Machine Learning Department
School of Computer Science
Carnegie Mellon University

Learning Partially Observed Graphical Models + Variational EM

Matt Gormley
Lecture 25
Nov. 20, 2019

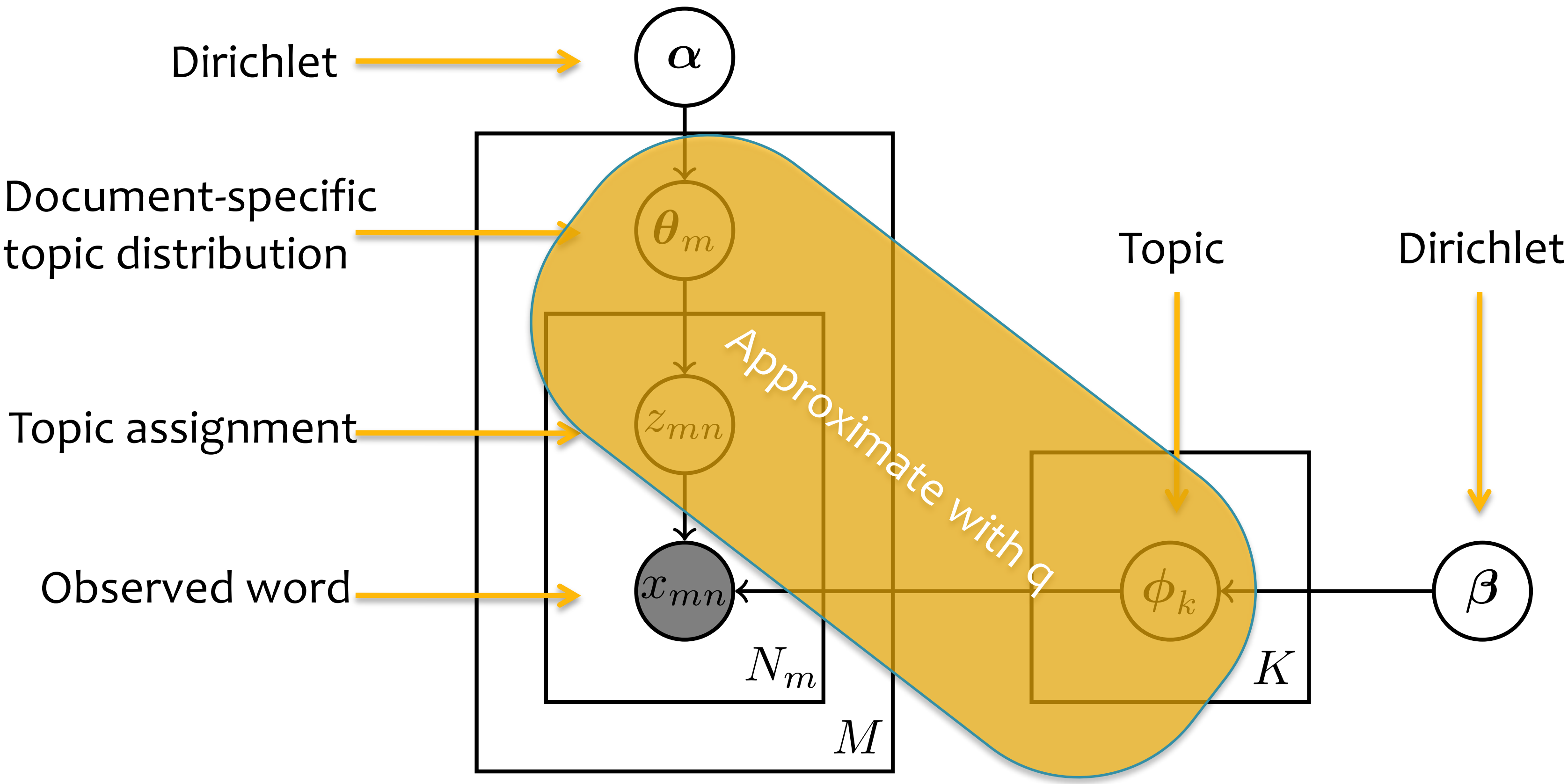
Reminders

- **Homework 4: Topic Modeling**
  - Out: Wed, Nov. 6
  - Due: Mon, Nov. 18 at 11:59pm
- **Homework 5: Variational Inference**
  - Out: Wed, Nov. 20
  - Due: Mon, Dec. 2 at 11:59pm
- **618 Midway Poster:**
  - Submission: Thu, Nov. 21 at 11:59pm
  - Presentation: Fri, Nov. 22 or Mon, Nov. 25

VARIATIONAL INFERENCE RESULTS

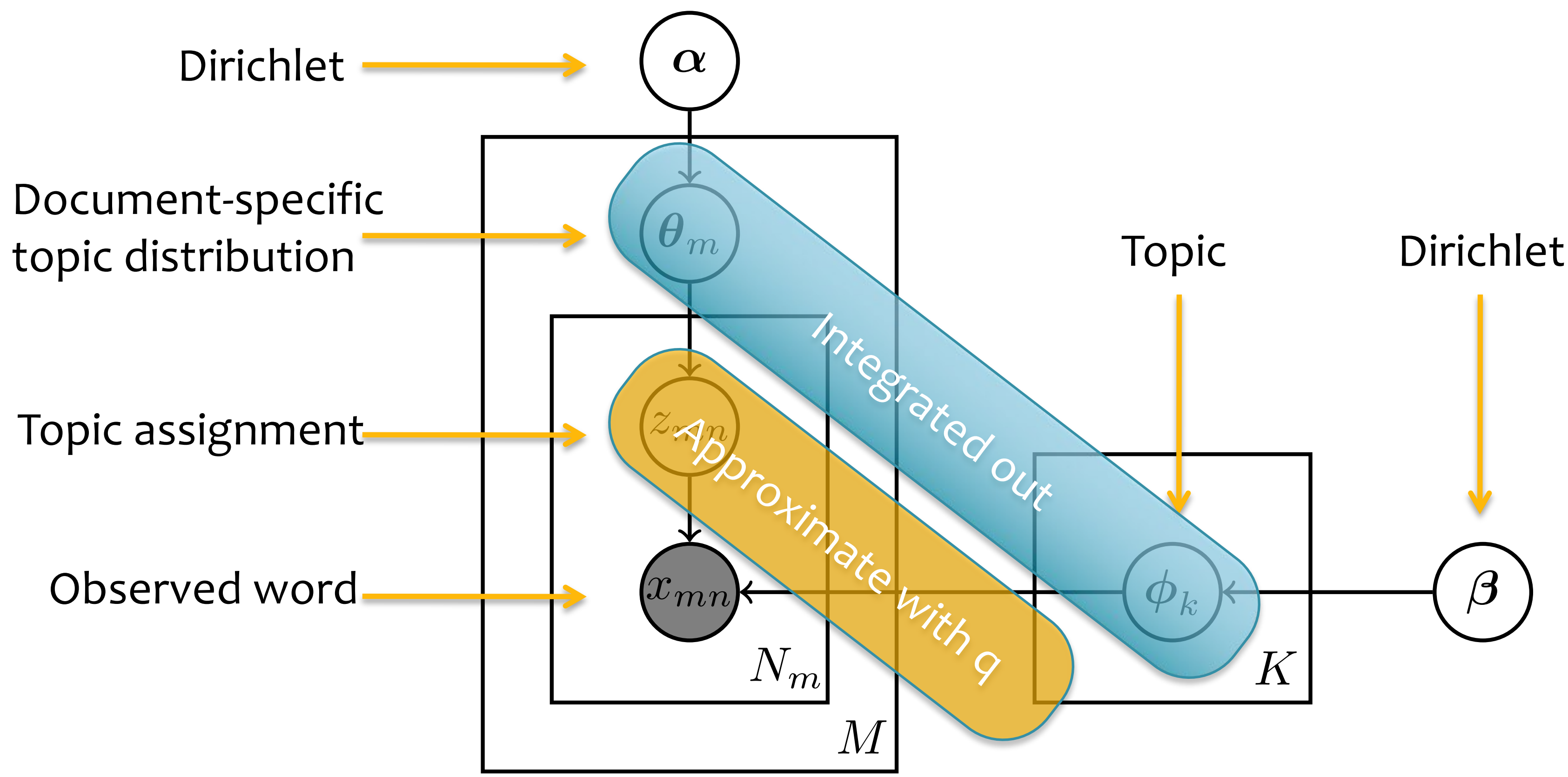
Collapsed Variational Bayesian LDA

- Explicit Variational Inference



Collapsed Variational Bayesian LDA

- Collapsed Variational Inference

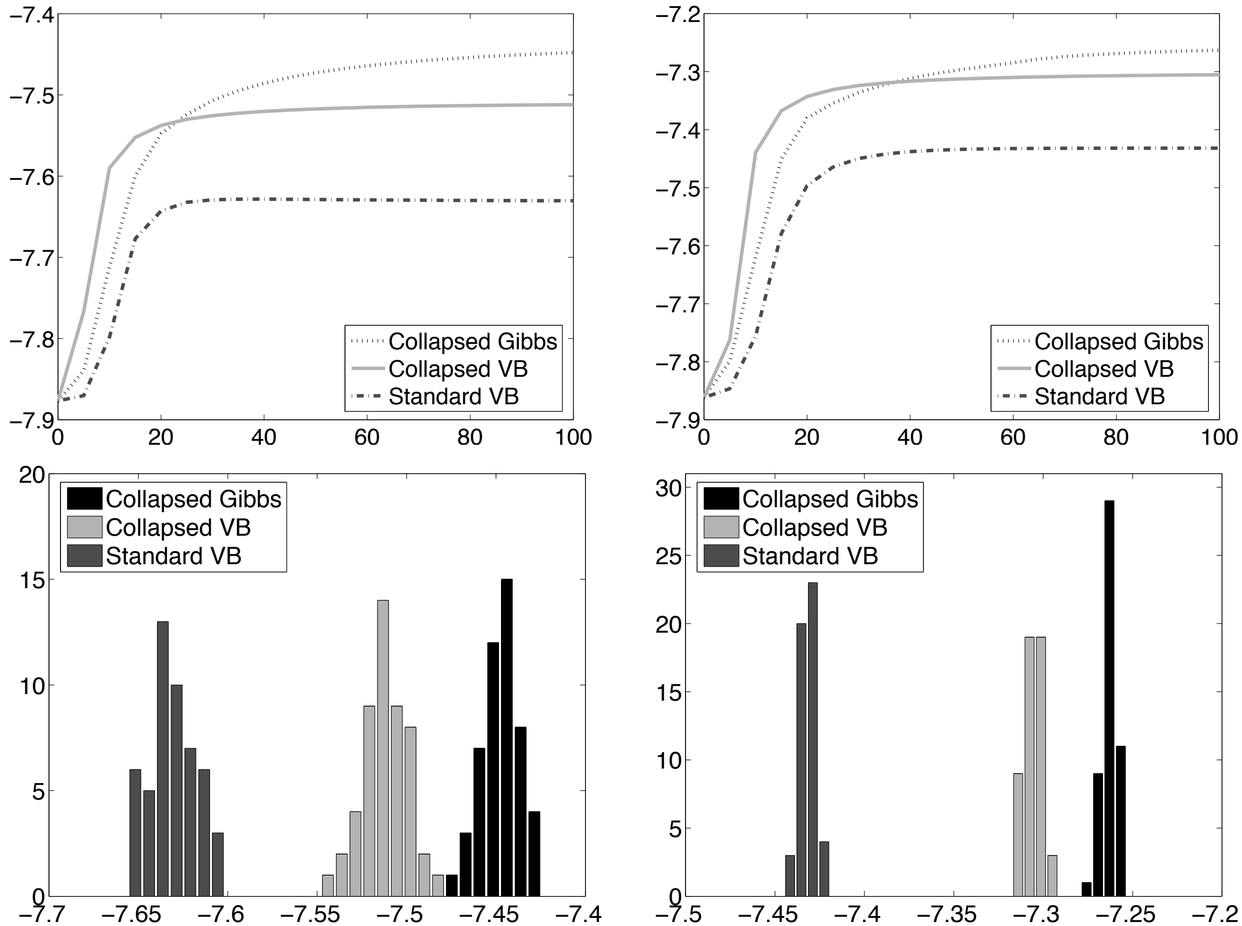


Collapsed Variational Bayesian LDA

- **First row:** test set per word log probabilities as functions of numbers of iterations for VB, CVB and Gibbs.
- **Second row:** histograms of final test set per word log probabilities across 50 random initializations.

