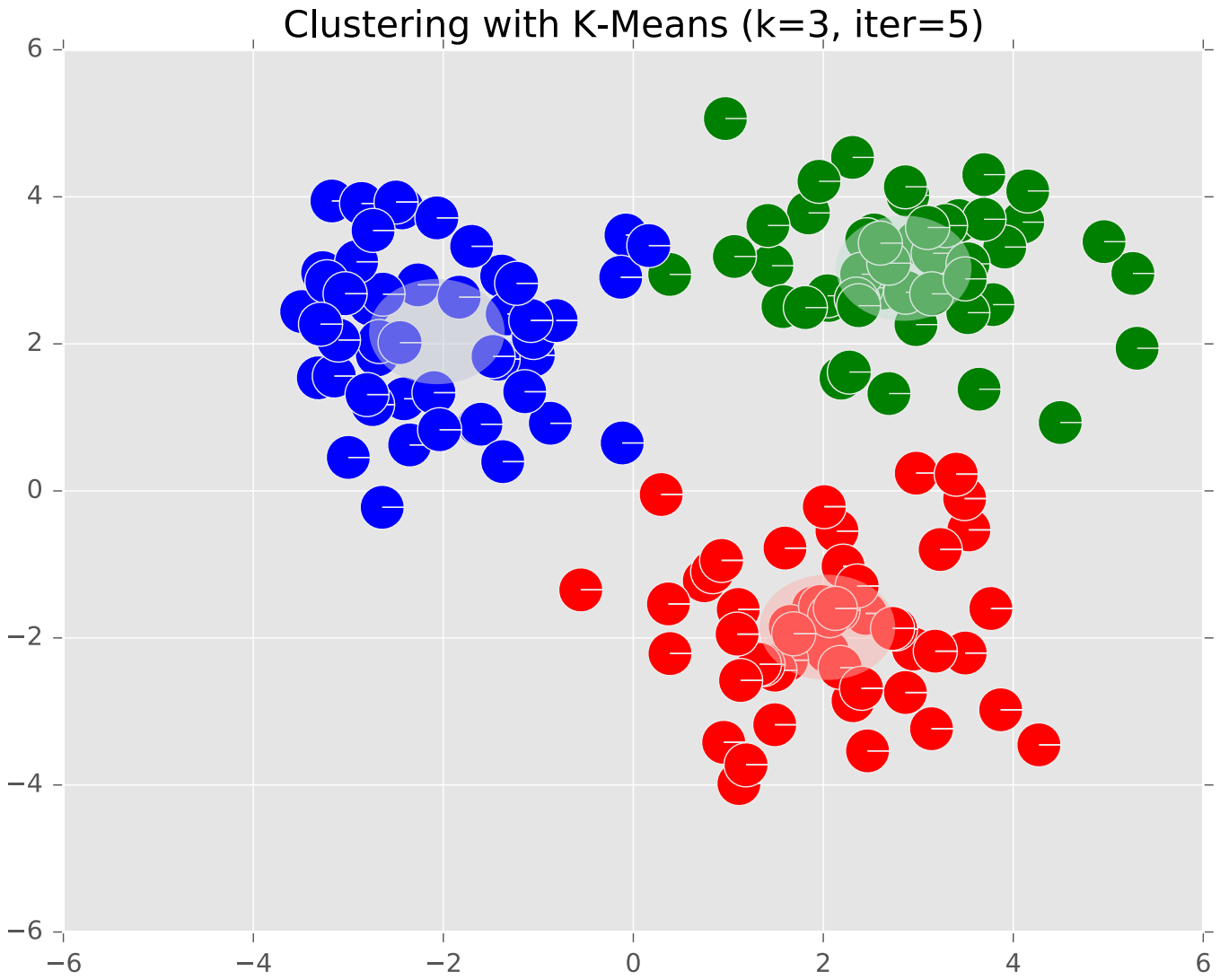
Example: K-Means



Example: K-Means

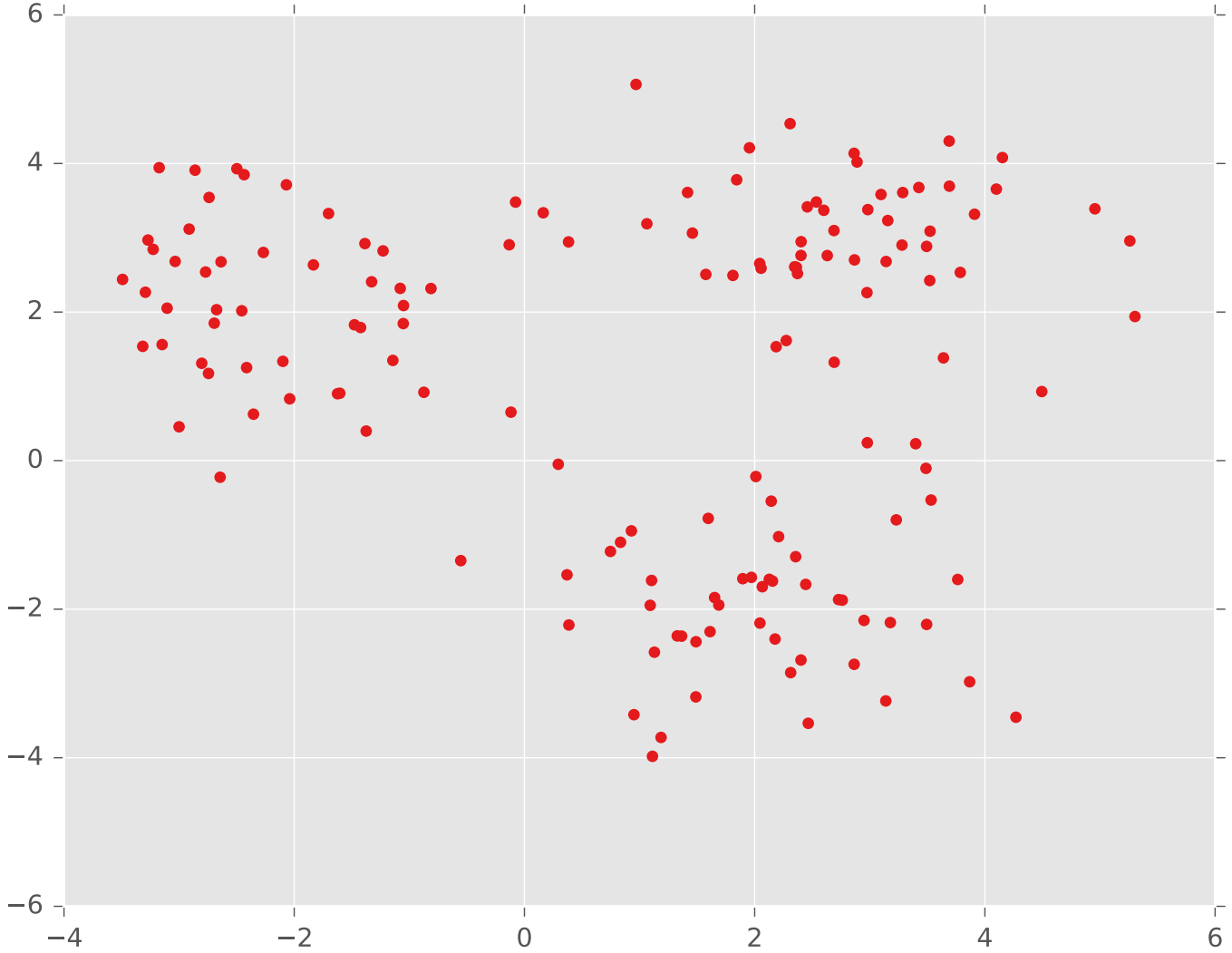


Example: K-Means

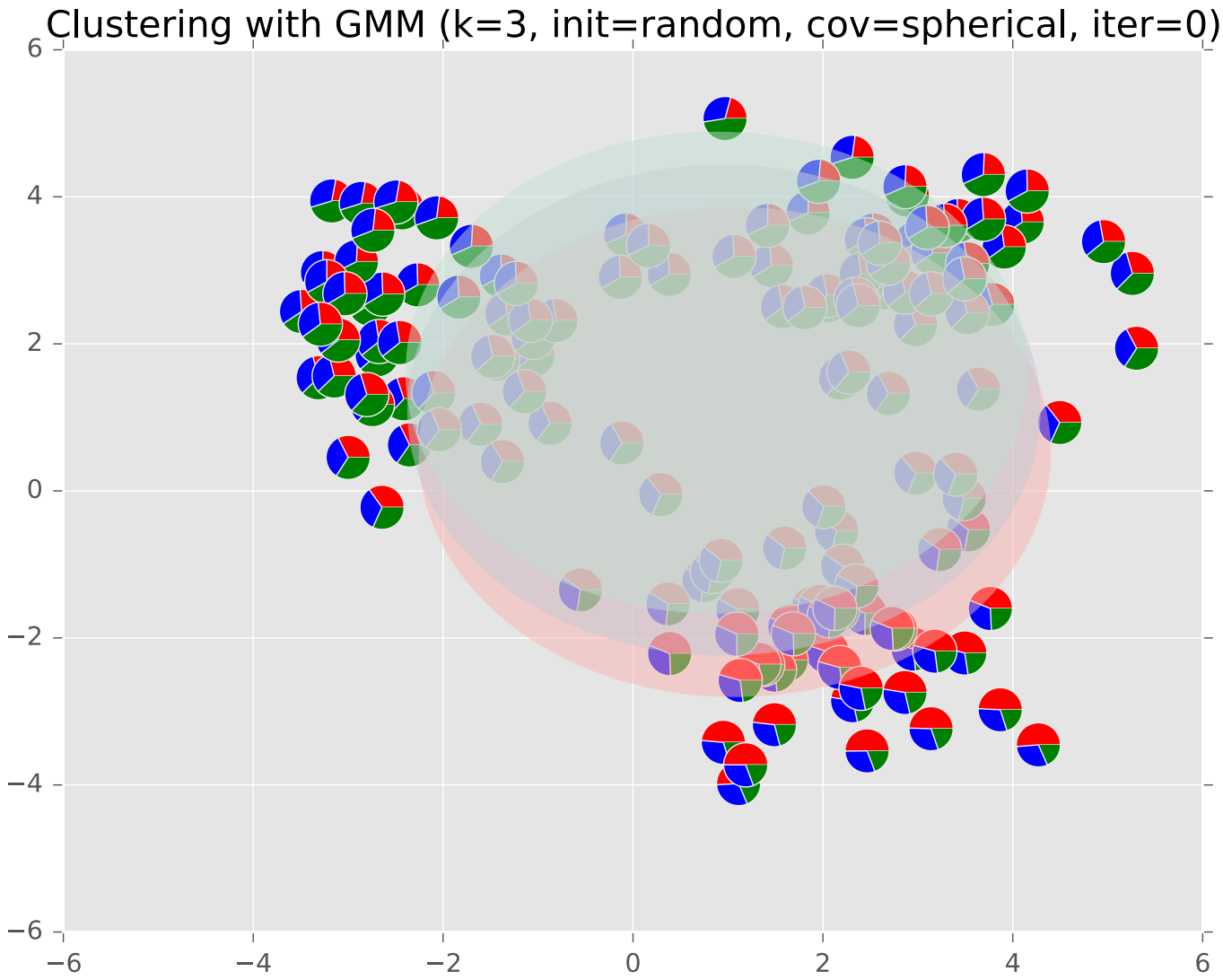


Example: GMM

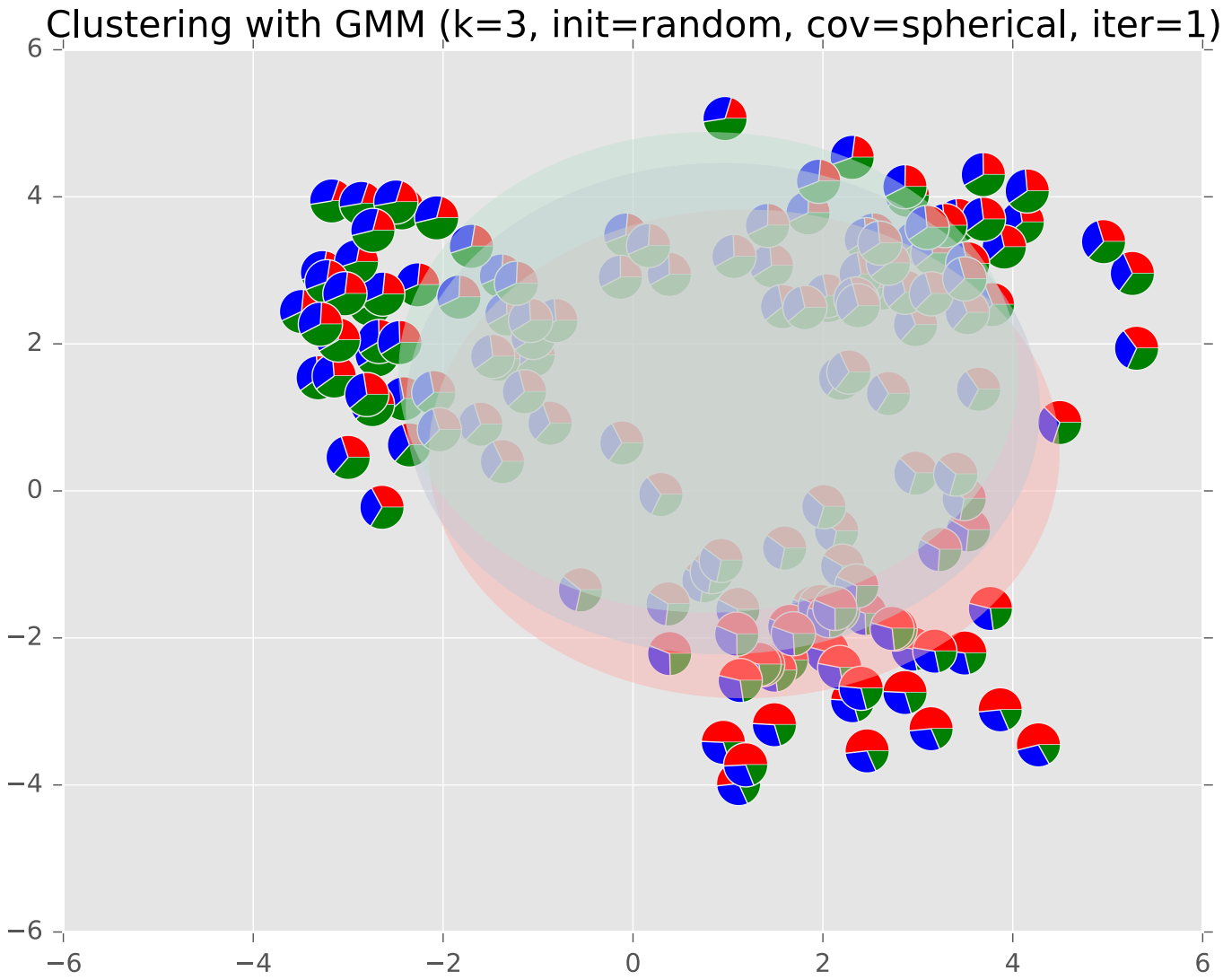
Example: GMM



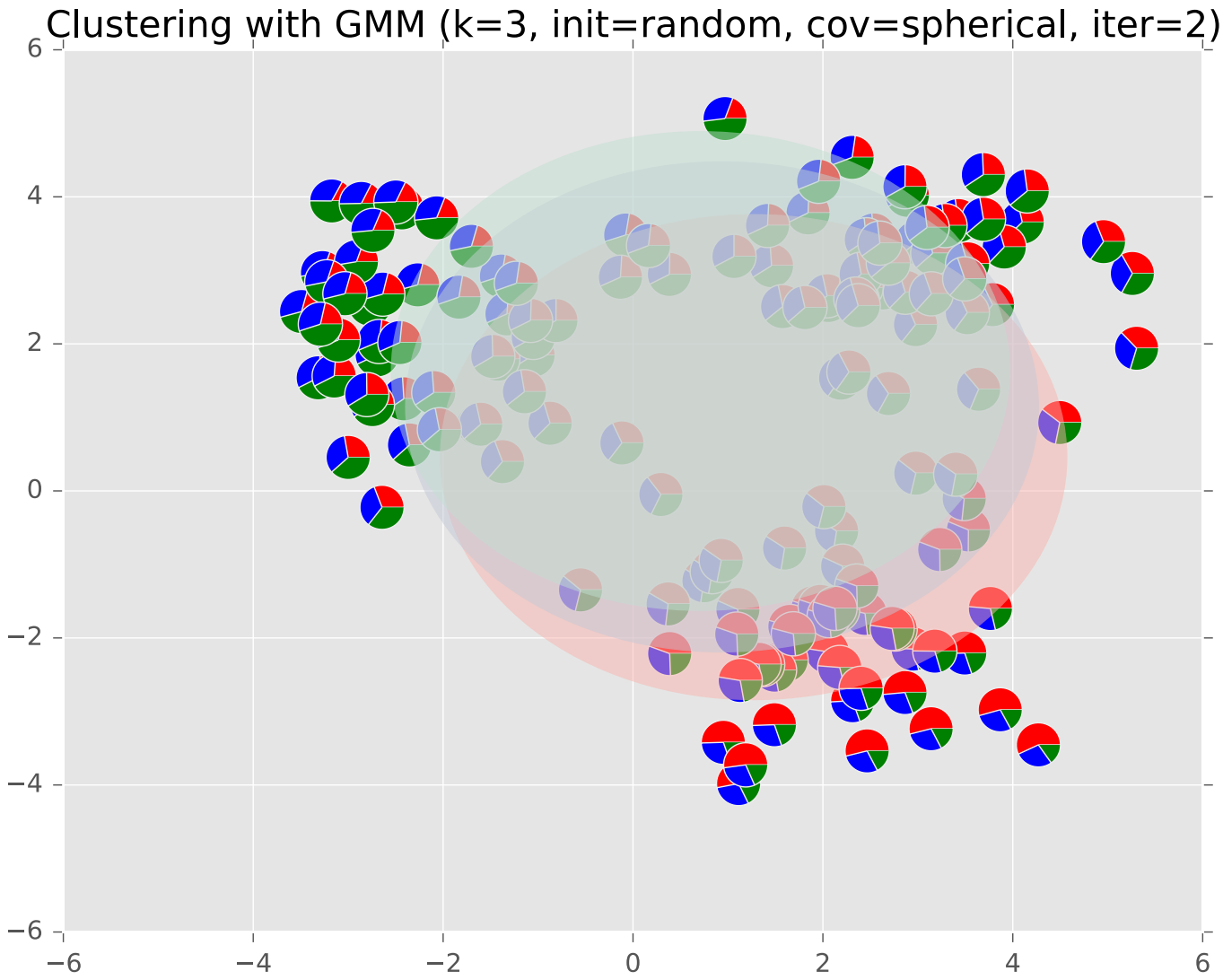
Example: GMM


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

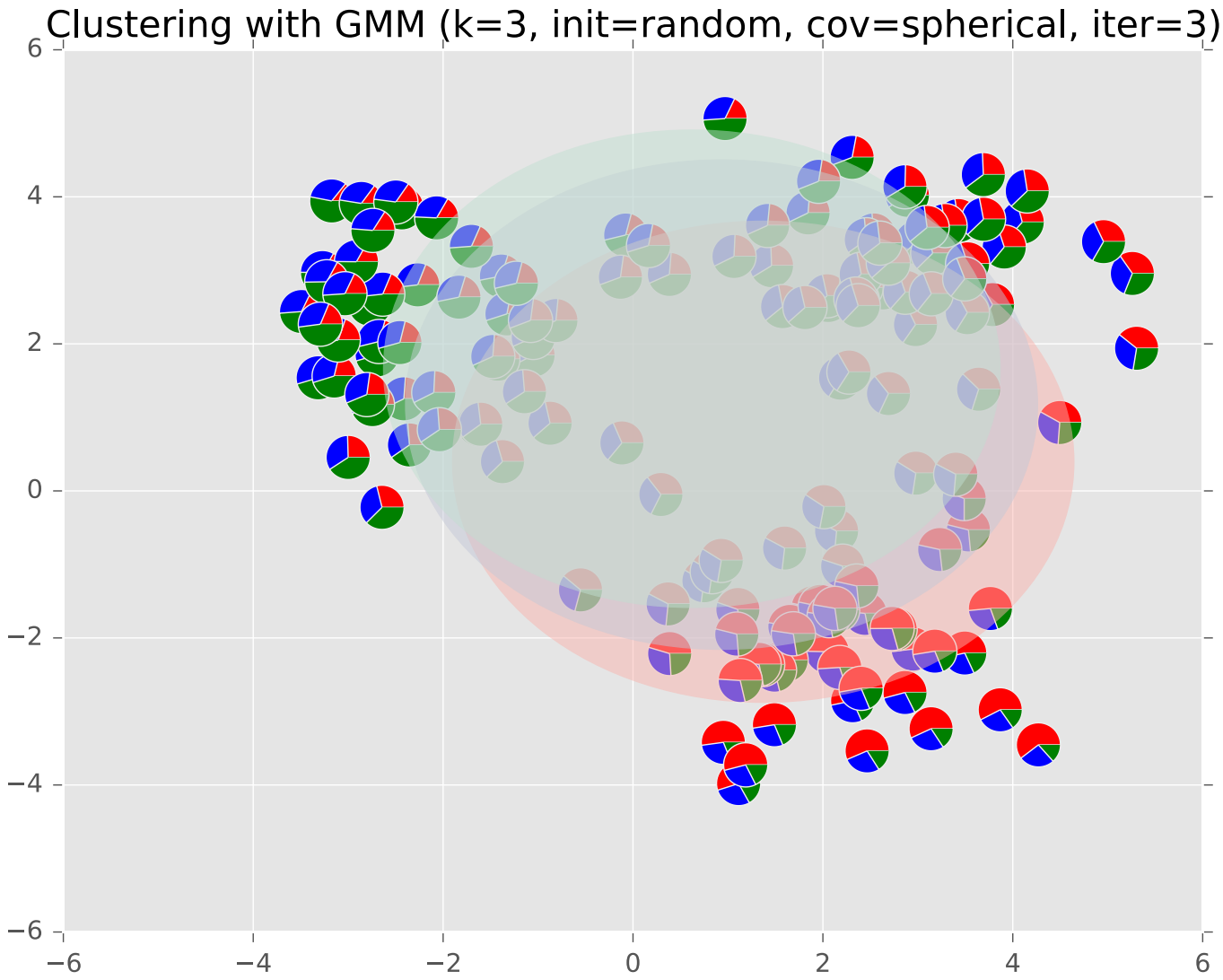
Example: GMM


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