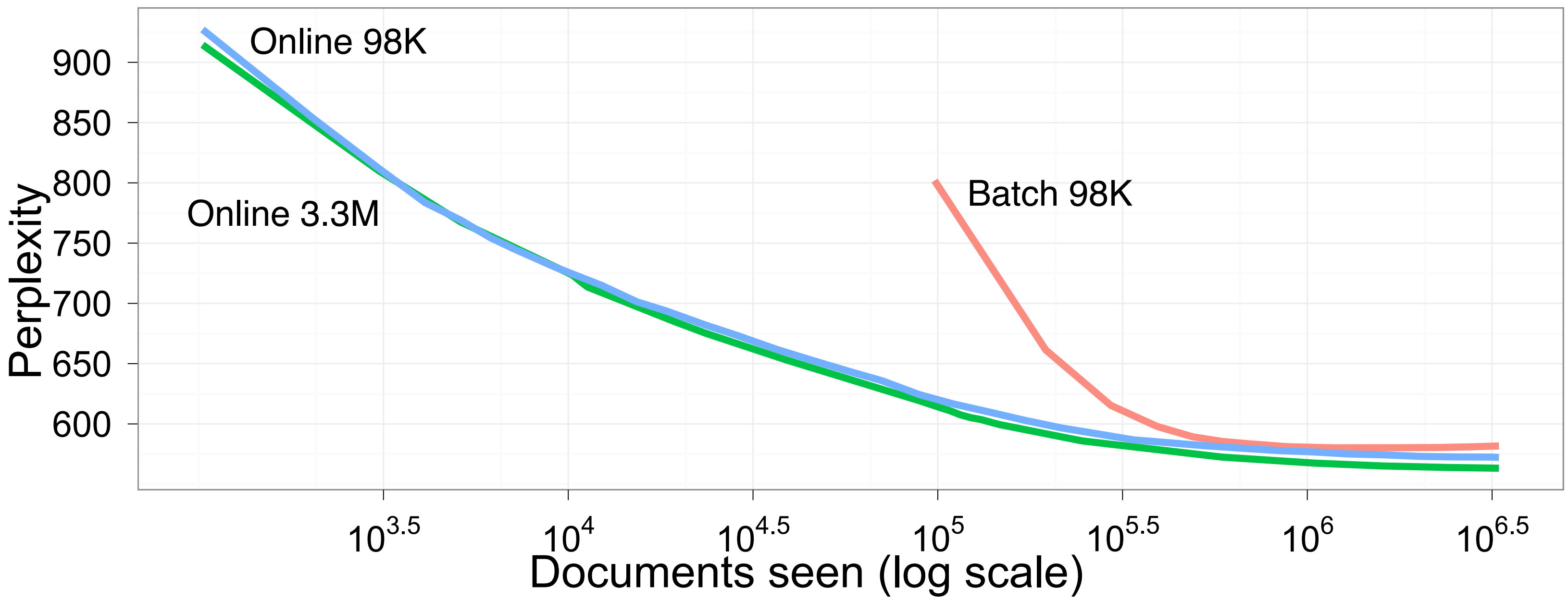
Data from dailykos.com

Data from NeurIPS proceedings



Slide from Teh et al (2007)

Online Variational Bayes for LDA



Documents analyzed	**2048**	**4096**	**8192**	**12288**	**16384**	**32768**	**49152**	**65536**
Top eight words	systems road made service announced national west language	systems health communication service billion language care road	service systems health companies market communication company billion	service systems companies business company billion health industry	service companies systems business company industry market billion	business service companies industry company management systems services	business service companies industry services company management public	business industry service companies services company management public

Figures from Hoffman et al. (2010)

Online Variational Bayes for LDA

Algorithm 1 Batch variational Bayes for LDA

Initialize $\boldsymbol{\lambda}$ randomly.
while relative improvement in $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\phi}, \boldsymbol{\gamma}, \boldsymbol{\lambda}) > 0.00001$ **do**
  E step:
  for $d = 1$ to D **do**
    Initialize $\gamma_{dk} = 1$. (The constant 1 is arbitrary.)
    repeat
      Set $\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log \theta_{dk}] + \mathbb{E}_q[\log \beta_{kw}]\}$
      Set $\gamma_{dk} = \alpha + \sum_w \phi_{dwk} n_{dw}$
    until $\frac{1}{K} \sum_k |\text{change in} \gamma_{dk}| < 0.00001$
  end for
  M step:
  Set $\lambda_{kw} = \eta + \sum_d n_{dw} \phi_{dwk}$
end while

Algorithm 2 Online variational Bayes for LDA

Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$
Initialize $\boldsymbol{\lambda}$ randomly.
for $t = 0$ to ∞ **do**
  E step:
  Initialize $\gamma_{tk} = 1$. (The constant 1 is arbitrary.)
  repeat
    Set $\phi_{twk} \propto \exp\{\mathbb{E}_q[\log \theta_{tk}] + \mathbb{E}_q[\log \beta_{kw}]\}$
    Set $\gamma_{tk} = \alpha + \sum_w \phi_{twk} n_{tw}$
  until $\frac{1}{K} \sum_k |\text{change in} \gamma_{tk}| < 0.00001$
  M step:
  Compute $\tilde{\lambda}_{kw} = \eta + D n_{tw} \phi_{twk}$
  Set $\boldsymbol{\lambda} = (1 - \rho_t)\boldsymbol{\lambda} + \rho_t \tilde{\boldsymbol{\lambda}}$.
end for

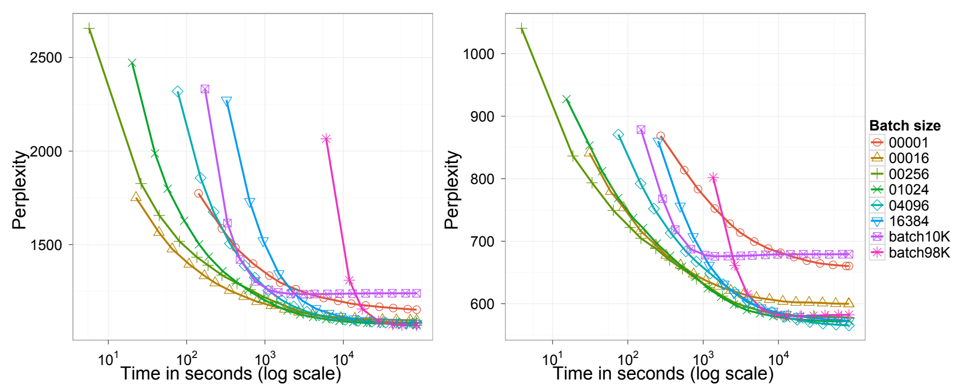


Figure 2: Held-out perplexity obtained on the *Nature* (left) and Wikipedia (right) corpora as a function of CPU time. For moderately large mini-batch sizes, online LDA finds solutions as good as those that the batch LDA finds, but with much less computation. When fit to a 10,000-document subset of the training corpus batch LDA's speed improves, but its performance suffers.

Figures from Hoffman et al. (2010)

Fully–Connected CRF

Model

$$p(\mathbf{x}|\mathbf{i}) = \frac{1}{Z(\mathbf{i})} \exp(-E(\mathbf{x}))$$

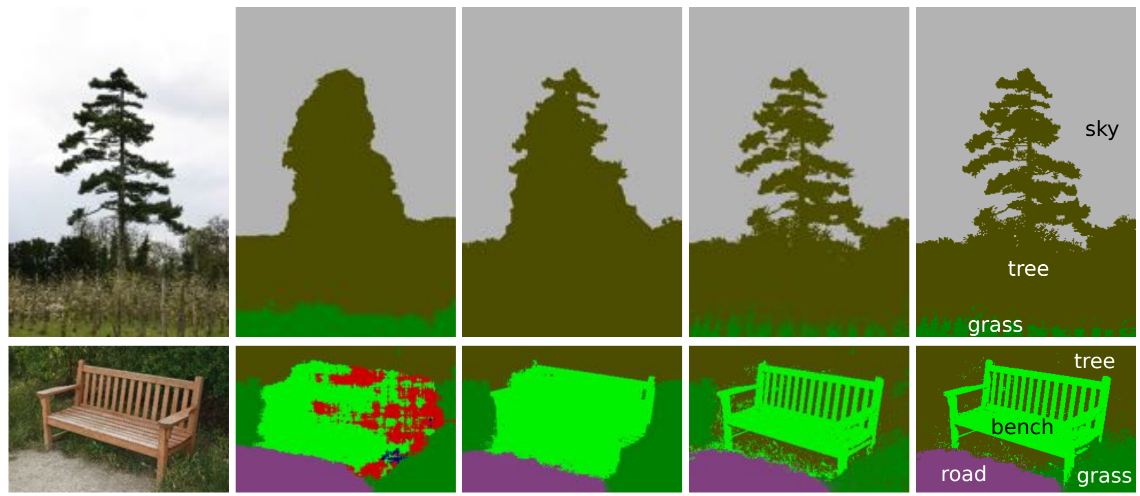
$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

This is a fully connected graph!

Inference

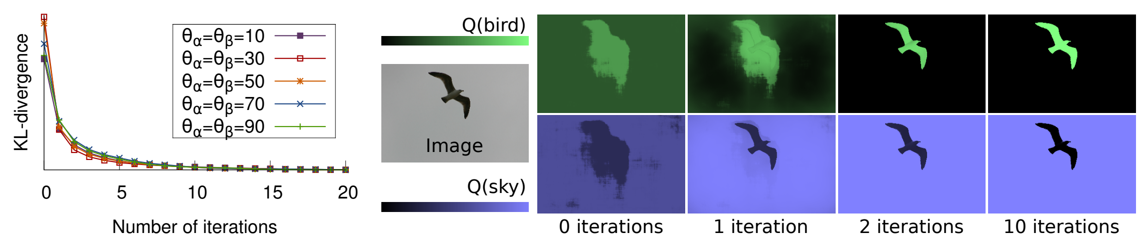
- Can do MCMC, but slow
- Instead use Variational Inference
- Then filter some variables for speed up

Results



(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF, MCMC inference, 36 hrs (e) Fully connected CRF, our approach, 0.2 seconds

Figure 1: Pixel-level classification with a fully connected CRF. (a) Input image from the MSRC-21 dataset. (b) The response of unary classifiers used by our models. (c) Classification produced by the Robust P^n CRF [9]. (d) Classification produced by MCMC inference [17] in a fully connected pixel-level CRF model; the algorithm was run for 36 hours and only partially converged for the bottom image. (e) Classification produced by our inference algorithm in the fully connected model in 0.2 seconds.



(a) KL-divergence (b) Distributions $Q(X_i = \text{“bird”})$ (top) and $Q(X_i = \text{“sky”})$ (bottom)

Figure 2: Convergence analysis. (a) KL-divergence of the mean field approximation during successive iterations of the inference algorithm, averaged across 94 images from the MSRC-21 dataset. (b) Visualization of convergence on distributions for two class labels over an image from the dataset.

Figures from Krähenbühl & Koltun (2011)

Fully–Connected CRF

Model

$$p(\mathbf{x}|\mathbf{i}) = \frac{1}{Z(\mathbf{i})} \exp(-E(\mathbf{x}))$$

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

This is a fully connected graph!

Inference

- Can do MCMC, but slow
- Instead use Variational Inference
- Then filter some variables for speed up

Figures from Krähenbühl & Koltun (2011)

Follow-up Work (combine with CNN)

Published as a conference paper at ICLR 2015

SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS

Liang-Chieh Chen
Univ. of California, Los Angeles
lcchen@cs.ucla.edu

George Papandreou *
Google Inc.
gpapan@google.com

Iasonas Kokkinos
CentraleSupélec and INRIA
iasonas.kokkinos@ecp.fr

Kevin Murphy
Google Inc.
kpmurphy@google.com

Alan L. Yuille
Univ. of California, Los Angeles
yuille@stat.ucla.edu

ABSTRACT

Deep Convolutional Neural Networks (DCNNs) have recently shown state of the art performance in high level vision tasks, such as image classification and object detection. This work brings together methods from DCNNs and probabilistic graphical models for addressing the task of pixel-level classification (also called "semantic image segmentation"). We show that responses at the final layer of DCNNs are not sufficiently localized for accurate object segmentation. This is due to the very invariance properties that make DCNNs good for high level tasks. We overcome this poor localization property of deep networks by combining the responses at the final DCNN layer with a fully connected Conditional Random Field (CRF). Qualitatively, our "DeepLab" system is able to localize segment boundaries at a level of accuracy which is beyond previous methods. Quantitatively, our method sets the new state-of-art at the PASCAL VOC-2012 semantic image segmentation task, reaching 71.6% IOU accuracy in the test set. We show how these results can be obtained efficiently: Careful network re-purposing and a novel application of the 'hole' algorithm from the wavelet community allow dense computation of neural net responses at 8 frames per second on a modern GPU.