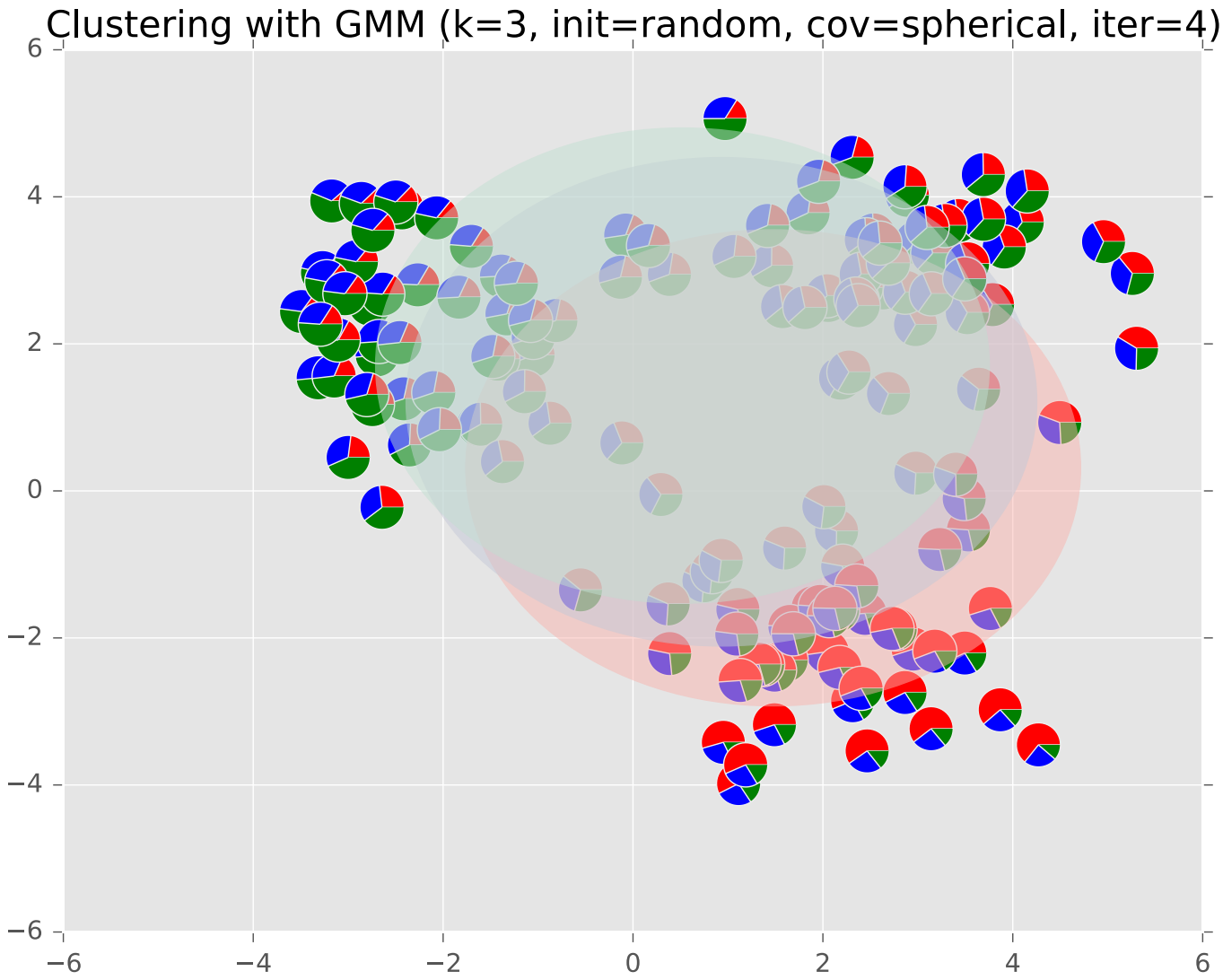
Example: GMM


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

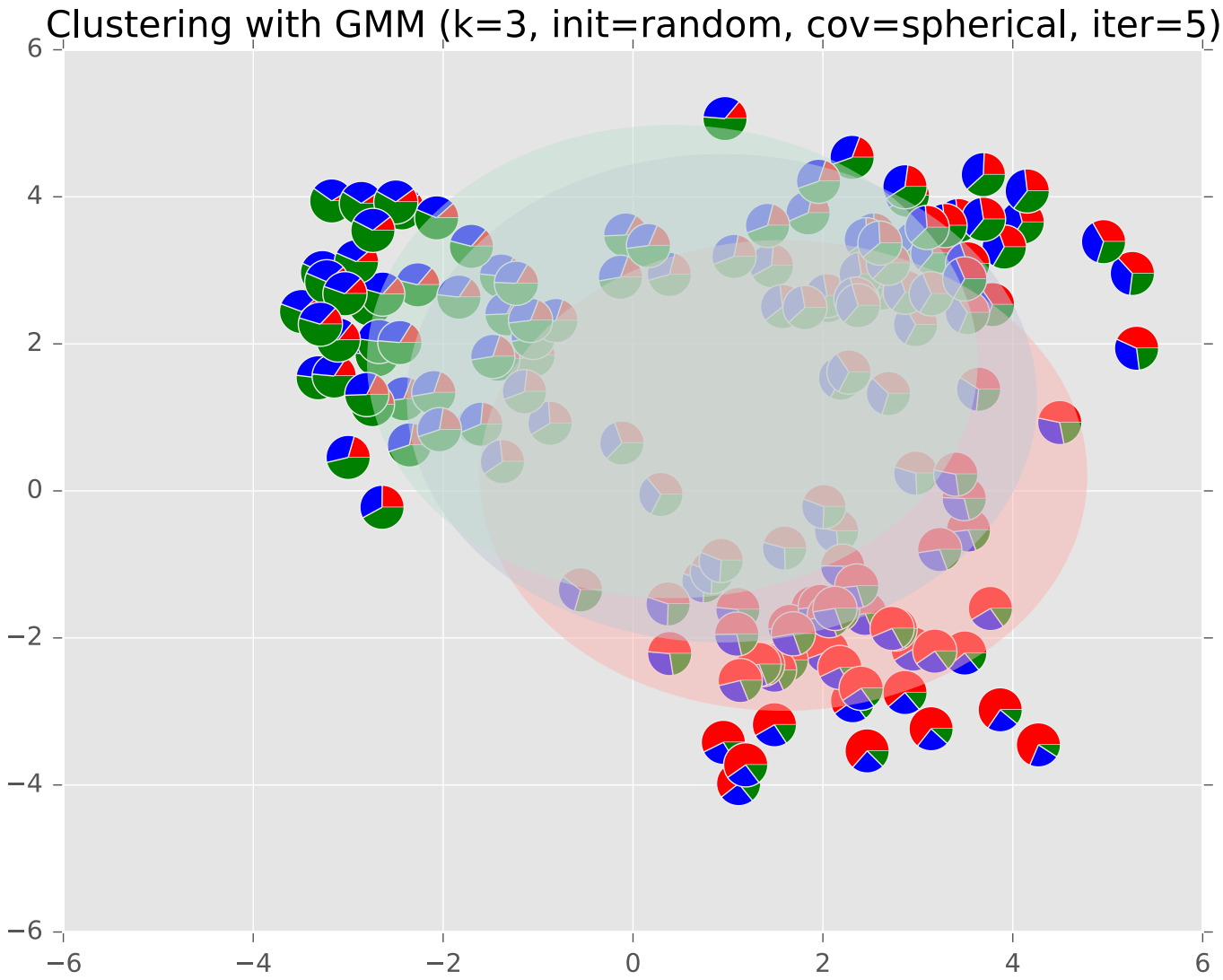
Example: GMM


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

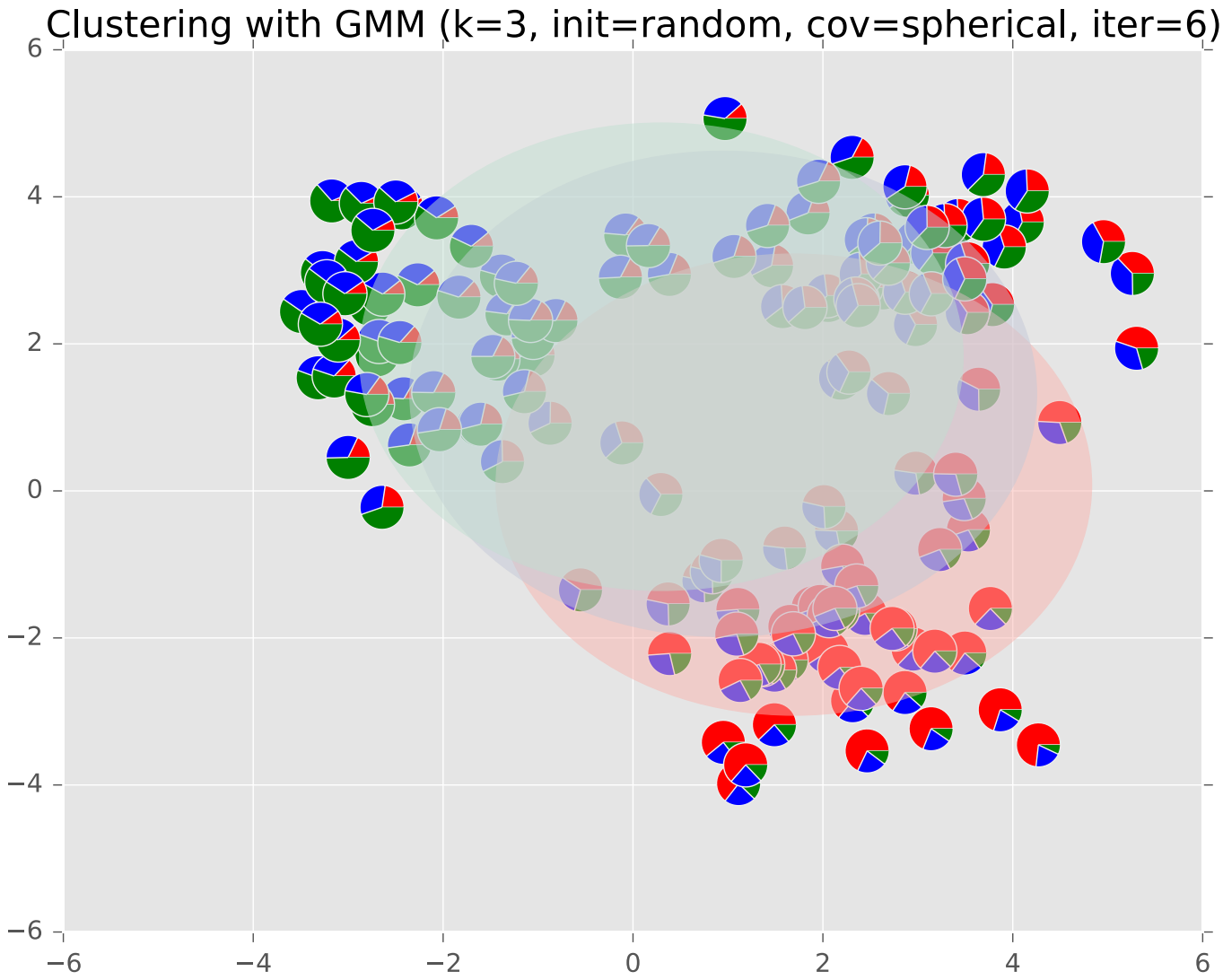
Example: GMM


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

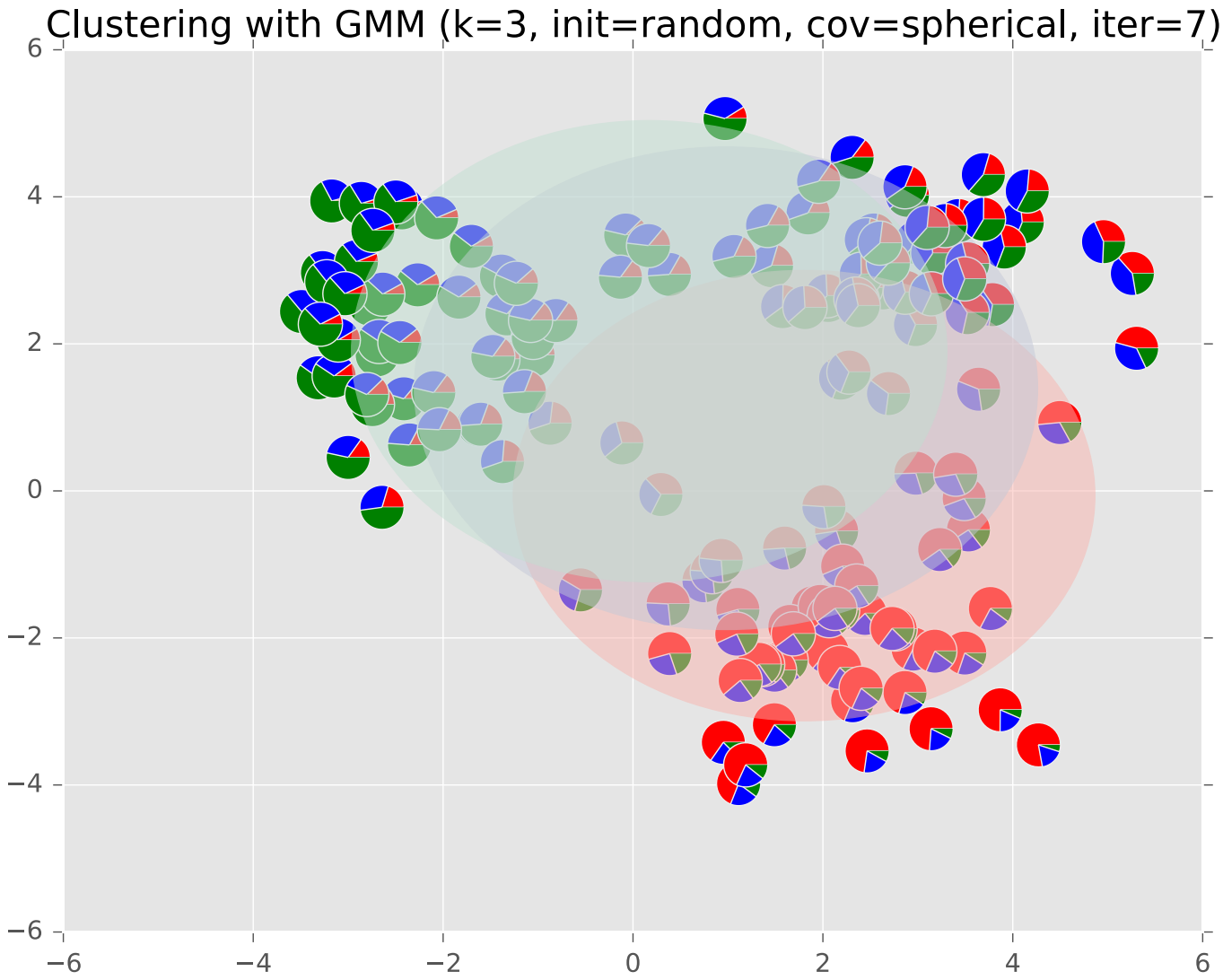
Example: GMM


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

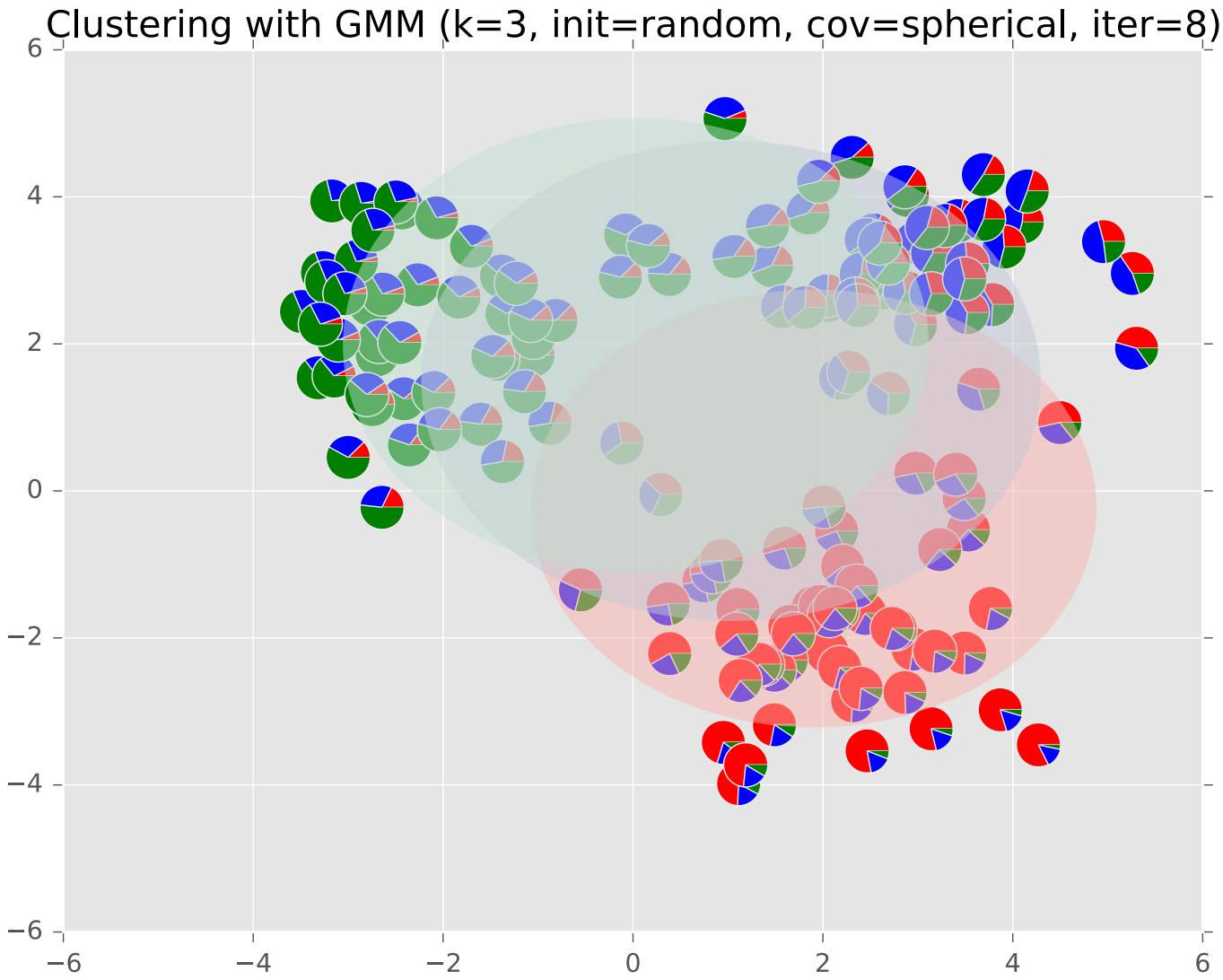