Joint Parsing and Alignment with Weakly Synchronized Grammars

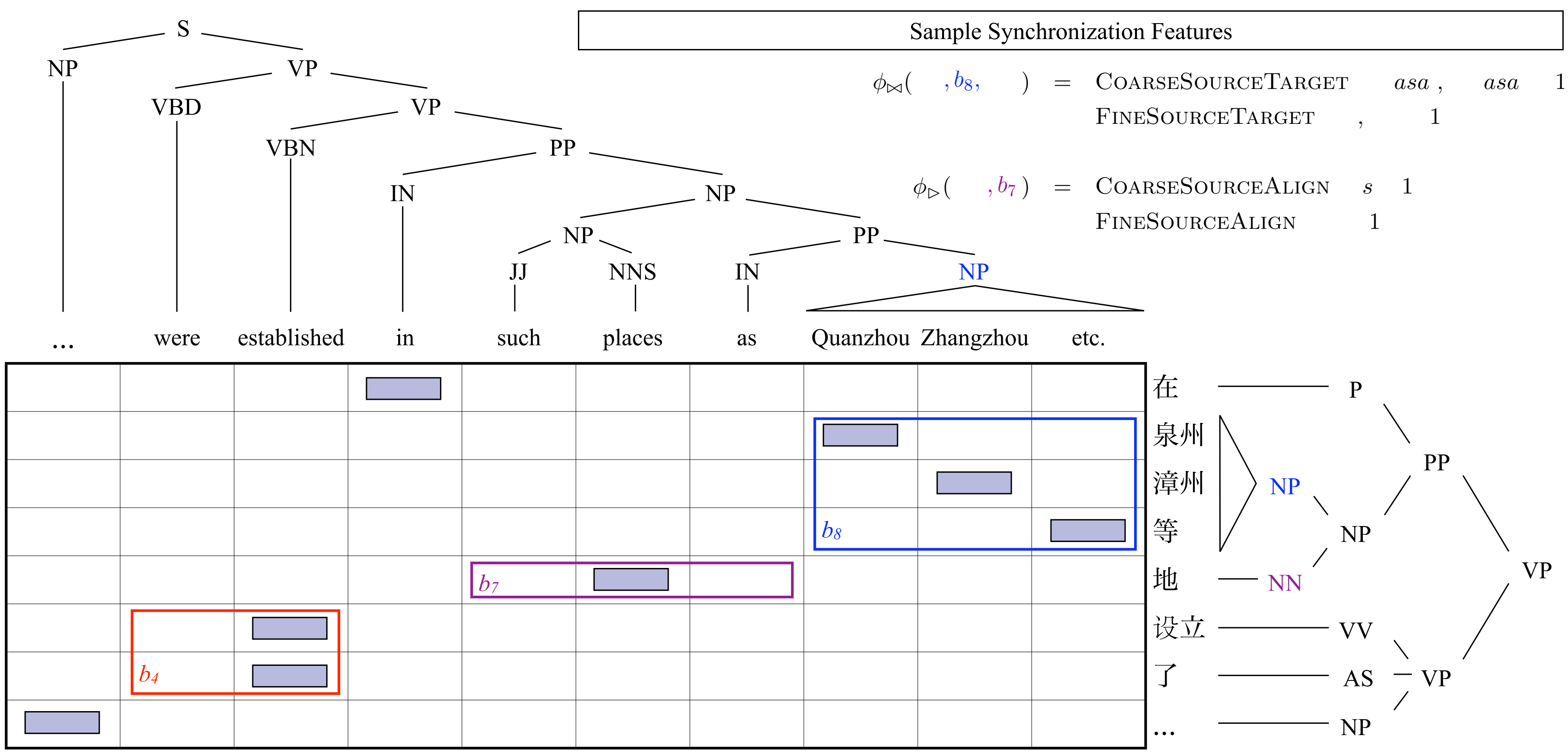
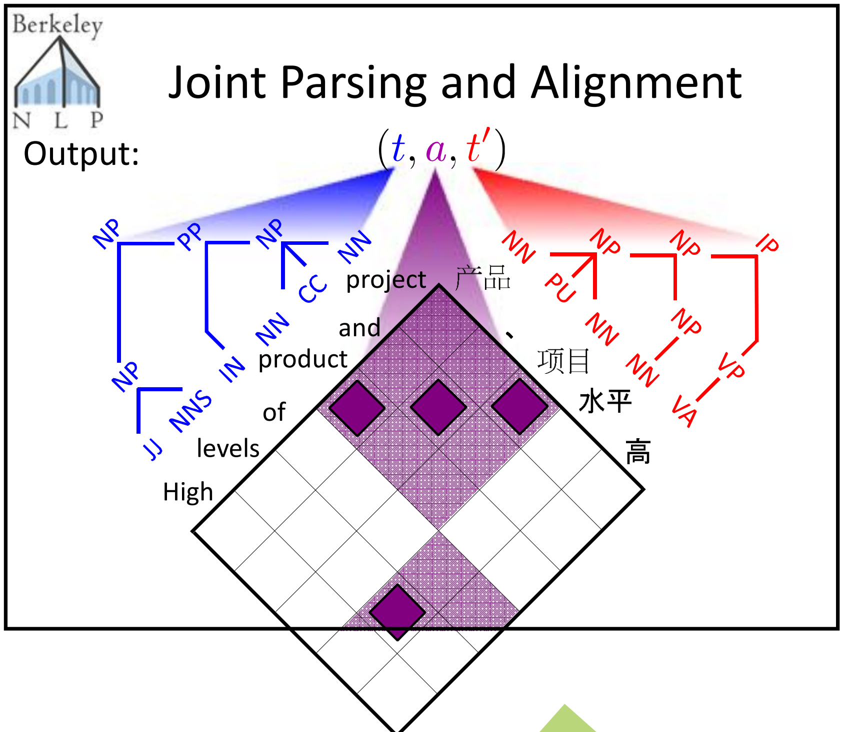


Figure 2: An example of a Chinese-English sentence pair with parses, word alignments, and a subset of the full optimal ITG derivation, including one totally unsynchronized bispan (b_4), one partially synchronized bispan (b_7), and and fully synchronized bispan (b_8). The inset provides some examples of active synchronization features (see Section 4.3) on these bispans. On this example, the monolingual English parser erroneously attached the lower PP to the VP headed by *established*, and the non-syntactic ITG word aligner misaligned 等 to *such* instead of to *etc.* Our joint model corrected both of these mistakes because it was rewarded for the synchronization of the two NPs joined by b_8.

Figures from Burkett et al. (2010)

Joint Parsing and Alignment with Weakly Synchronized Grammars

Figures from Burkett & Klein (ACL 2013 tutorial)



This uses
Structured
Mean Field V.I.

	Test Results		
	Ch F_1	Eng F_1	Tot F_1
Monolingual	83.6	81.2	82.5
Reranker	**86.0**	83.8	84.9
Joint	85.7	**84.5**	**85.1**

Table 1: Parsing results. Our joint model has the highest reported F_1 for English-Chinese bilingual parsing.

	Test Results			
	Precision	Recall	AER	F_1
HMM	86.0	58.4	30.0	69.5
ITG	**86.8**	73.4	20.2	79.5
Joint	85.5	**84.6**	**14.9**	**85.0**

Table 2: Word alignment results. Our joint model has the highest reported F_1 for English-Chinese word alignment.

Figures from Burkett et al. (2010)

HIDDEN STATE CRFS

Case Study: Object Recognition

Data consists of images $\boldsymbol{x}$ and labels y.

pigeon

rhinoceros

leopard

llama

Case Study: Object Recognition

Data consists of images x and labels y.

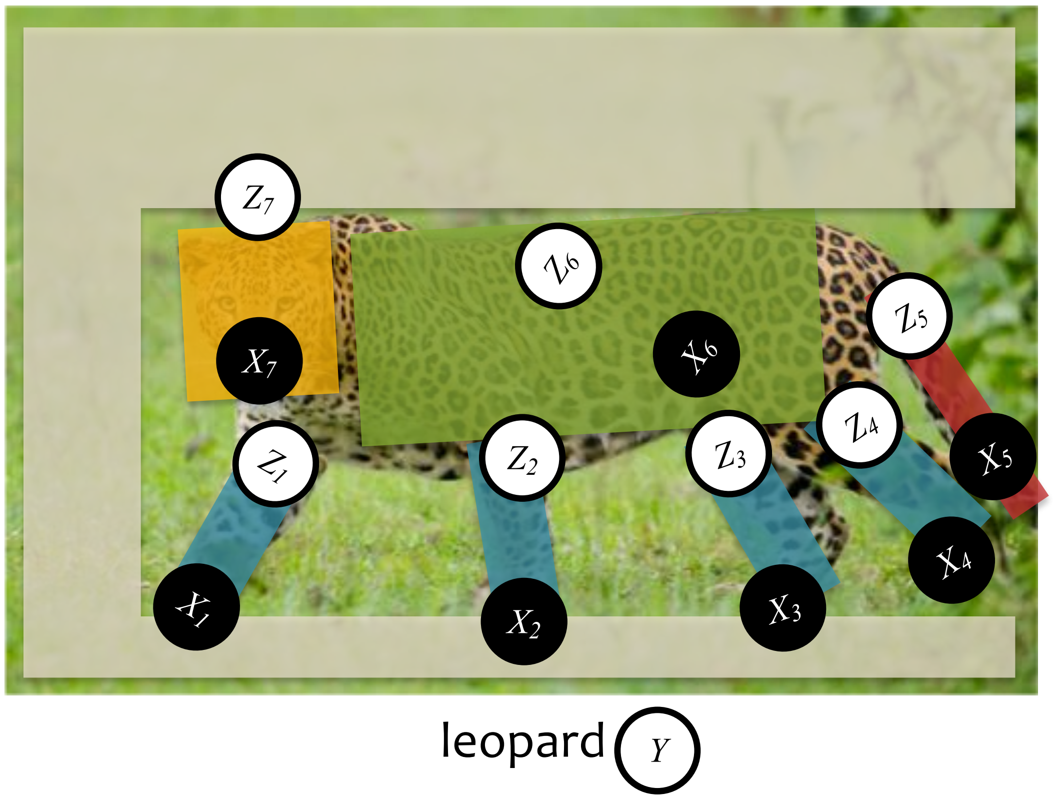
- Preprocess data into "patches"
- Posit a latent labeling z describing the object's parts (e.g. head, leg, tail, torso, grass)
- Define graphical model with these latent variables in mind
- z is not observed at train or test time

leopard

Case Study: Object Recognition

Data consists of images x and labels y.

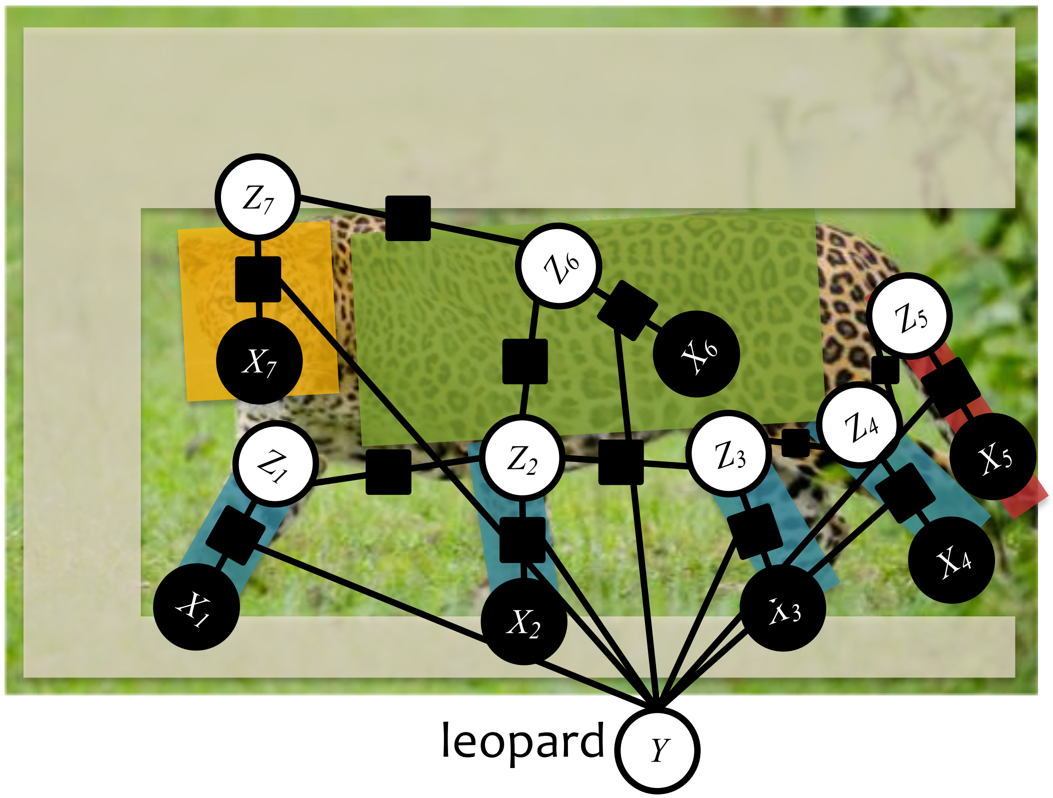
- Preprocess data into "patches"
- Posit a latent labeling z describing the object's parts (e.g. head, leg, tail, torso, grass)
- Define graphical model with these latent variables in mind
- z is not observed at train or test time



Case Study: Object Recognition

Data consists of images $\boldsymbol{x}$ and labels y.