Example: GMM


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

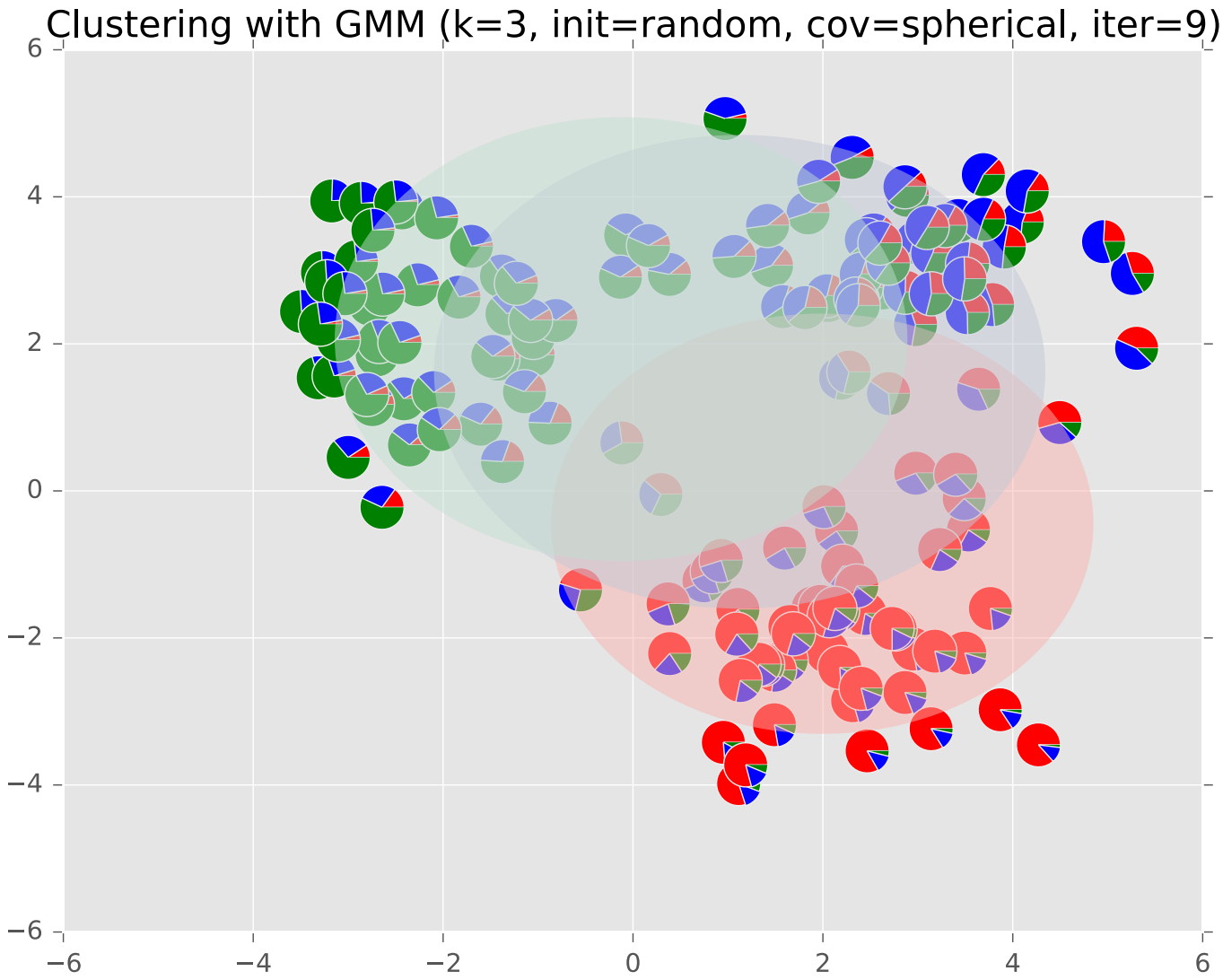
Example: GMM


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

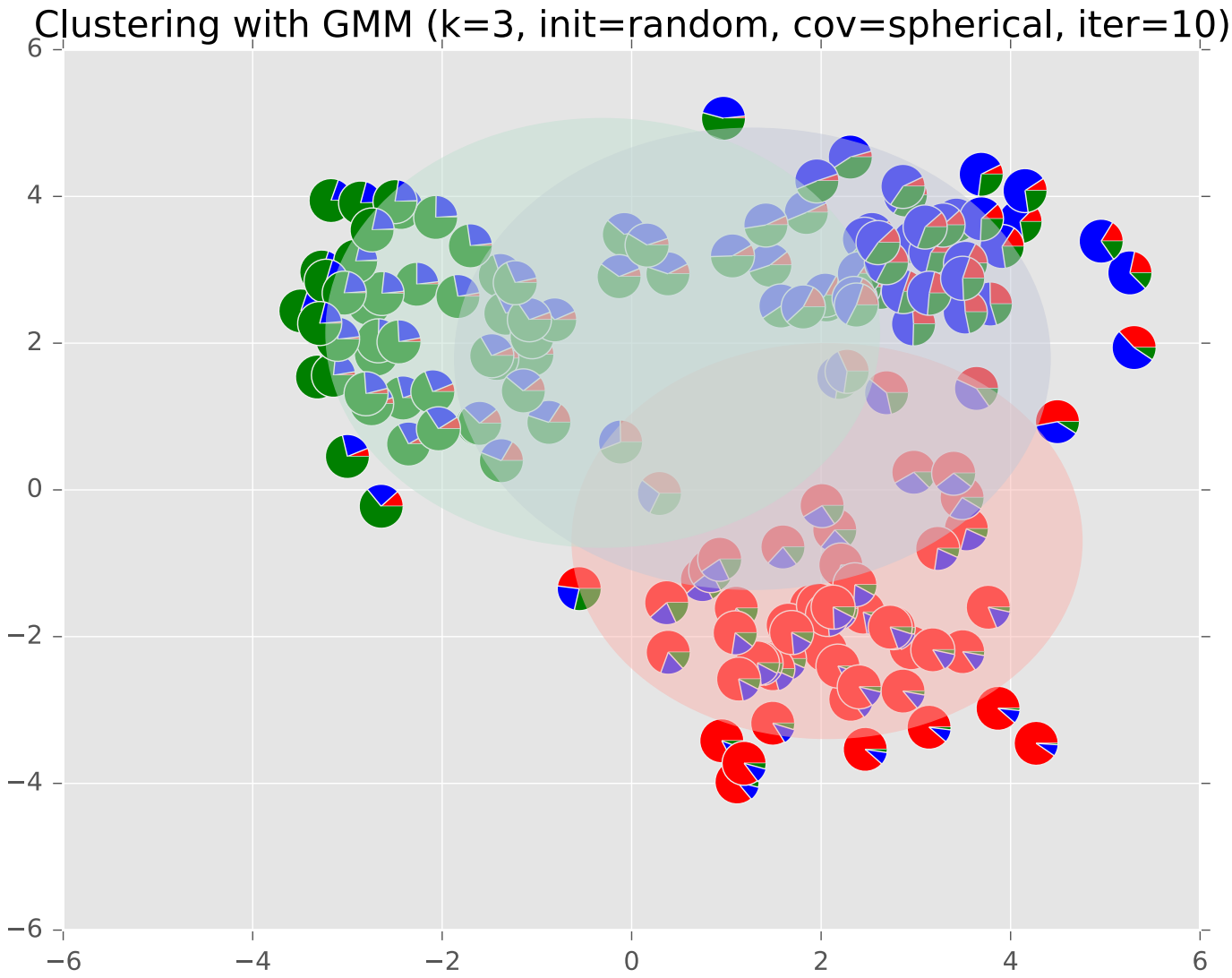
Example: GMM


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

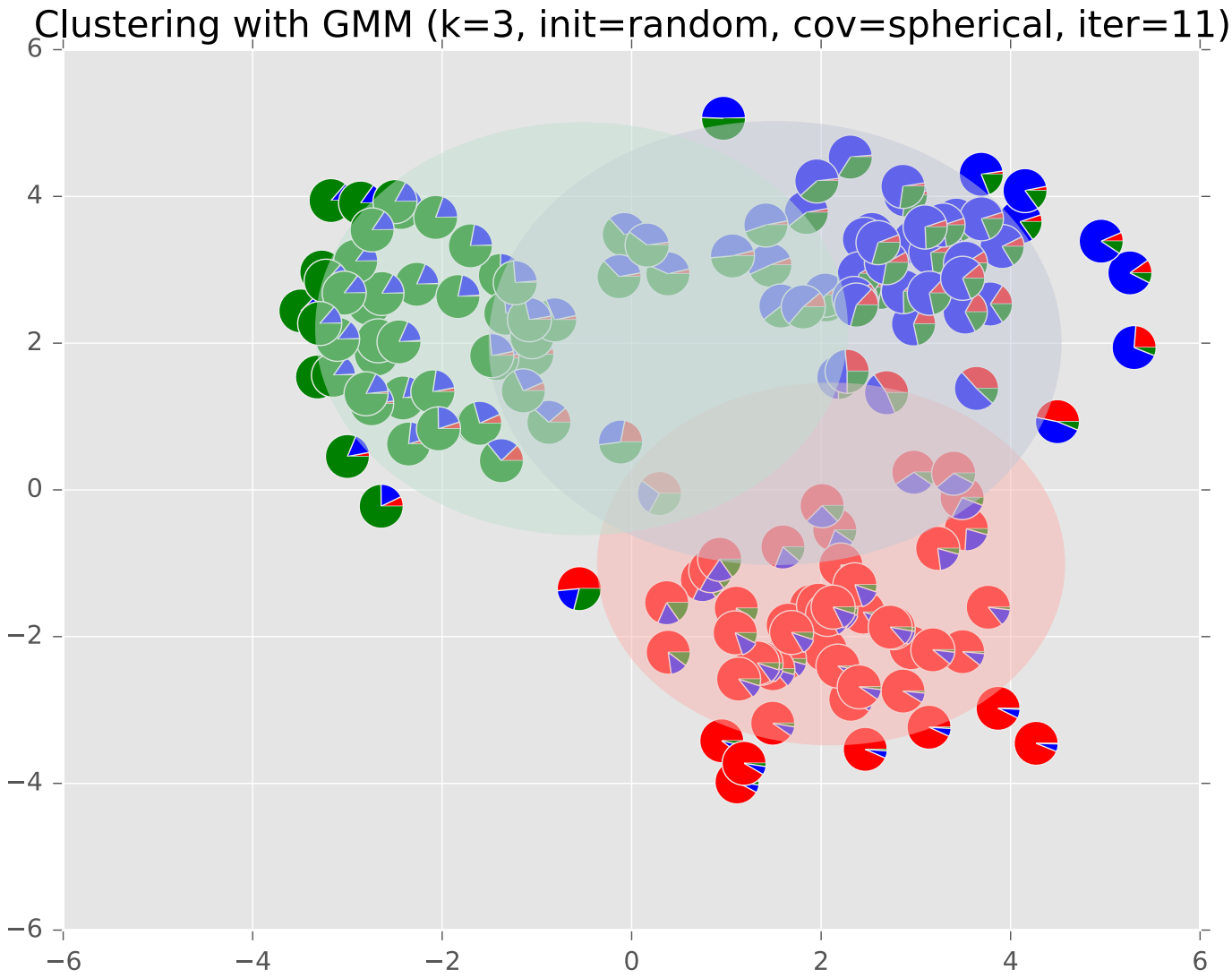
Example: GMM


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

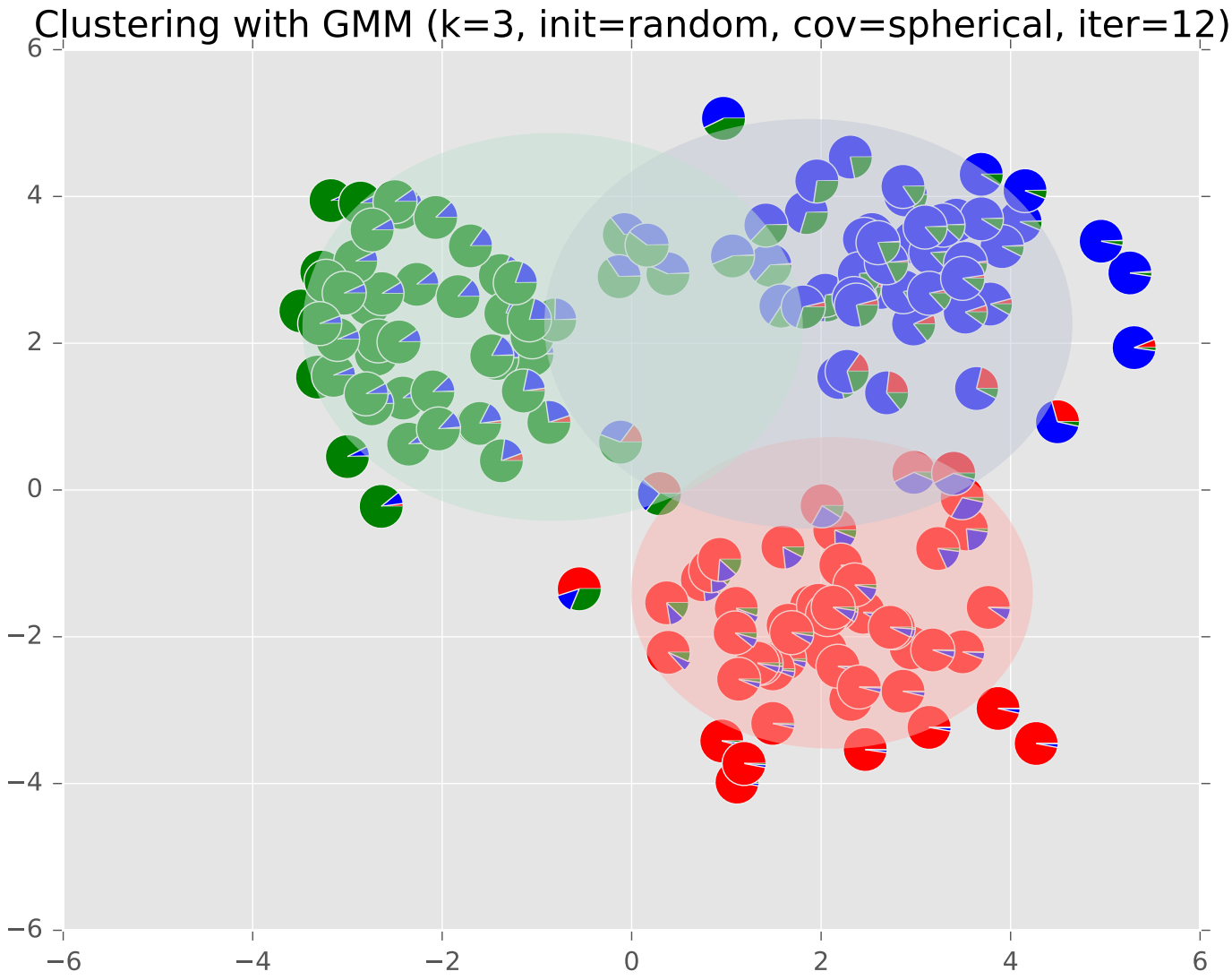
Example: GMM