- Preprocess data into "patches"
- Posit a latent labeling z describing the object's parts (e.g. head, leg, tail, torso, grass)
- Define graphical model with these latent variables in mind
- z is not observed at train or test time

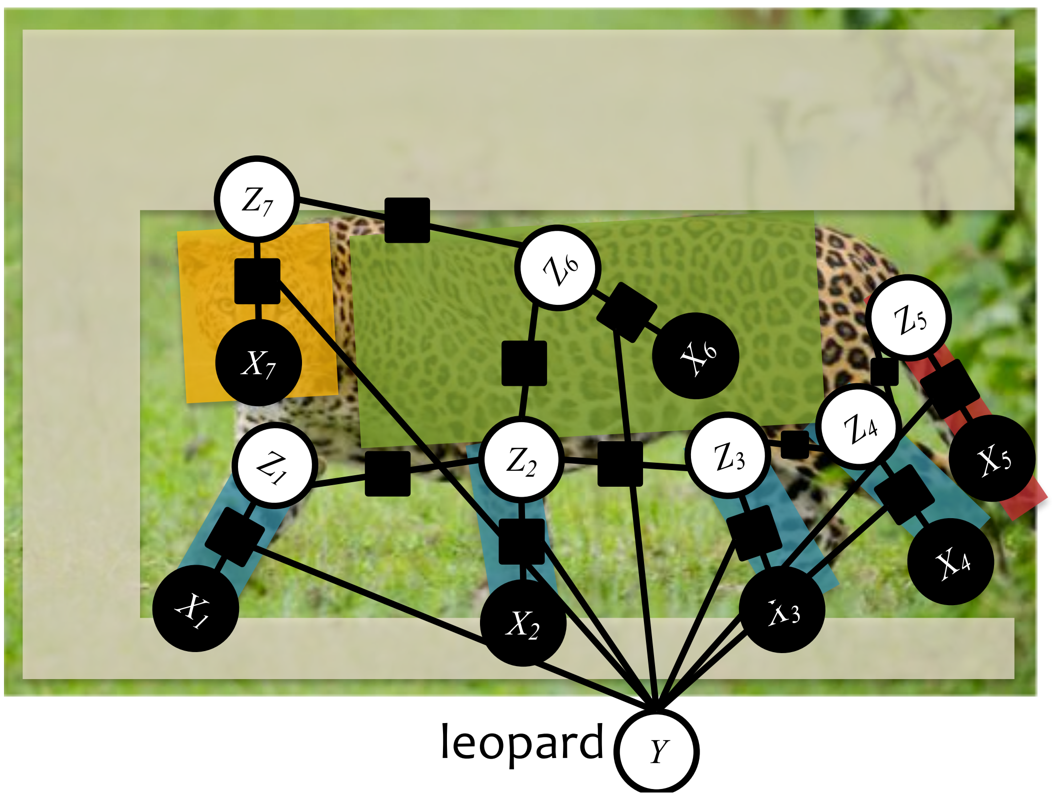


Hidden-state CRFs

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Joint model: $p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\theta})} \prod_{\alpha} \psi_\alpha(\boldsymbol{y}_\alpha, \boldsymbol{z}_\alpha, \boldsymbol{x})$

Marginalized model: $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x})$



Hidden-state CRFs

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Joint model: $p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\theta})} \prod_{\alpha} \psi_{\alpha}(\boldsymbol{y}_{\alpha}, \boldsymbol{z}_{\alpha}, \boldsymbol{x})$

Marginalized model: $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x})$

We can train using gradient based methods:

(the values $\boldsymbol{x}$ are omitted below for clarity)

$$\frac{d\ell(\boldsymbol{\theta}|\mathcal{D})}{d\boldsymbol{\theta}} = \sum_{n=1}^{N} \left(\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{y}^{(n)})}[f_j(\boldsymbol{y}^{(n)}, \boldsymbol{z})] - \mathbb{E}_{\boldsymbol{y},\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\cdot,\cdot)}[f_j(\boldsymbol{y}, \boldsymbol{z})] \right)$$

$$= \sum_{n=1}^{N} \sum_{\alpha} \left(\sum_{\boldsymbol{z}_{\boldsymbol{\alpha}}} \underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{z}_{\boldsymbol{\alpha}} \mid \boldsymbol{y}^{(n)})}_{\text{Inference on clamped factor graph}} f_{\alpha,j}(\boldsymbol{y}_{\alpha}^{(n)}, \boldsymbol{z}_{\alpha}) - \sum_{\boldsymbol{y}_{\boldsymbol{\alpha}}, \boldsymbol{z}_{\boldsymbol{\alpha}}} \underbrace{p_{\boldsymbol{\theta}}(\boldsymbol{y}_{\boldsymbol{\alpha}}, \boldsymbol{z}_{\boldsymbol{\alpha}})}_{\text{Inference on full factor graph}} f_{\alpha,j}(\boldsymbol{y}_{\boldsymbol{\alpha}}, \boldsymbol{z}_{\boldsymbol{\alpha}}) \right)$$

Inference on **clamped** factor graph

Inference on **full** factor graph

GAUSSIAN MIXTURE MODEL (GMM)

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:

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

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:

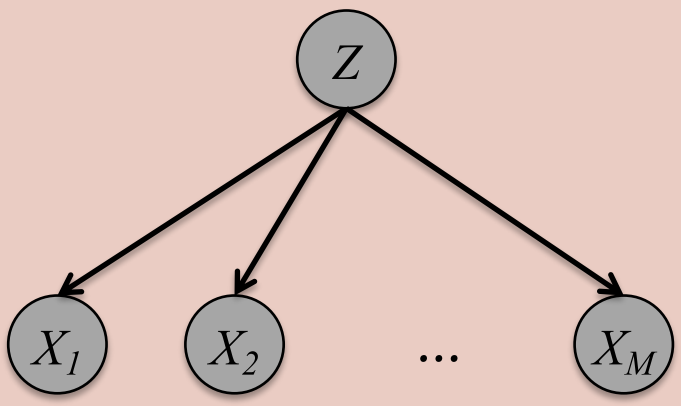
$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)})$$

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

Learning a Mixture Model

Supervised Learning: The parameters decouple!

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



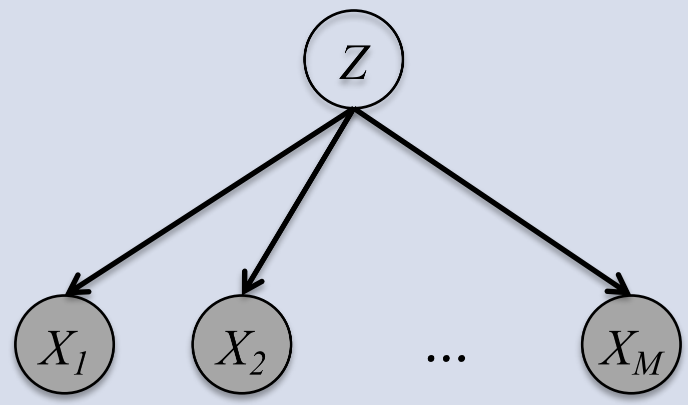
$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta}, \boldsymbol{\phi}}{\text{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} | z^{(i)}) p_{\boldsymbol{\phi}}(z^{(i)})$$

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

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

Unsupervised Learning: Parameters are coupled by marginalization.

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



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

Learning a Mixture Model

Supervised Learning: The parameters decouple!

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

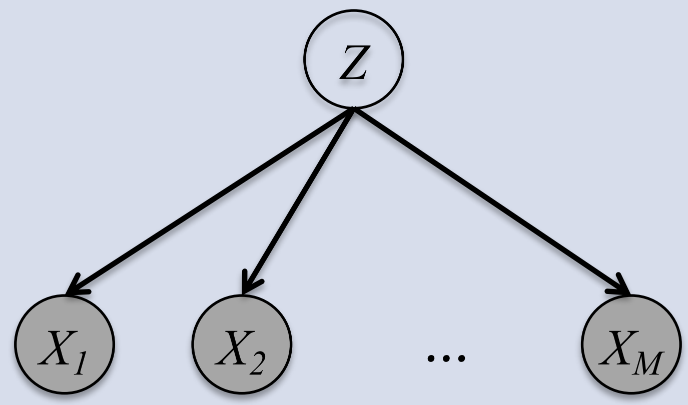
Training certainly isn't as simple as the supervised case.

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.

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



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

EXPECTATION MAXIMIZATION

Hard Expectation-Maximization

- Initialize **parameters** **randomly**
- **while** not converged
  1. **E-Step:**
     Set the **latent variables** to the the values that maximizes likelihood, treating parameters as observed
  2. **M-Step:**
     Set the **parameters** to the values that maximizes likelihood, treating latent variables as observed

Estimate unobserved variables

MLE given the estimated values of unobserved variables

(Soft) Expectation-Maximization

- Initialize **parameters** **randomly**
- **while** not converged
  1. **E-Step:**
     Create one training example for each possible value of the **latent variables**
     Weight each example according to model's confidence
     Treat parameters as observed
  2. **M-Step:**
     Set the **parameters** to the values that maximizes likelihood
     Treat pseudo-counts from above as observed

Estimate unobserved variables

MLE given the estimated values of unobserved variables

Hard EM vs. Soft EM

Algorithm 1 Hard EM for GMMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)
  Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
  while not converged **do**
    E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)} | z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z; \boldsymbol{\phi})$$

    M-Step:

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

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

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

  return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Algorithm 1 Soft EM for GMMs

procedure SOFTEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)
  Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
  while not converged **do**
    E-Step:

$$c_k^{(i)} \leftarrow p(z^{(i)} = k | \mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

    M-Step:

$$\phi_k \leftarrow \frac{1}{N} \sum_{i=1}^N c_k^{(i)}, \forall k$$

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

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

  return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Posterior Inference for Mixture Model

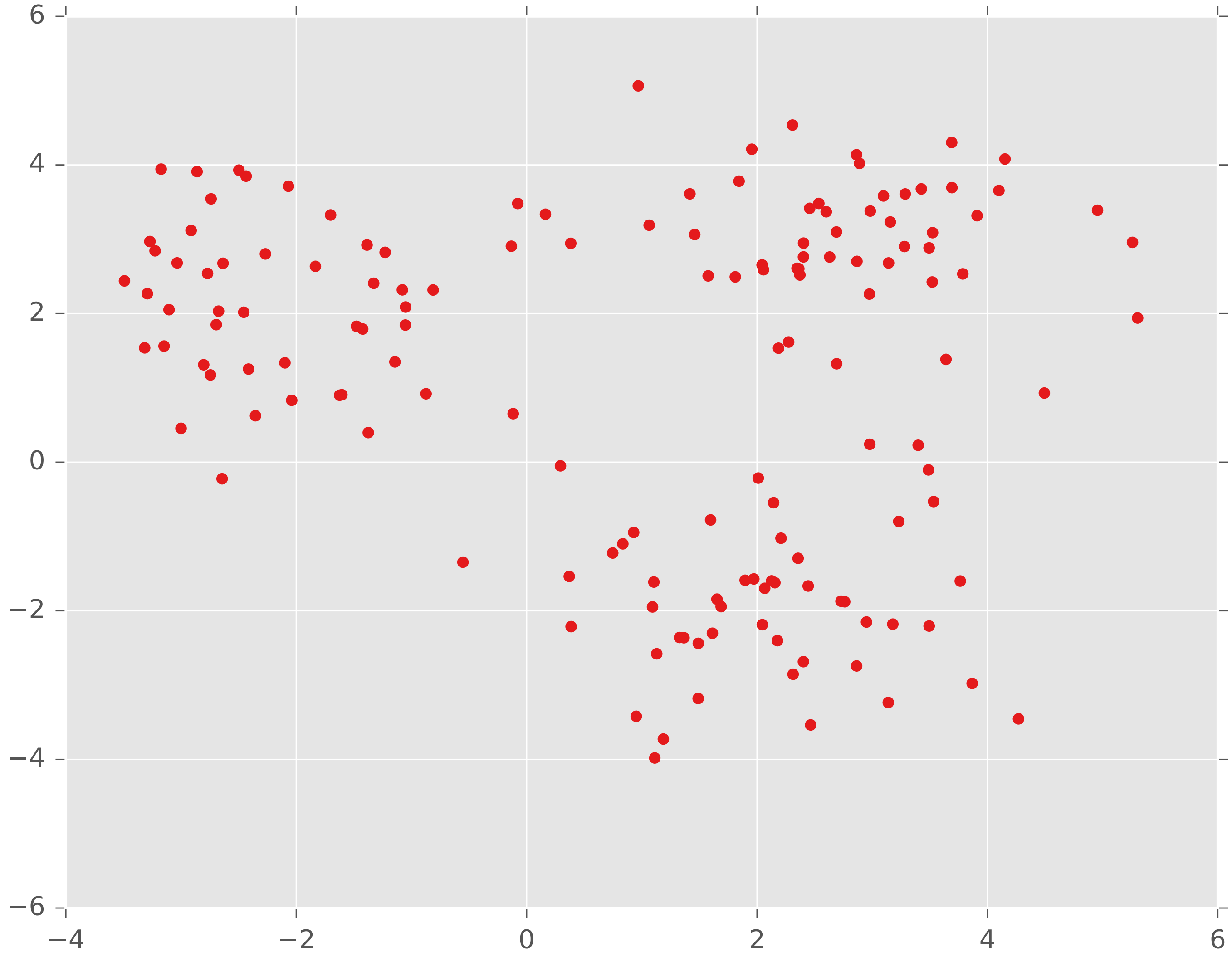
We obtain the posterior $p(z^{(i)} = k | x^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ as follows:

$$p(z^{(i)} = k | \mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{p(\mathbf{x}^{(i)} | z^{(i)} = k; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)} = k; \boldsymbol{\phi})}{\sum_{j=1}^{K} p(\mathbf{x}^{(i)} | z^{(i)} = j; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)} = j; \boldsymbol{\phi})} \quad (1)$$