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

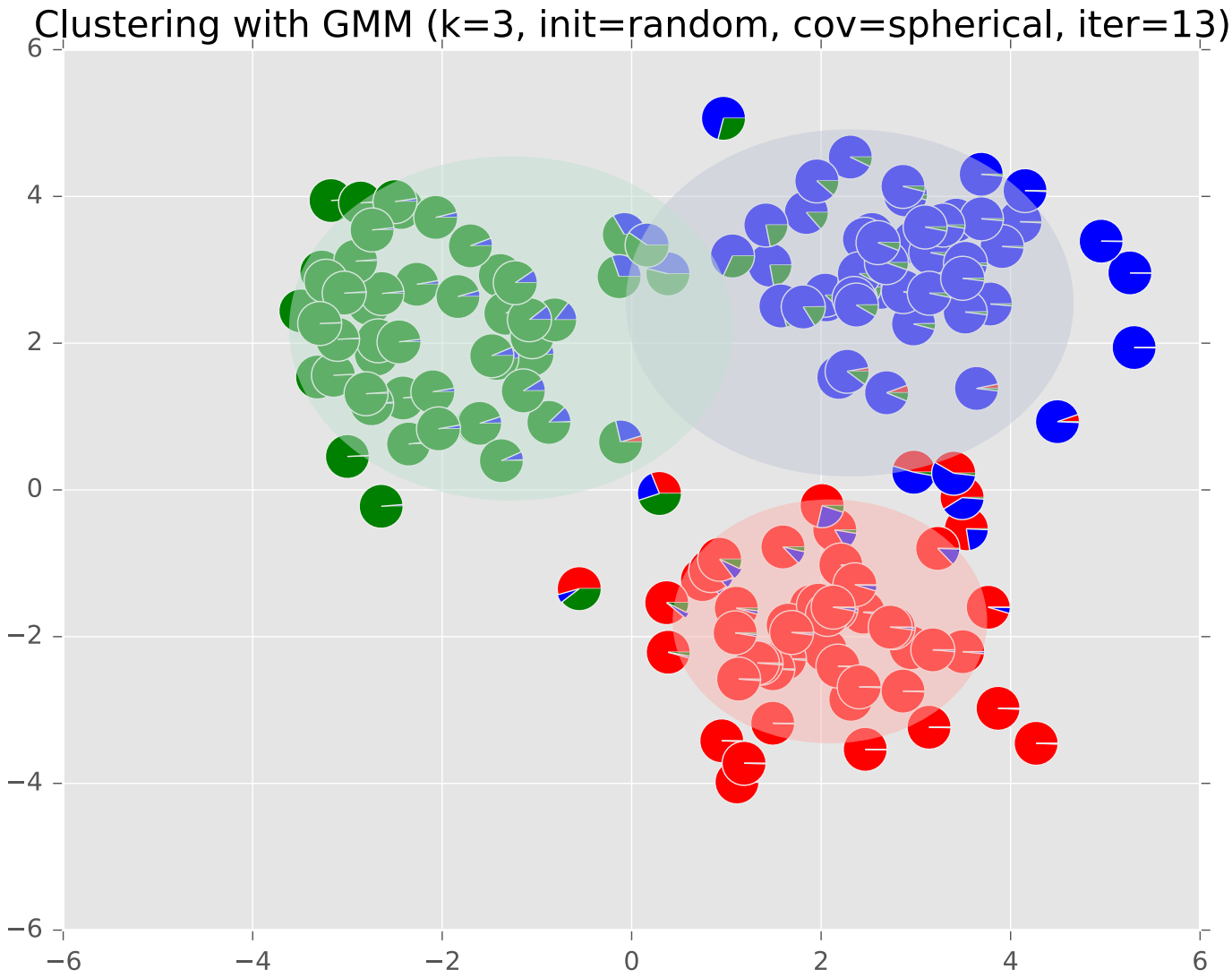
Example: GMM


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

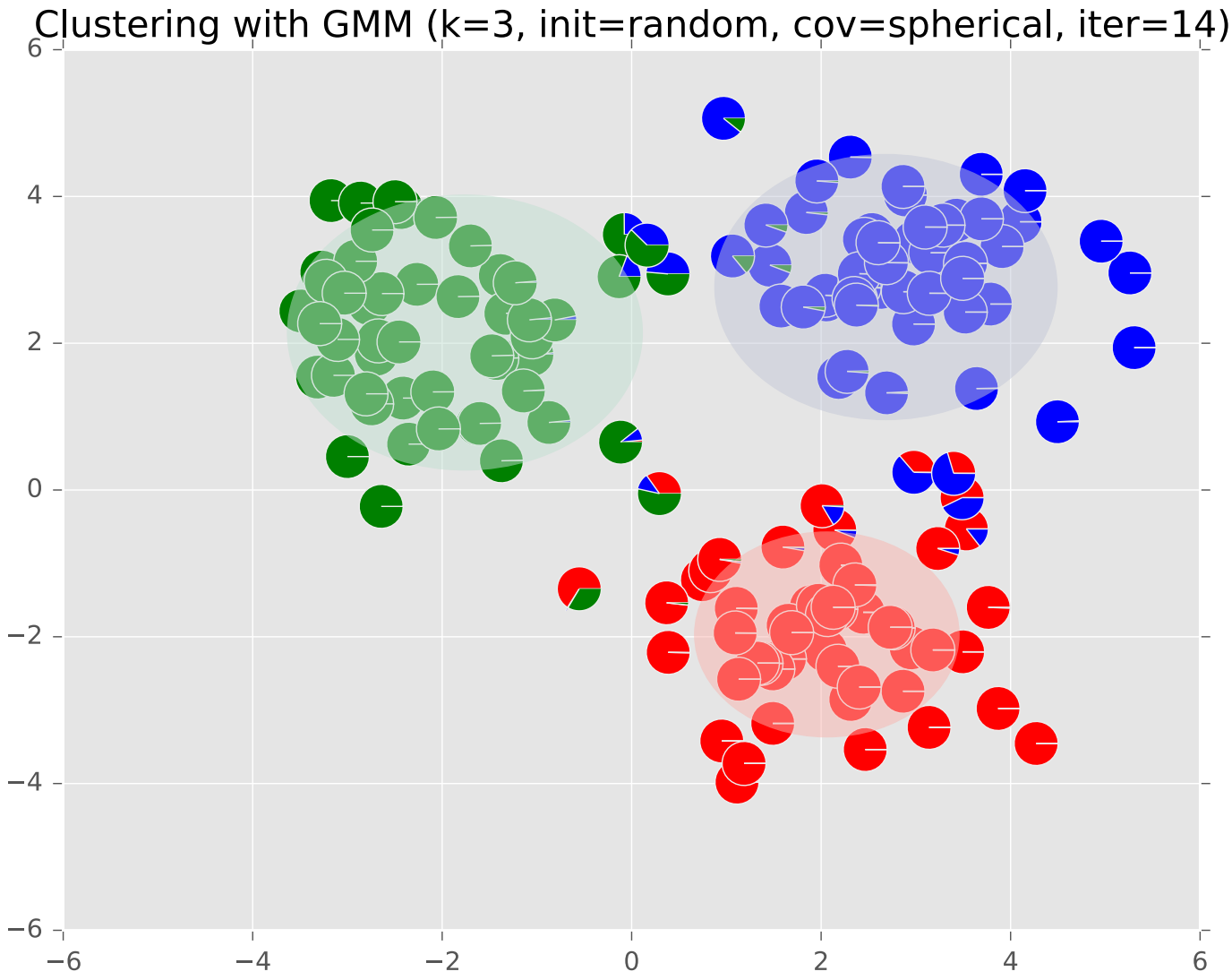
Example: GMM


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

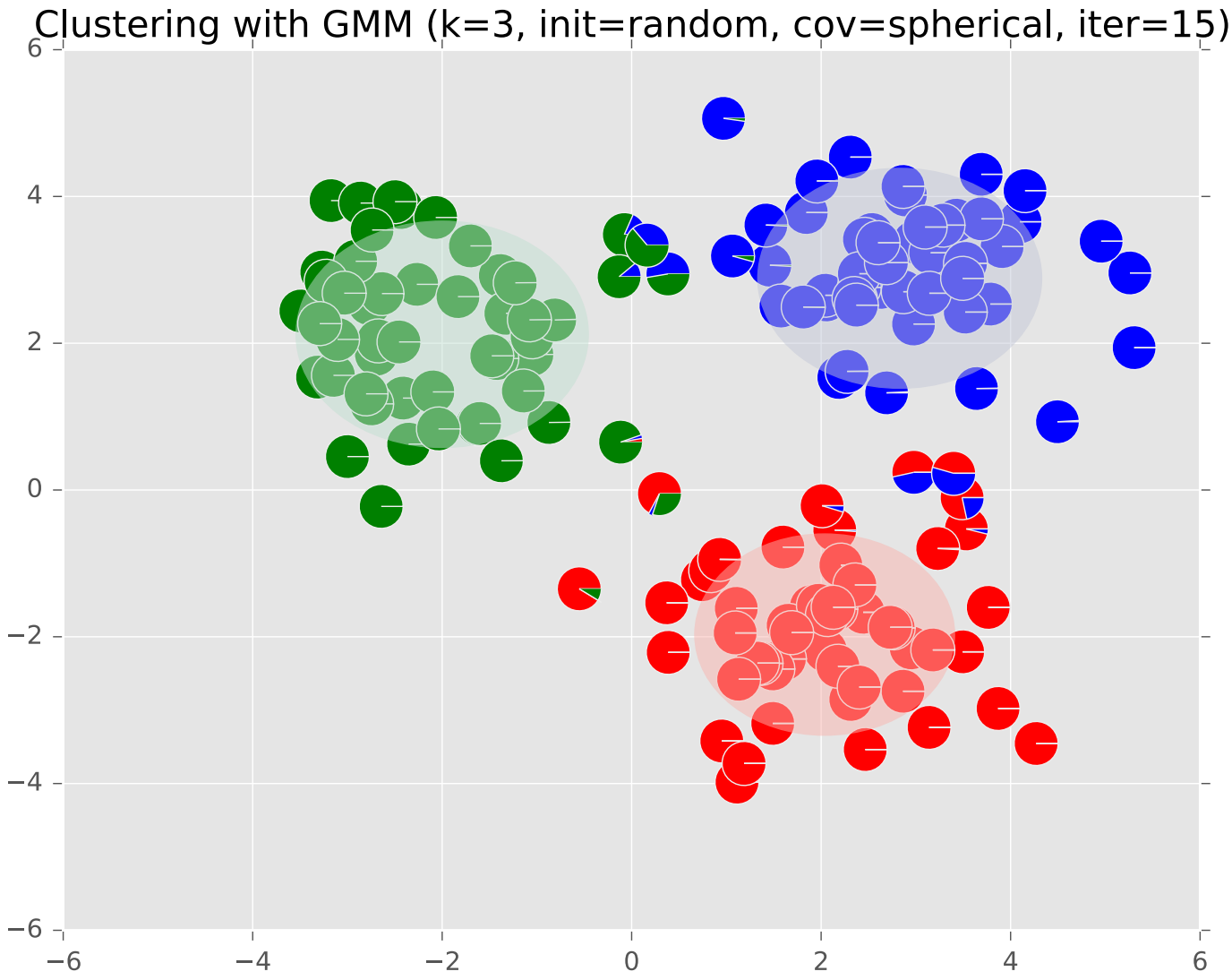
Example: GMM


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

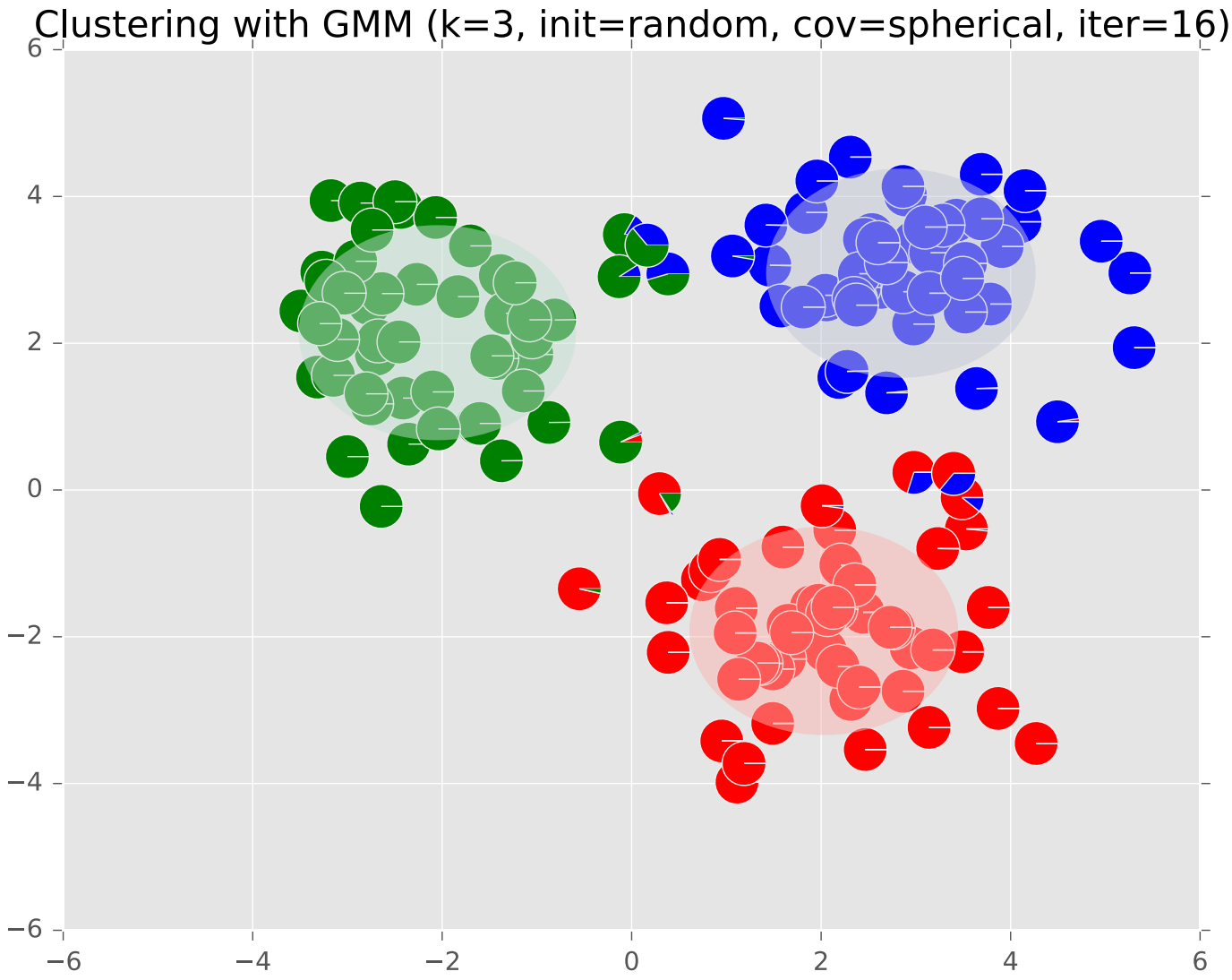
Example: GMM


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