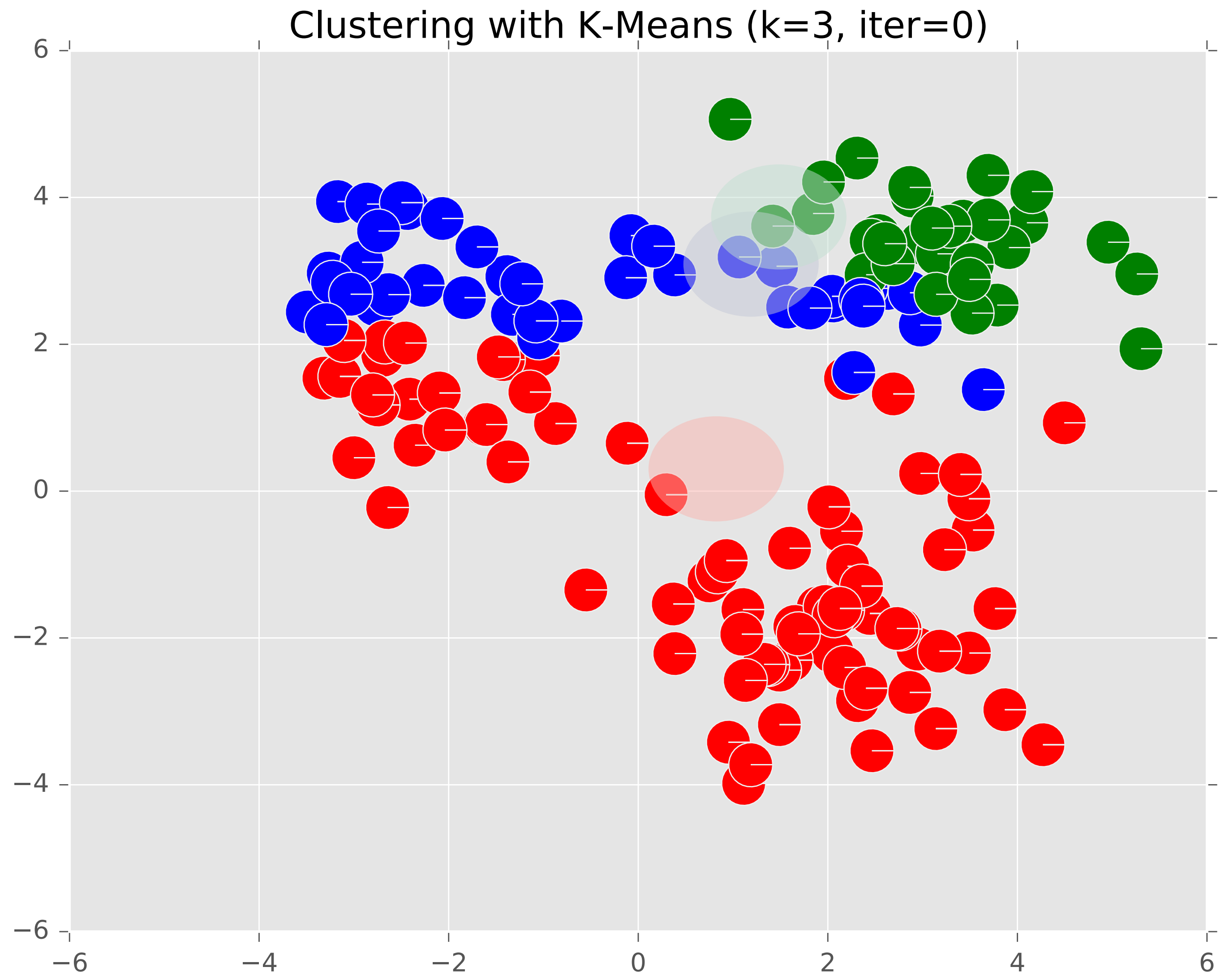
EXAMPLE: K-MEANS VS GMM

Example: K-Means

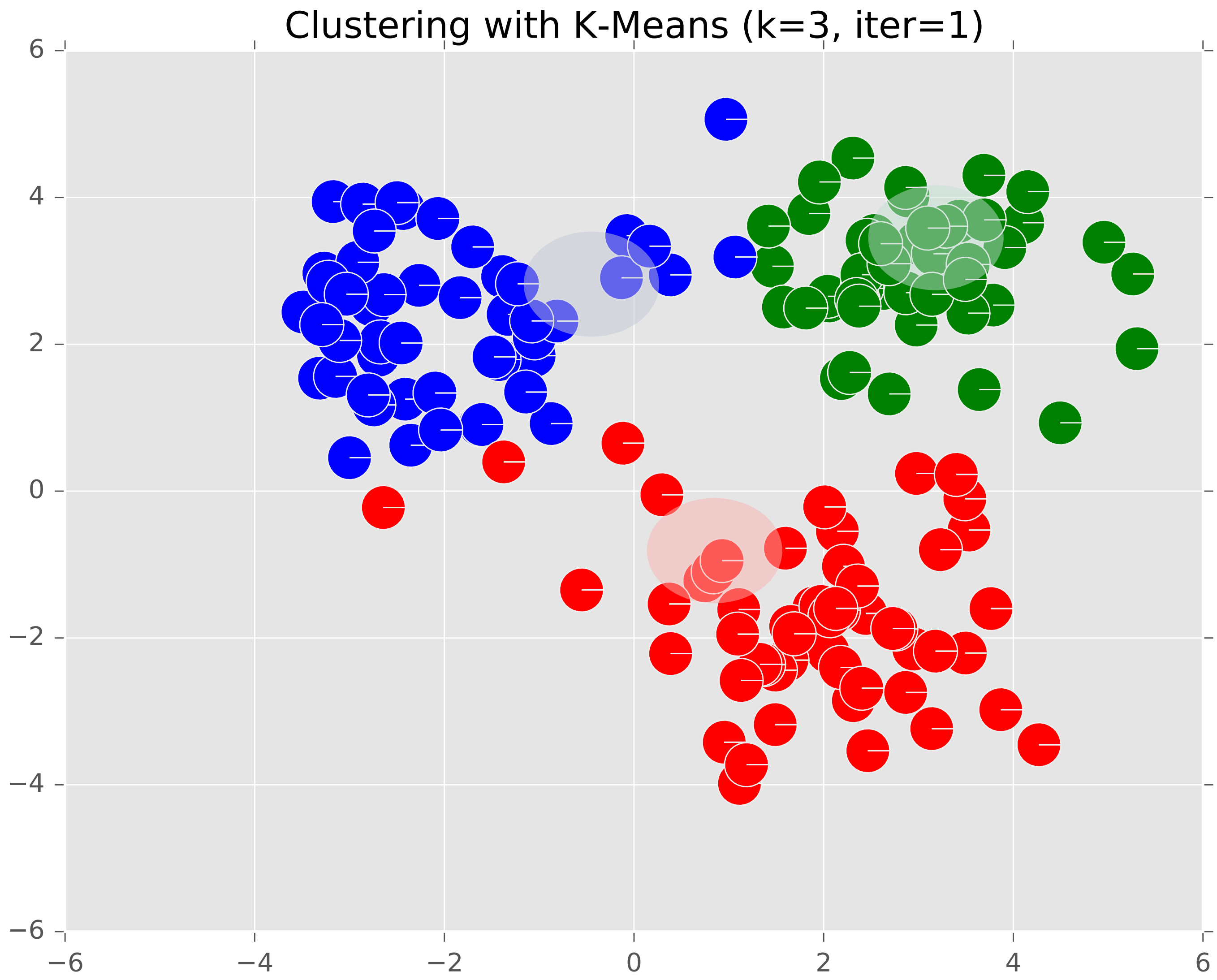
Example: K-Means



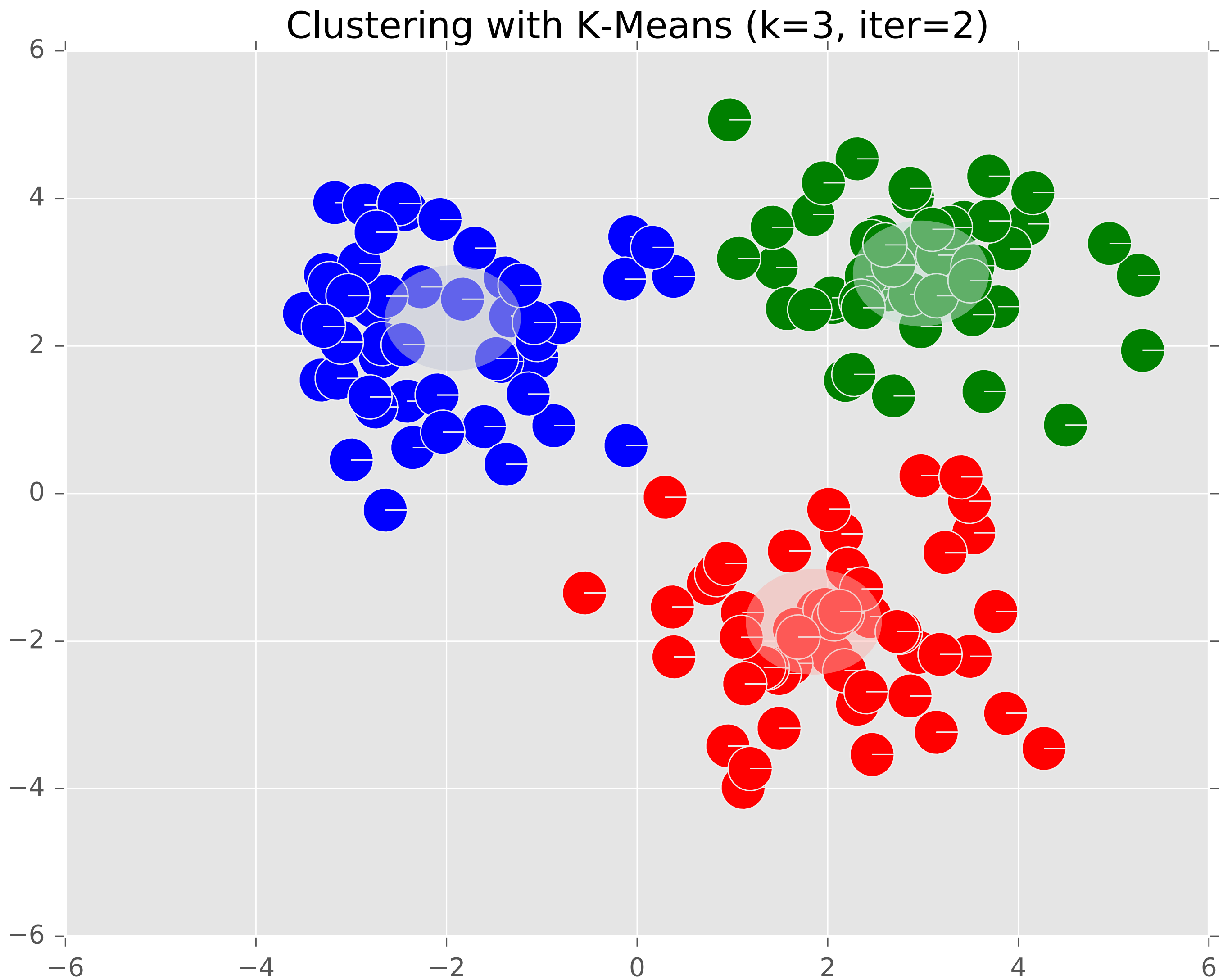
Example: K-Means



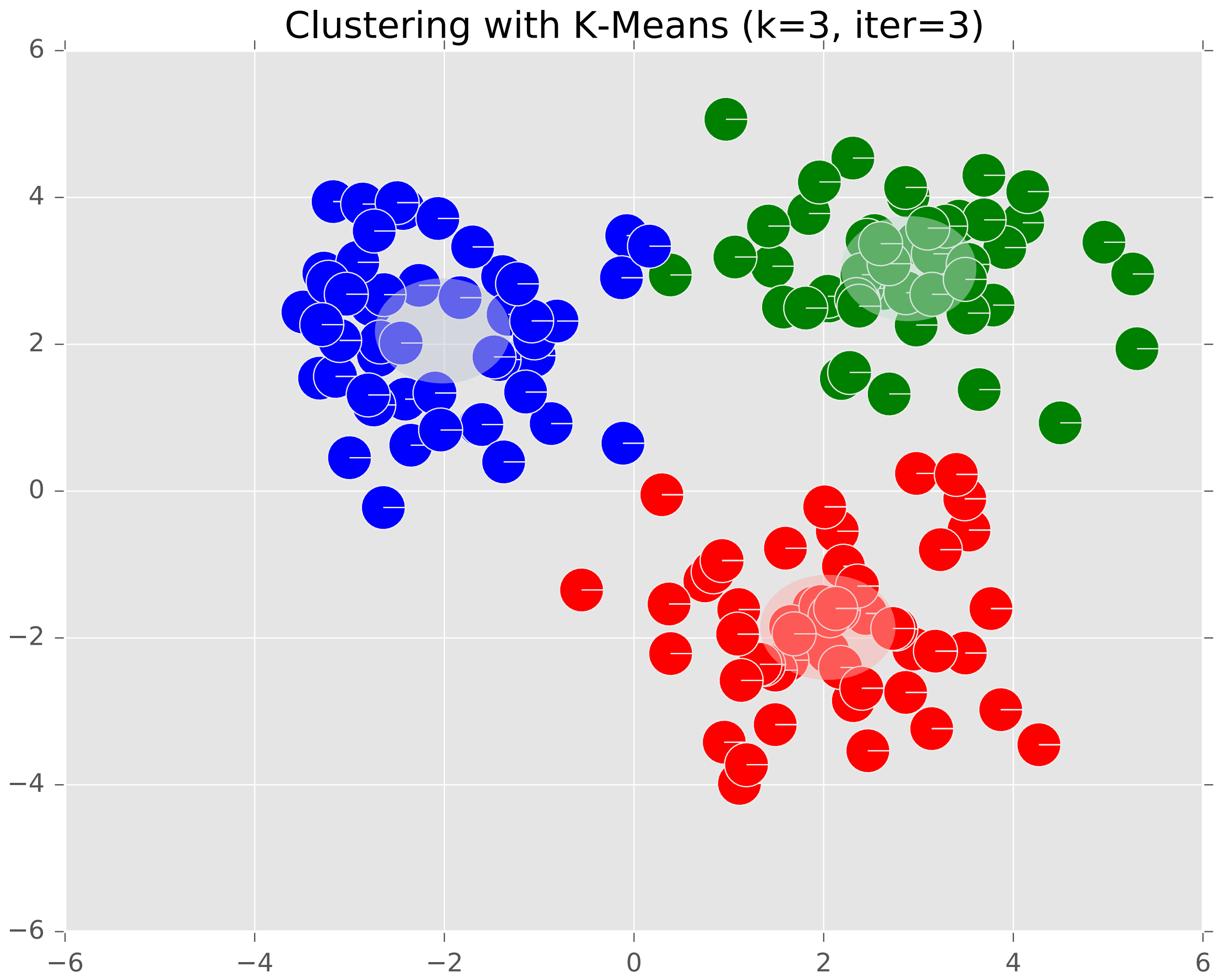