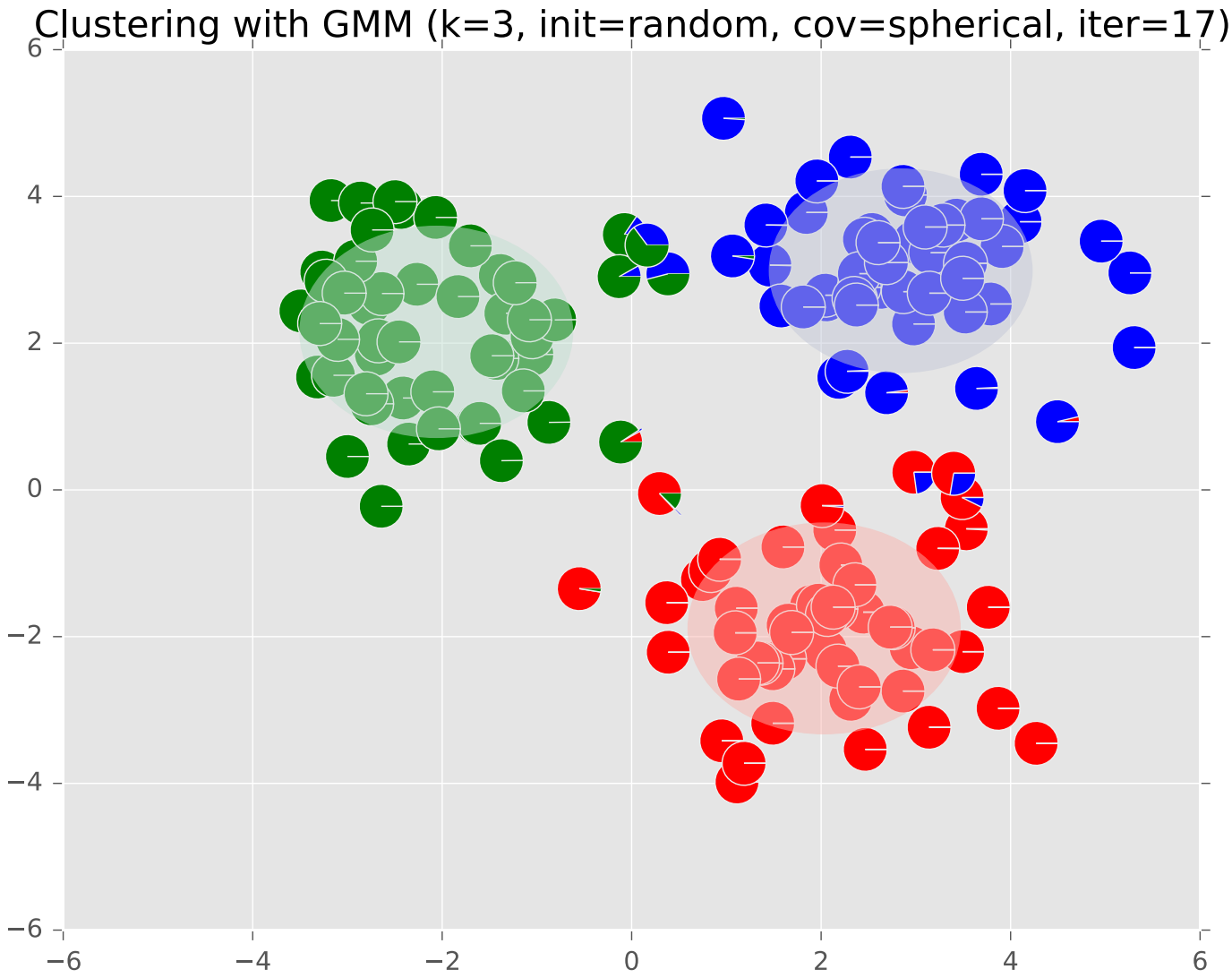
Example: GMM


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

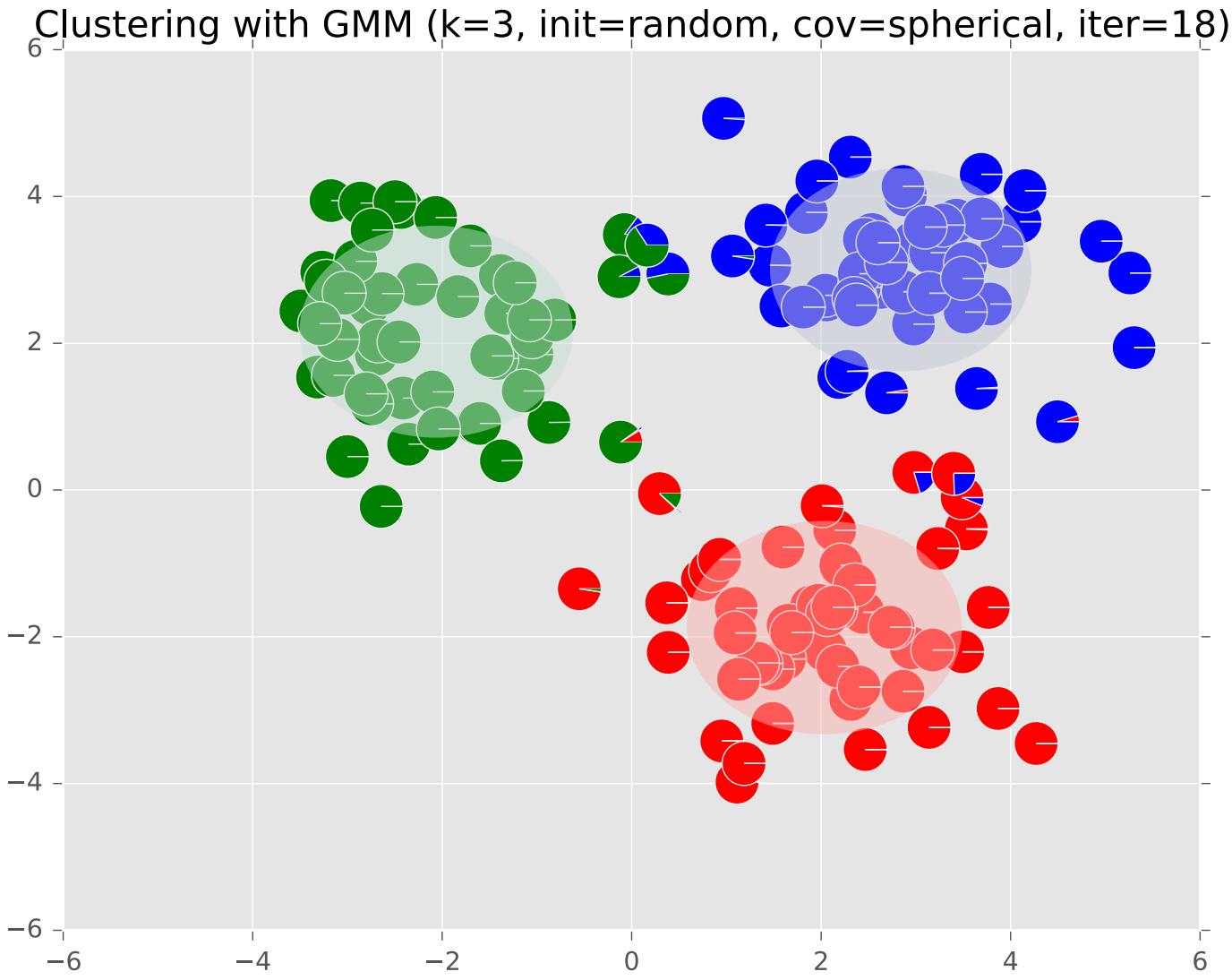
Example: GMM


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

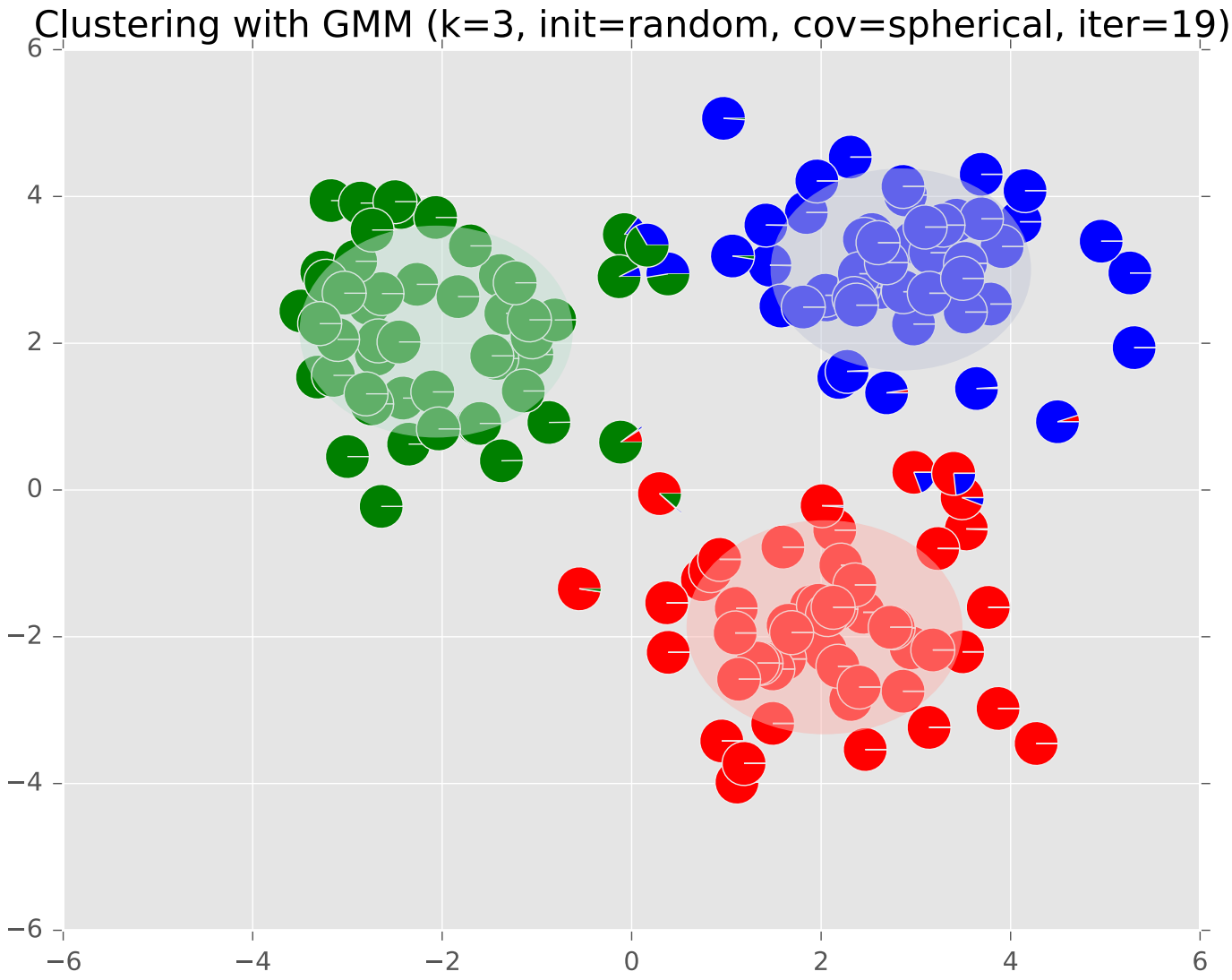
Example: GMM


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 **EM for GMM**

Speed:

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

Initialization:

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

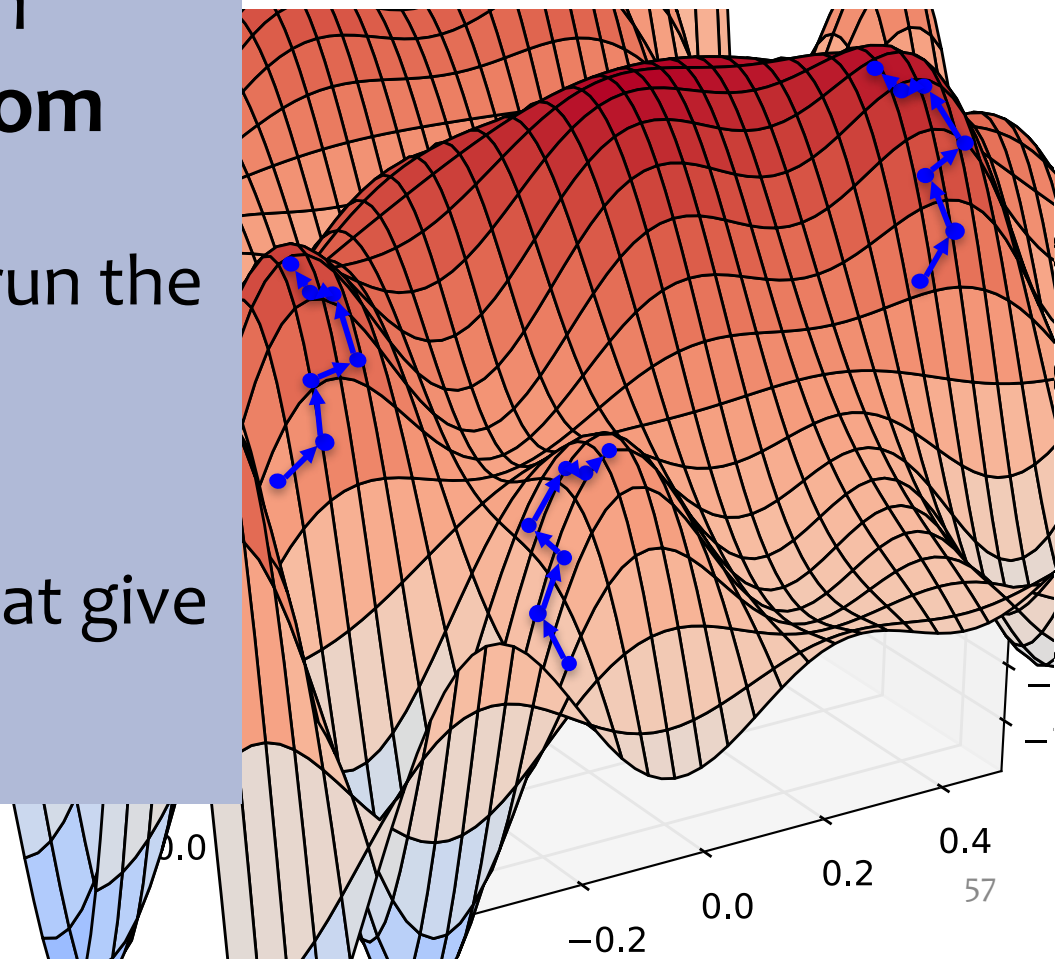
Output:

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

PROPERTIES OF EM

Properties of EM

- EM is *trying* to optimize a **nonconvex** function
- But EM is a **local** optimization algorithm
- Typical solution: **Random Restarts**
  - Just like K-Means, we run the algorithm many times
  - Each time initialize parameters randomly
  - Pick the parameters that give highest likelihood