Example: K-Means



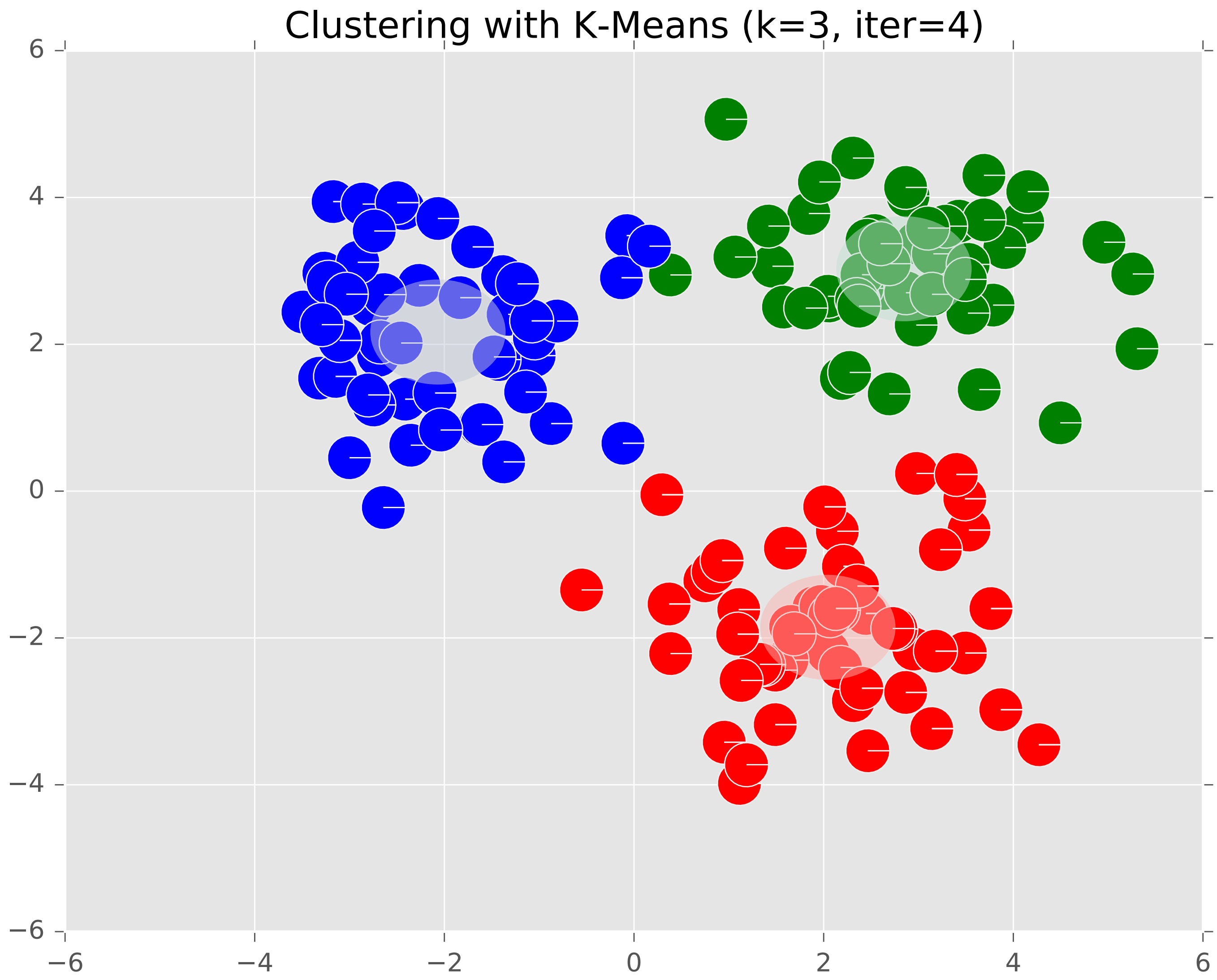
Example: K-Means



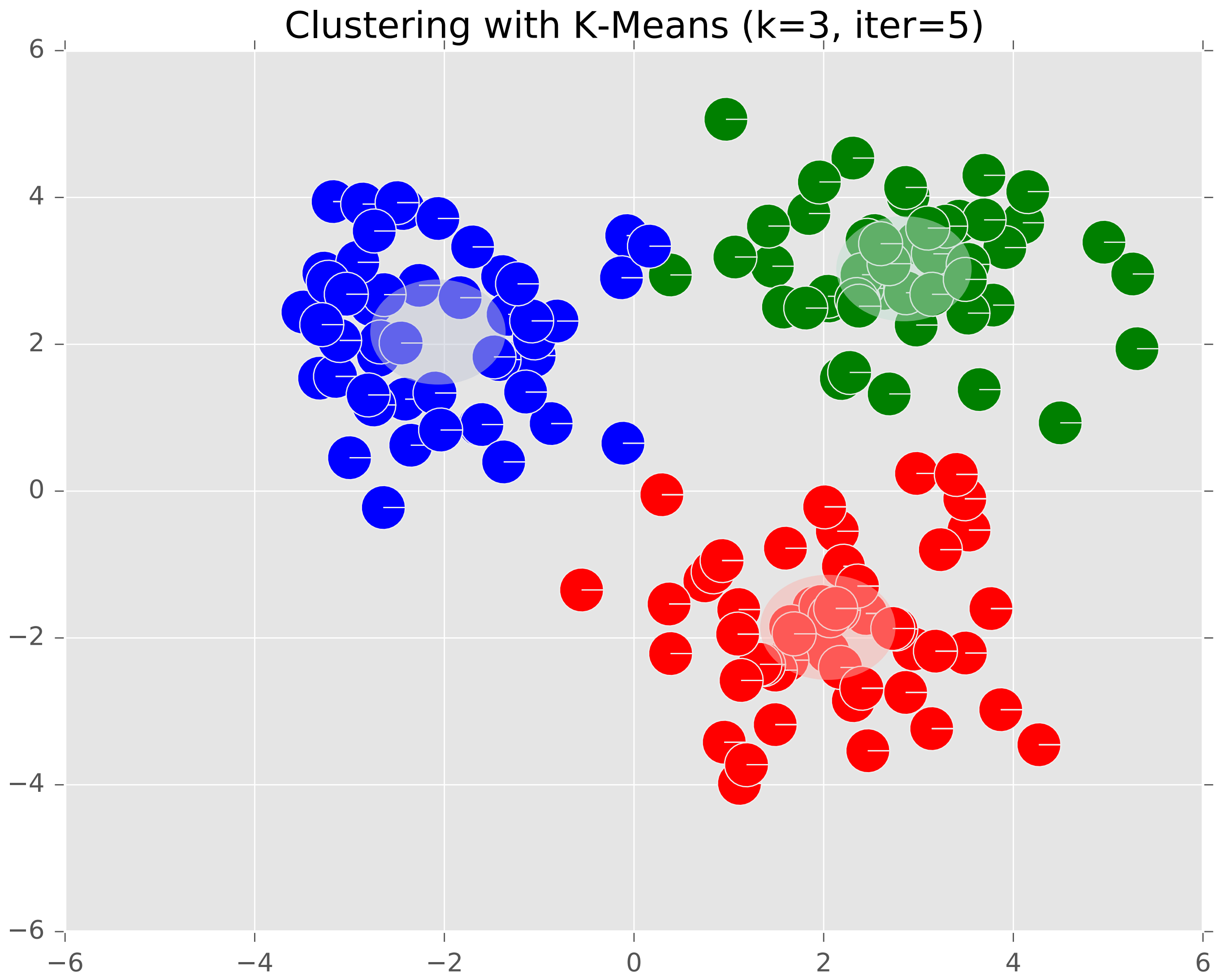
Example: K-Means



Example: K-Means

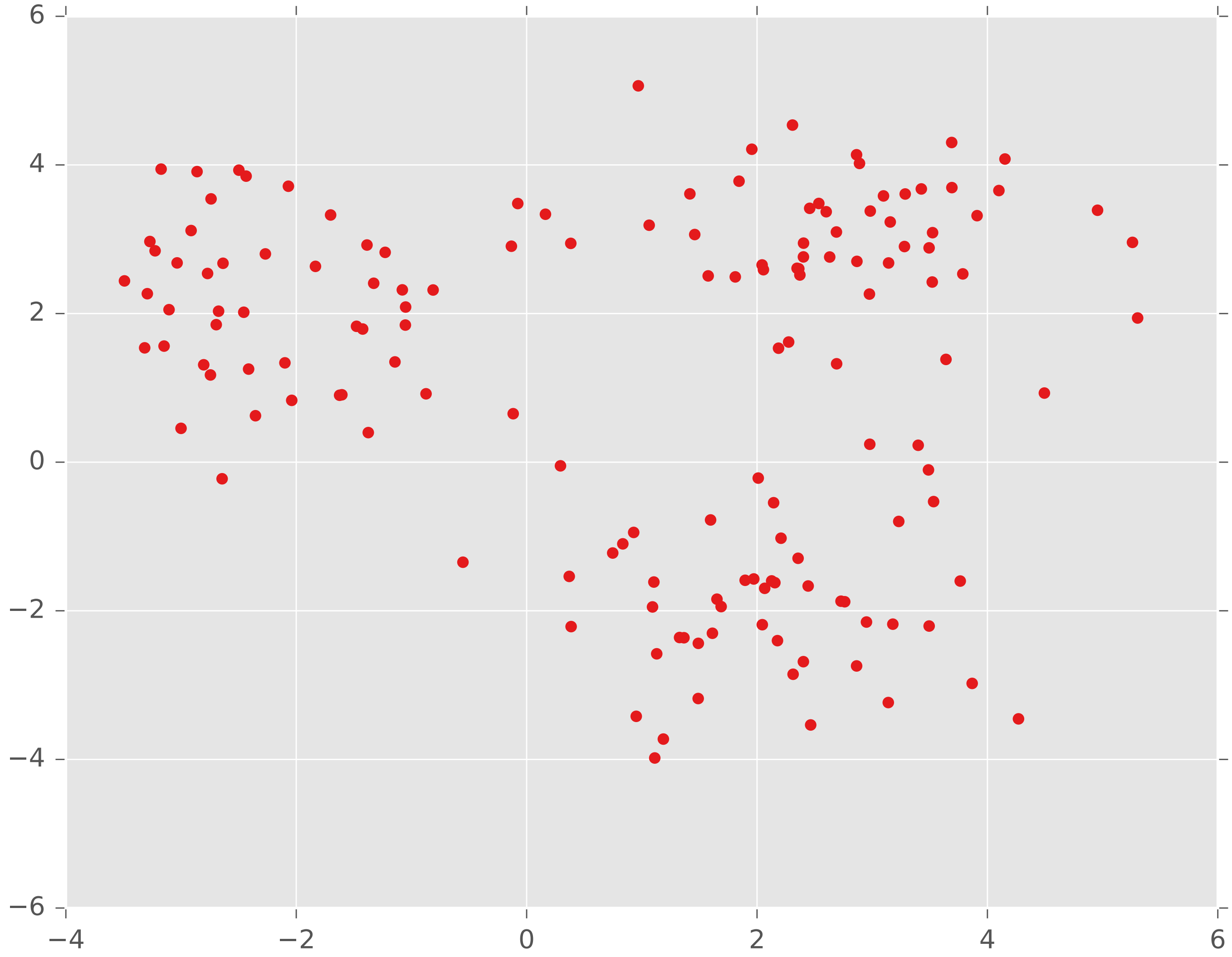


Example: K-Means

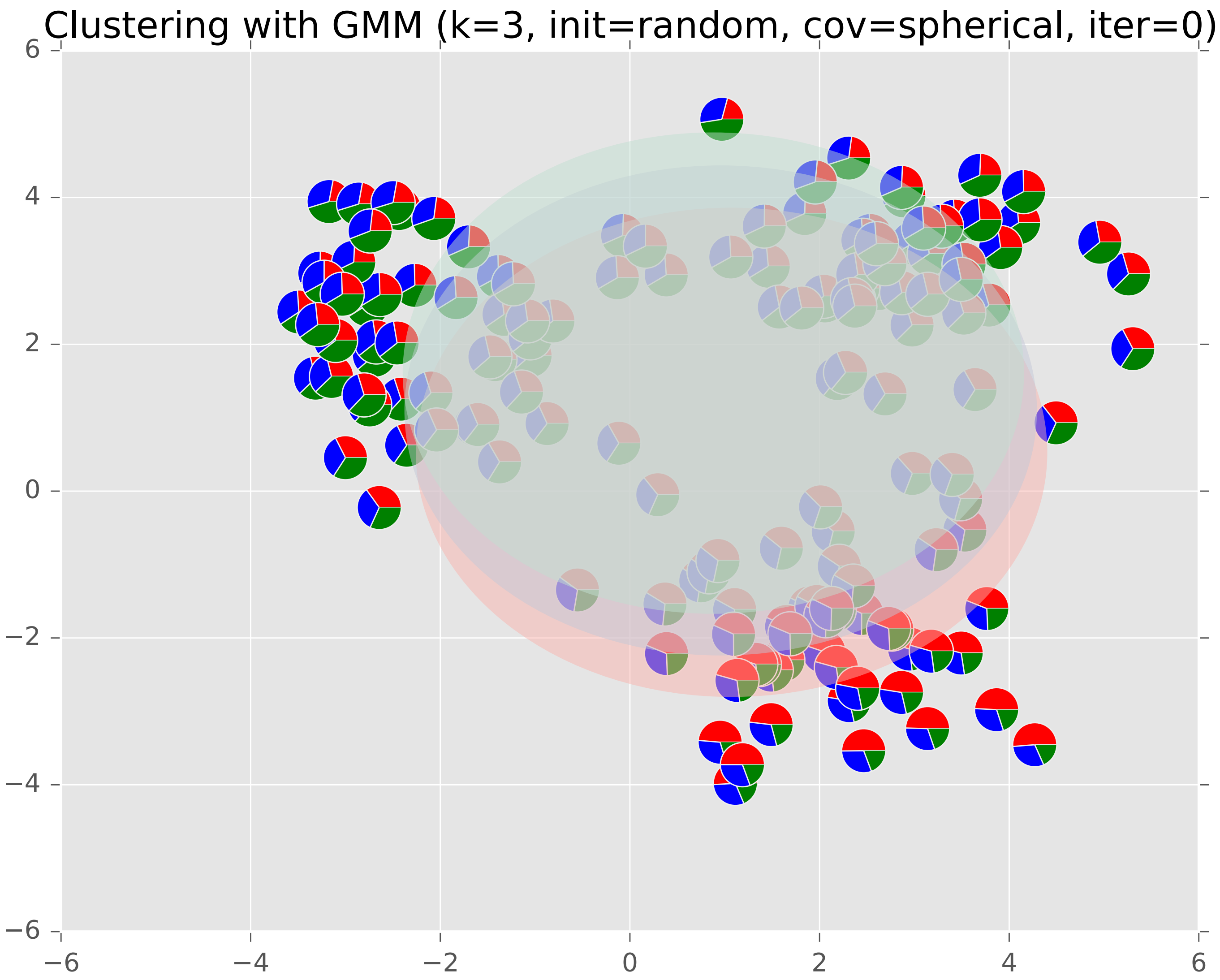


Example: GMM

Example: GMM

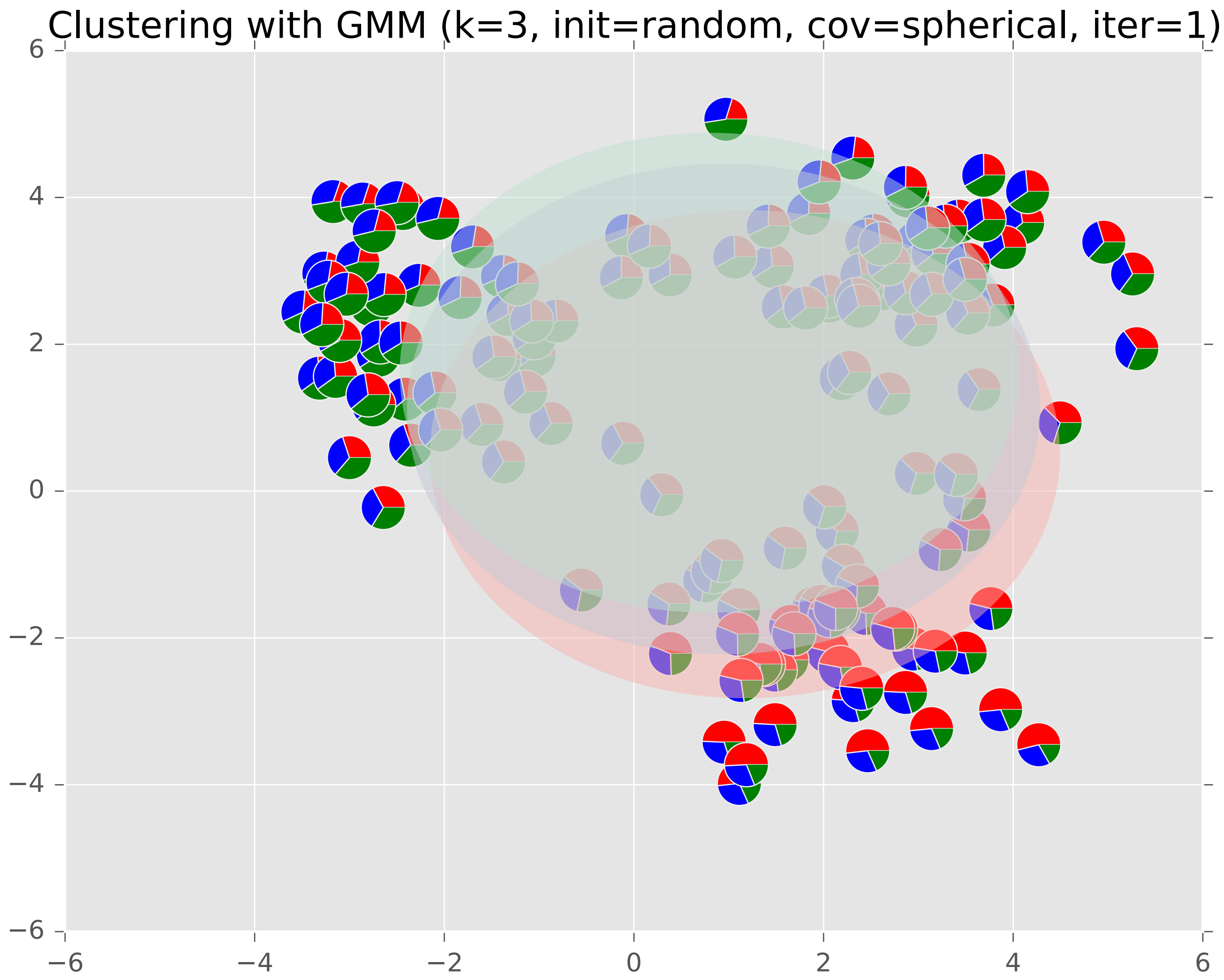


Example: GMM



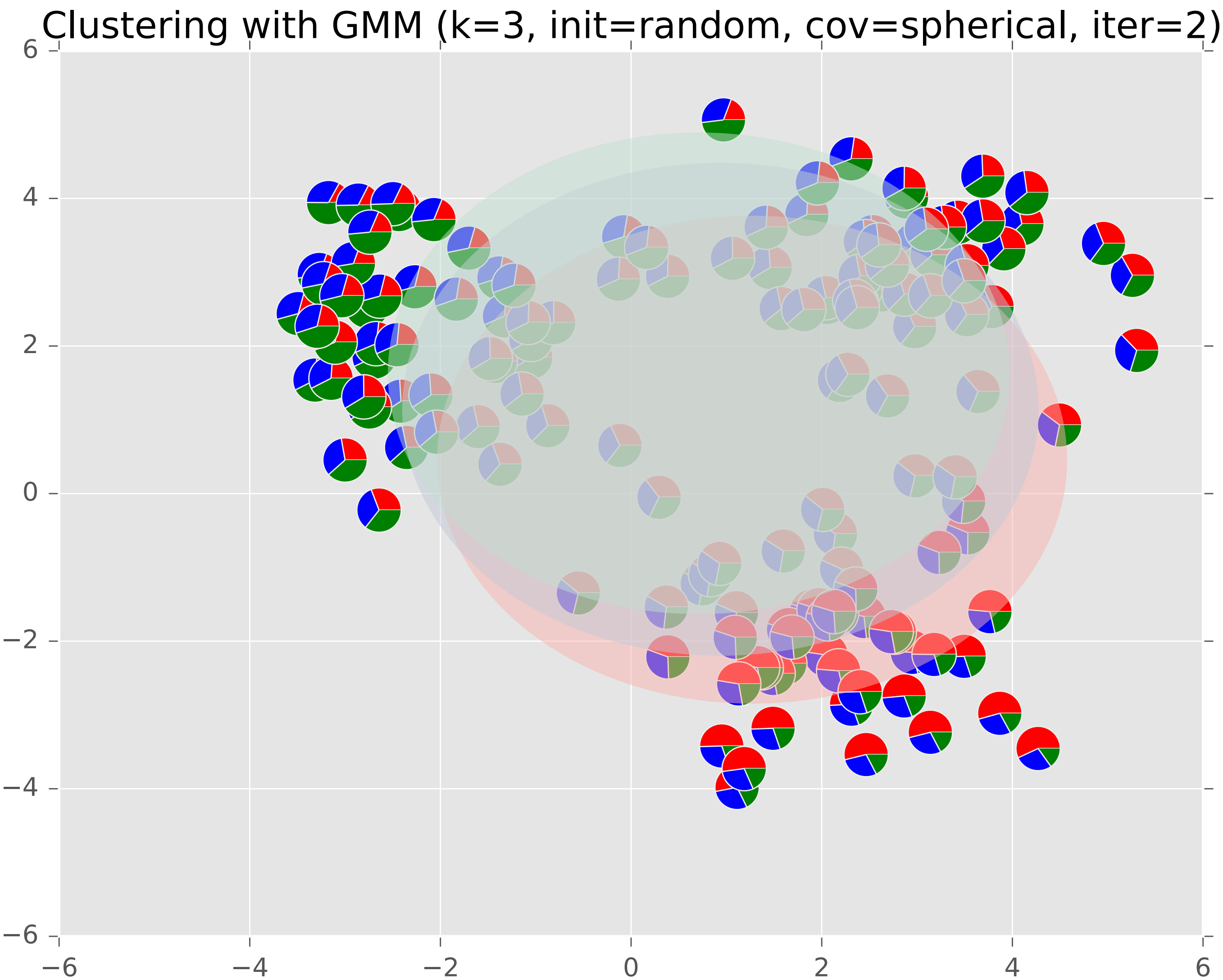
Clustering with GMM (k=3, init=random, cov=spherical, iter=0)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM