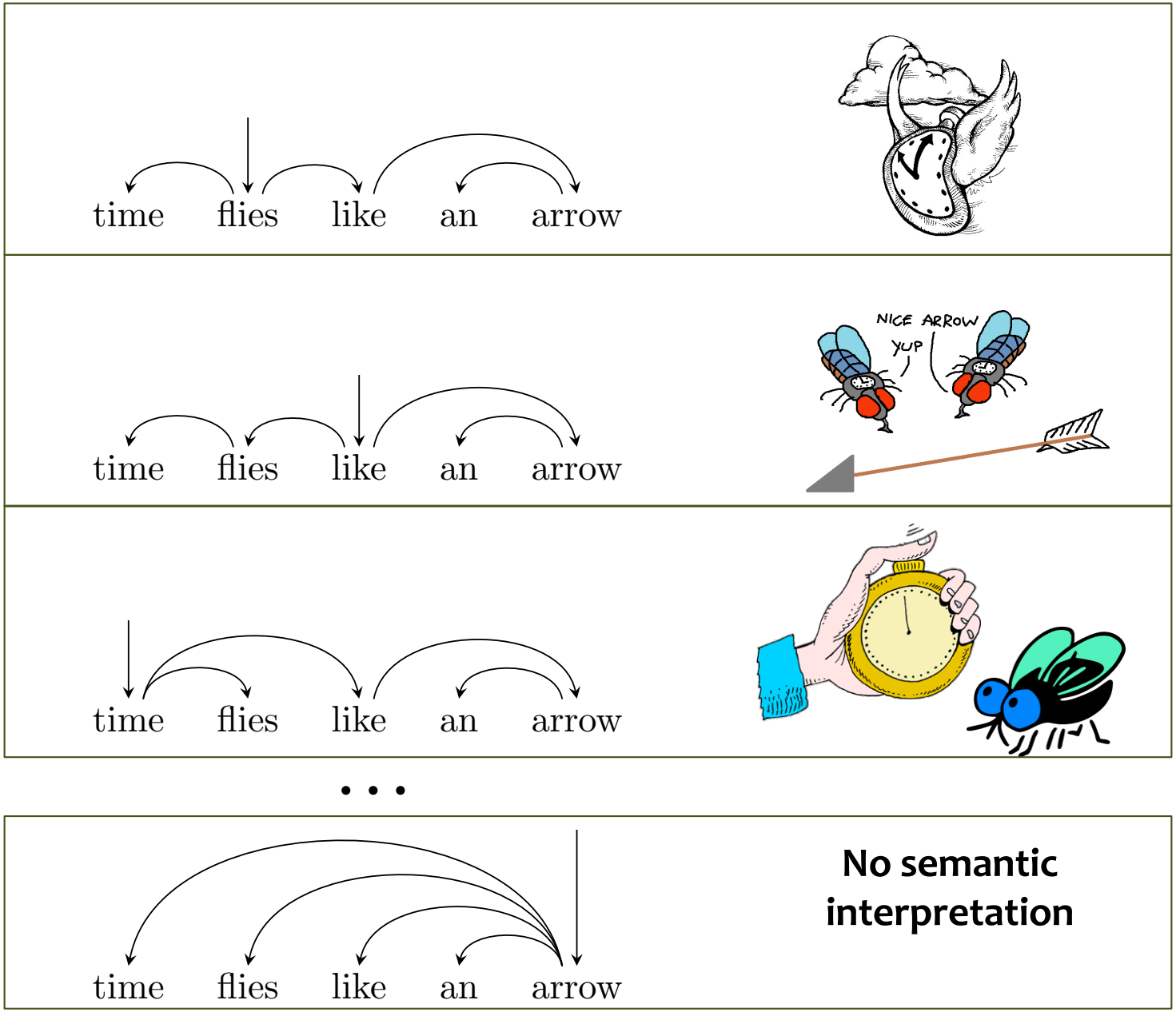
Example: Grammar Induction

Question: Can maximizing (unsupervised) marginal likelihood produce useful results?

Answer: Let's look at an example…

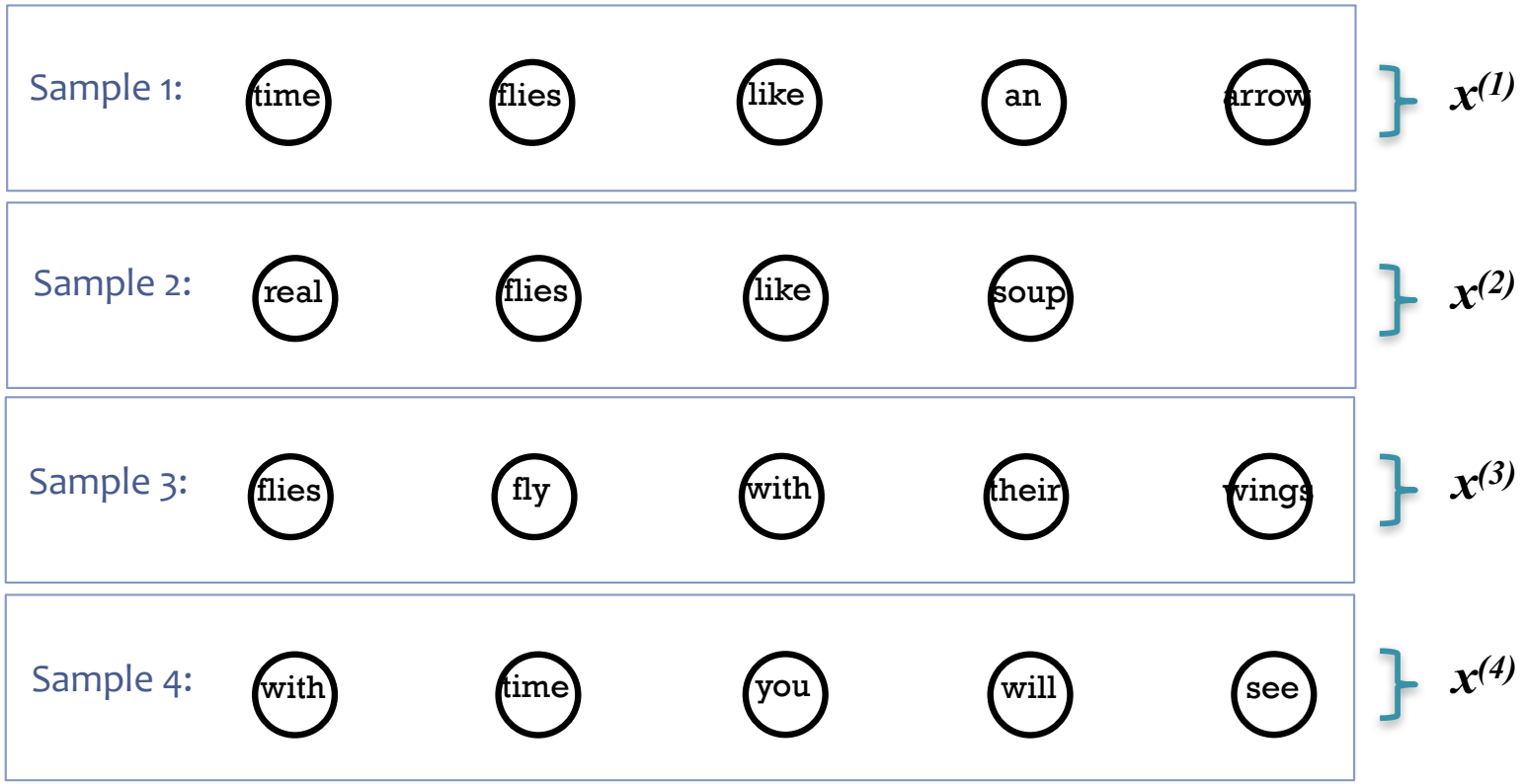
- **Babies** learn the syntax of their **native language** (e.g. English) just by **hearing** many sentences
- Can a **computer** similarly learn syntax of a **human language** just by looking at lots of example sentences?
  - This is the problem of Grammar Induction!
  - It's an unsupervised learning problem
  - We try to recover the **syntactic structure** for each sentence without any supervision

Example: Grammar Induction



Example: Grammar Induction

Training Data: Sentences only, without parses



Test Data: Sentences **with** parses, so we can evaluate accuracy

Example: Grammar Induction

Q: Does likelihood correlate with accuracy on a task we care about?

A: Yes, but there is still a wide range of accuracies for a particular likelihood value

Dependency Model with Valence (Klein & Manning, 2004)

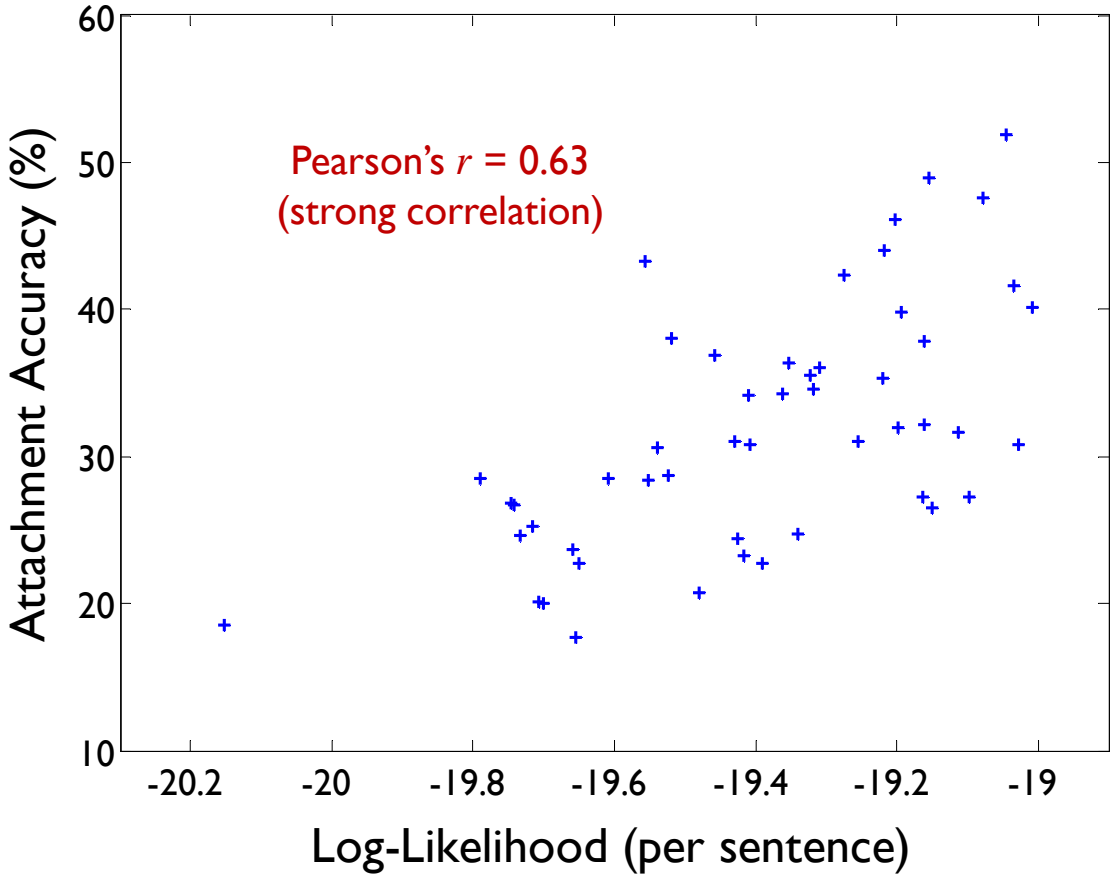


Figure from Gimpel & Smith (NAACL 2012) - slides

Variants of EM

- **Generalized EM**: Replace the M-Step by a single gradient-step that improves the likelihood
- **Monte Carlo EM**: Approximate the E-Step by sampling
- **Sparse EM:** Keep an "active list" of points (updated occasionally) from which we estimate the expected counts in the E-Step
- **Incremental EM / Stepwise EM**: If standard EM is described as a *batch* algorithm, these are the *online* equivalent
- **etc.**

A Report Card for EM

- Some good things about EM:
  - no learning rate (step-size) parameter
  - automatically enforces parameter constraints
  - very fast for low dimensions
  - each iteration guaranteed to improve likelihood

- Some bad things about EM:
  - can get stuck in local minima
  - can be slower than conjugate gradient (especially near convergence)
  - requires expensive inference step
  - is a maximum likelihood/MAP method

DIMENSIONALITY REDUCTION

PCA Outline

- **Dimensionality Reduction**
  - High-dimensional data
  - Learning (low dimensional) representations
- **Principal Component Analysis (PCA)**
  - Examples: 2D and 3D
  - Data for PCA
  - PCA Definition
  - Objective functions for PCA
  - PCA, Eigenvectors, and Eigenvalues
  - Algorithms for finding Eigenvectors / Eigenvalues
- **PCA Examples**
  - Face Recognition
  - Image Compression

Big & High-Dimensional Data

- High-Dimensions = Lot of Features

Document classification

Features per document =

thousands of words/unigrams

millions of bigrams, contextual information

Surveys - Netflix

480189 users x 17770 movies

	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6
Tom	5	?	?	1	3	?
George	?	?	3	1	2	5
Susan	4	3	1	?	5	1
Beth	4	3	?	2	4	2

Slide from Nina Balcan

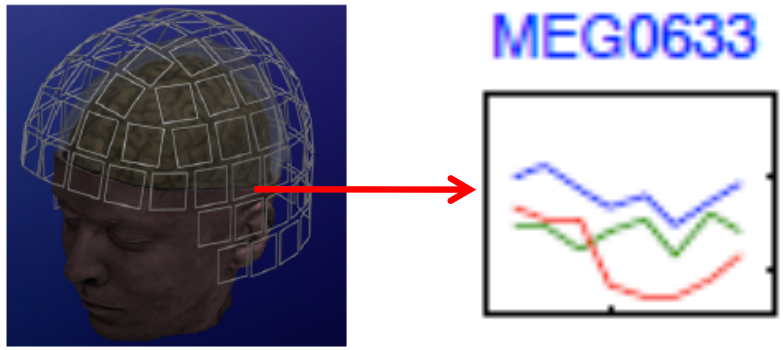
Big & High-Dimensional Data

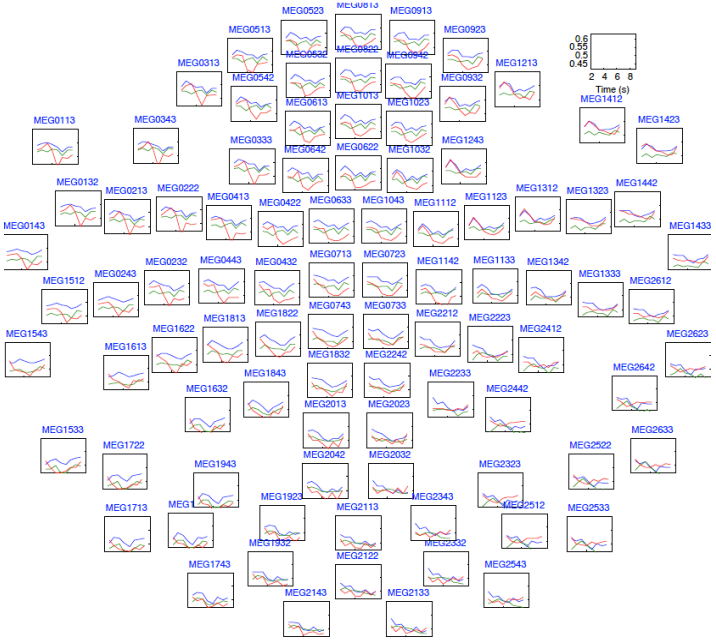
- High-Dimensions = Lot of Features

MEG Brain Imaging

120 locations x 500 time points

x 20 objects





Or any high-dimensional image data

Slide from Nina Balcan

- Big & High-Dimensional Data.

- Useful to learn lower dimensional representations of the data.

Slide from Nina Balcan

Learning Representations

PCA, Kernel PCA, ICA: Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

Useful for:

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

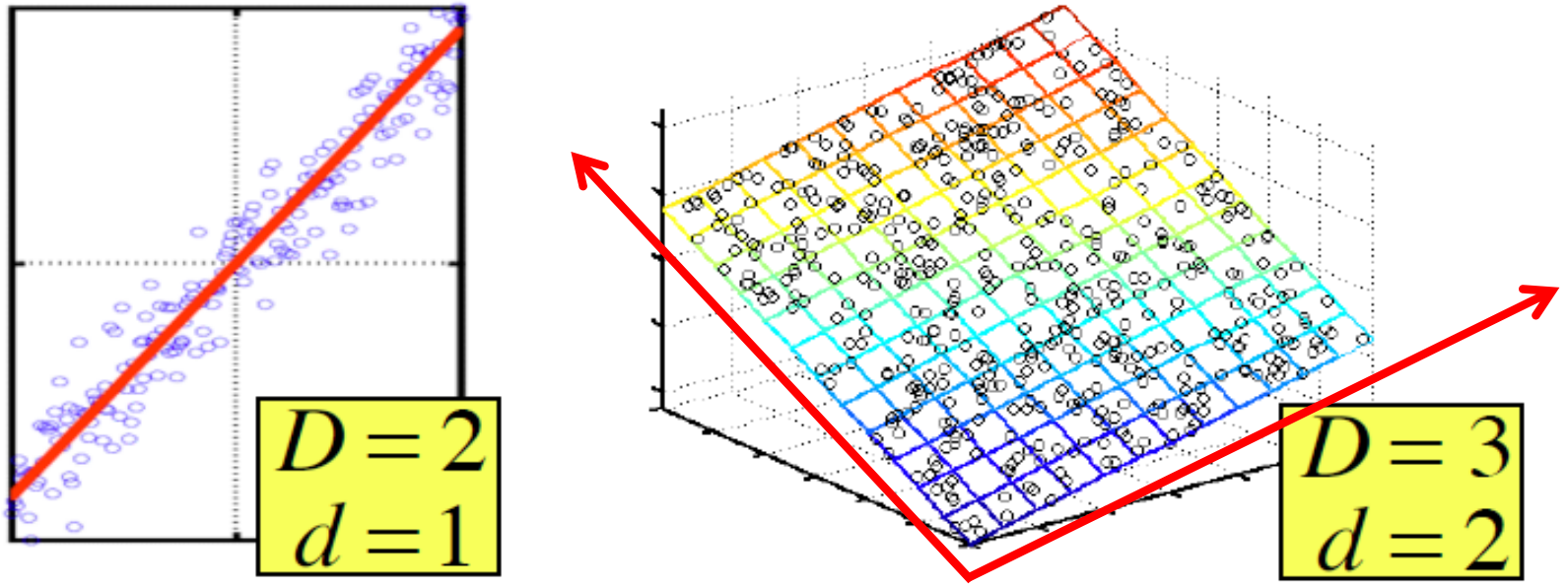
Slide from Nina Balcan

PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA Outline

- **Dimensionality Reduction**
  - High-dimensional data
  - Learning (low dimensional) representations
- **Principal Component Analysis (PCA)**
  - Examples: 2D and 3D
  - Data for PCA
  - PCA Definition
  - Objective functions for PCA
  - PCA, Eigenvectors, and Eigenvalues
  - Algorithms for finding Eigenvectors / Eigenvalues
- **PCA Examples**
  - Face Recognition
  - Image Compression

Principal Component Analysis (PCA)

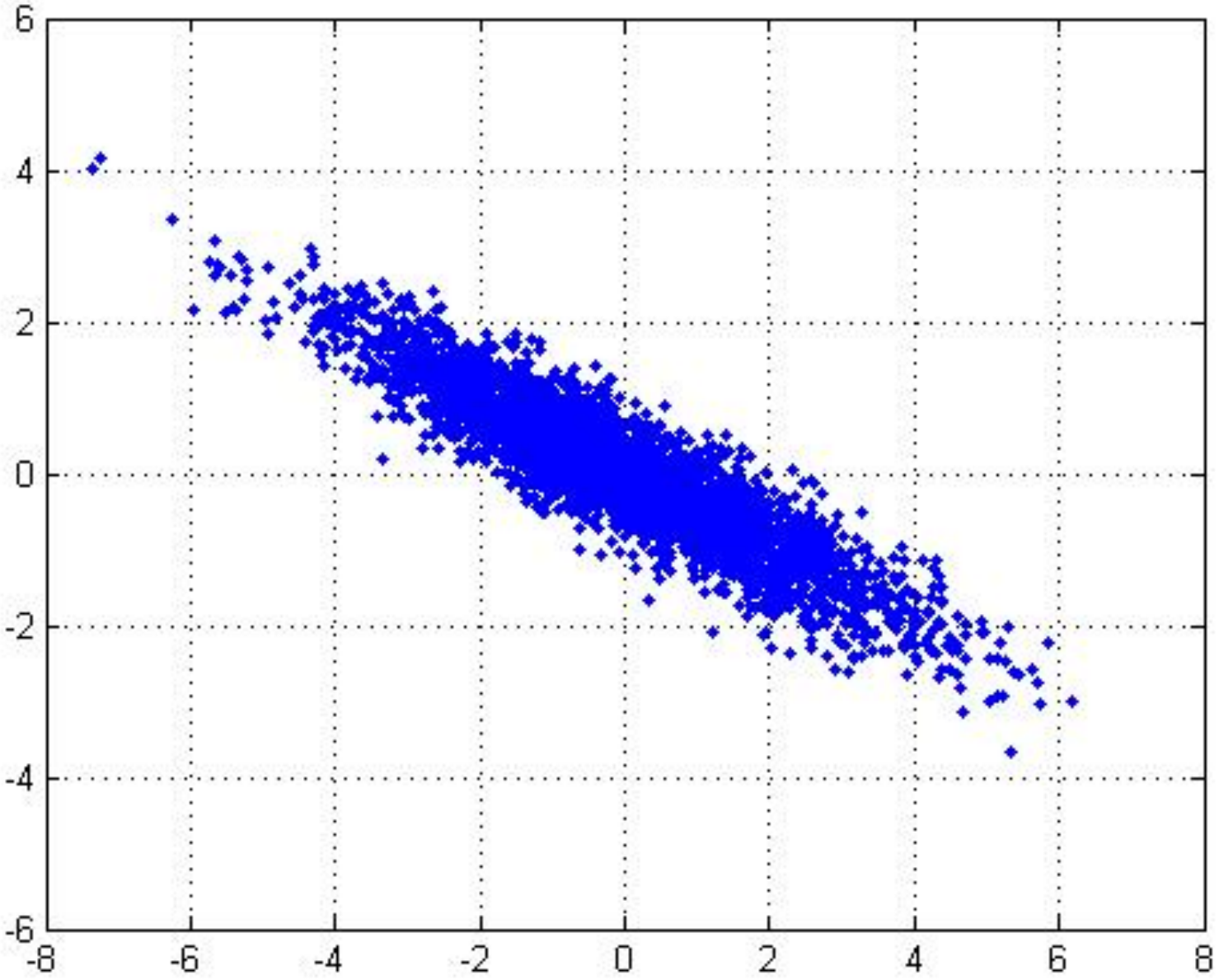


In case where data lies on or near a low d-dimensional linear subspace, axes of this subspace are an effective representation of the data.

Identifying the axes is known as Principal Components Analysis, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).

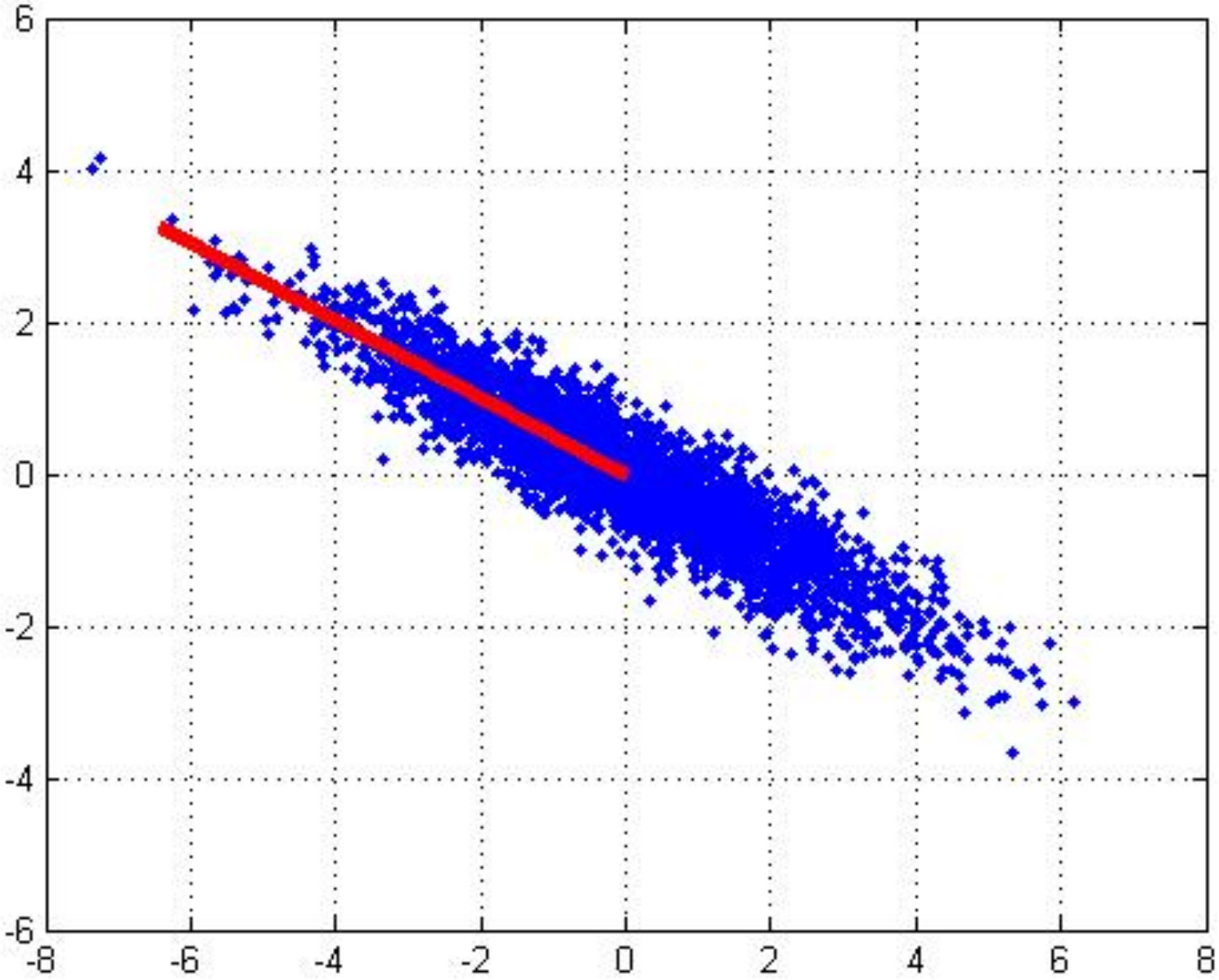
Slide from Nina Balcan

2D Gaussian dataset



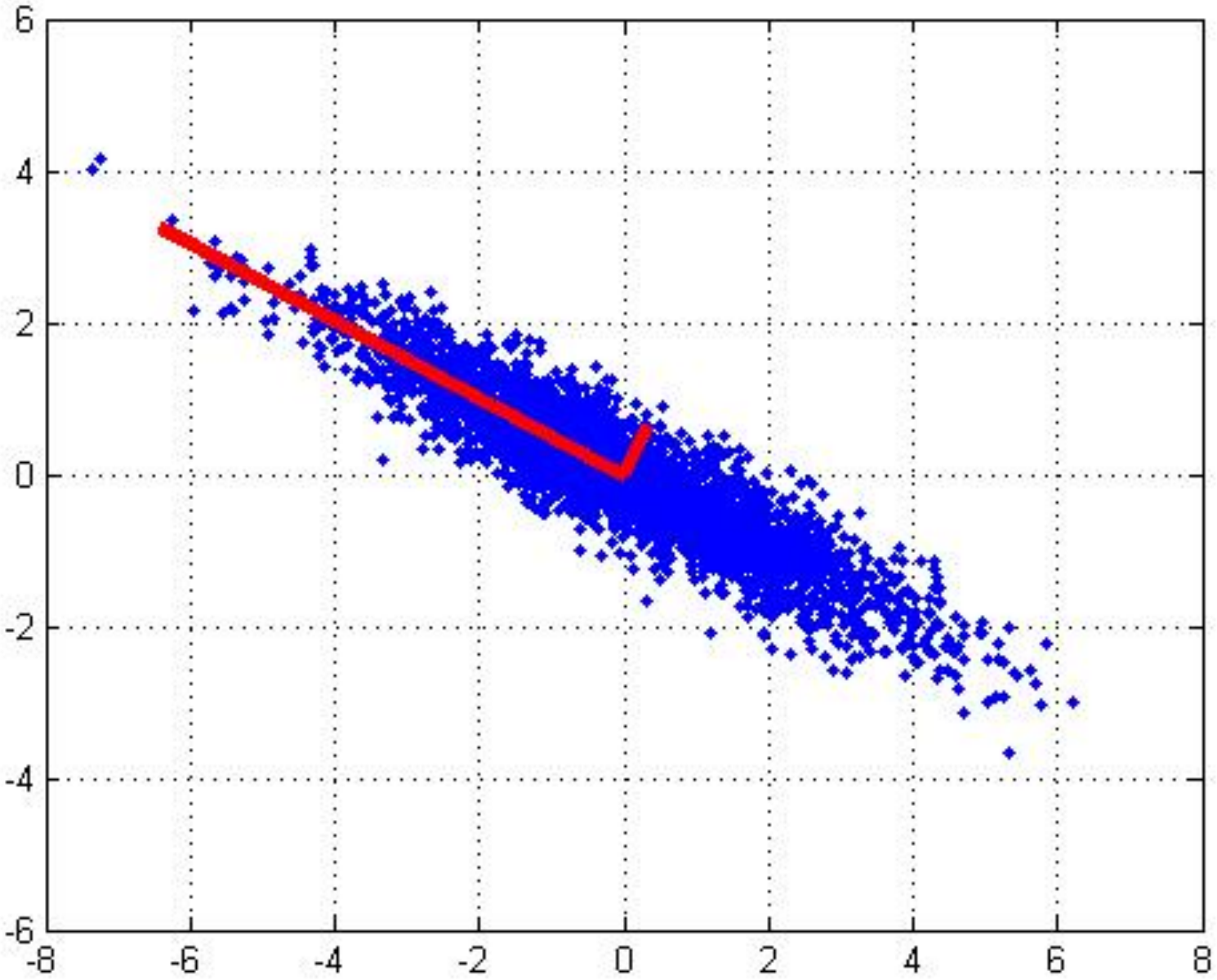
Slide from Barnabas Poczos

1st PCA axis



Slide from Barnabas Poczos

2nd PCA axis



Slide from Barnabas Poczos

Principal Component Analysis (PCA)

Whiteboard

- Data for PCA
- PCA Definition