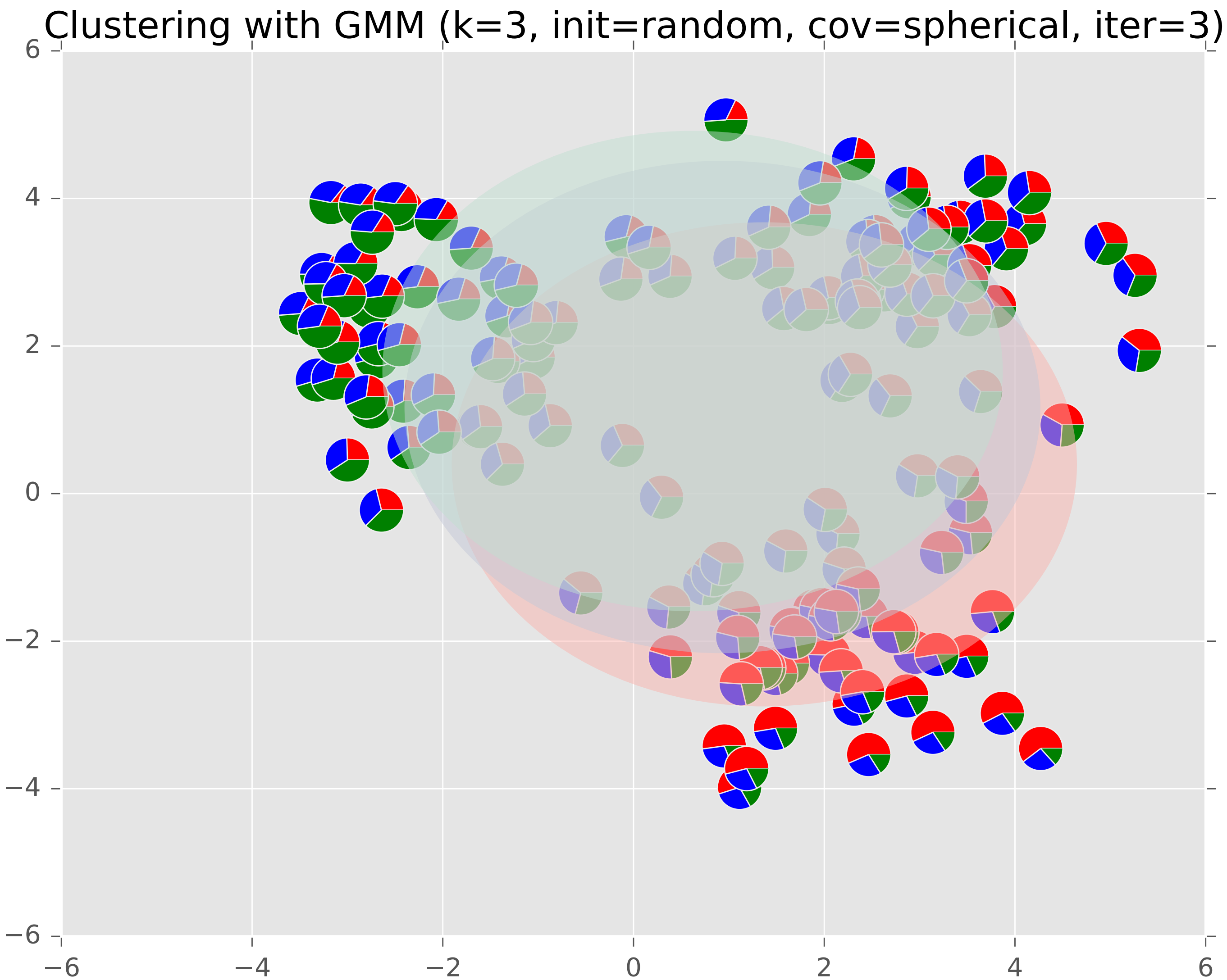
Clustering with GMM (k=3, init=random, cov=spherical, iter=1)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM



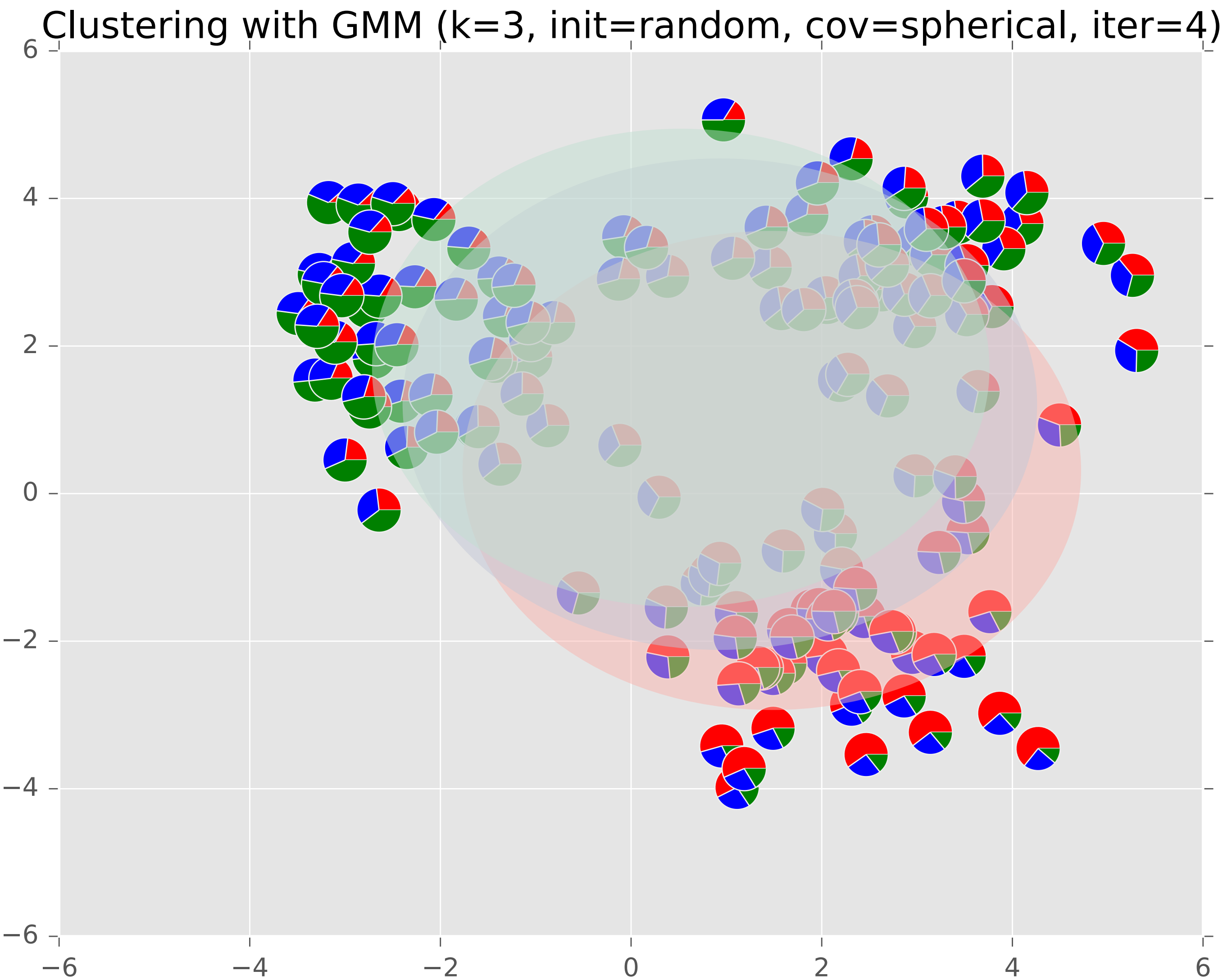
Clustering with GMM (k=3, init=random, cov=spherical, iter=2)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM



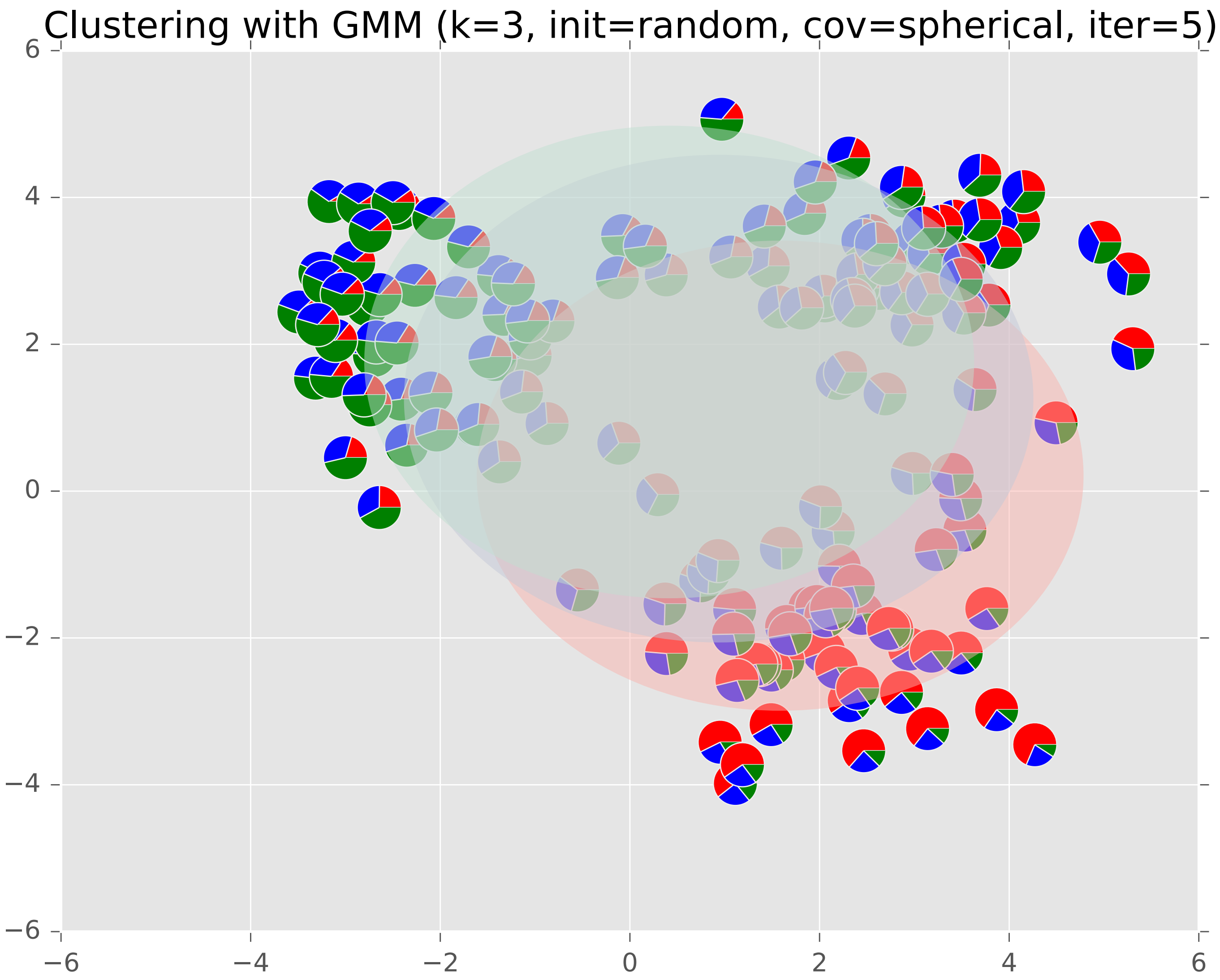
Clustering with GMM (k=3, init=random, cov=spherical, iter=3)
6
4
2
0
−2
−4
−6
−6 −4 −2 0 2 4 6

Example: GMM



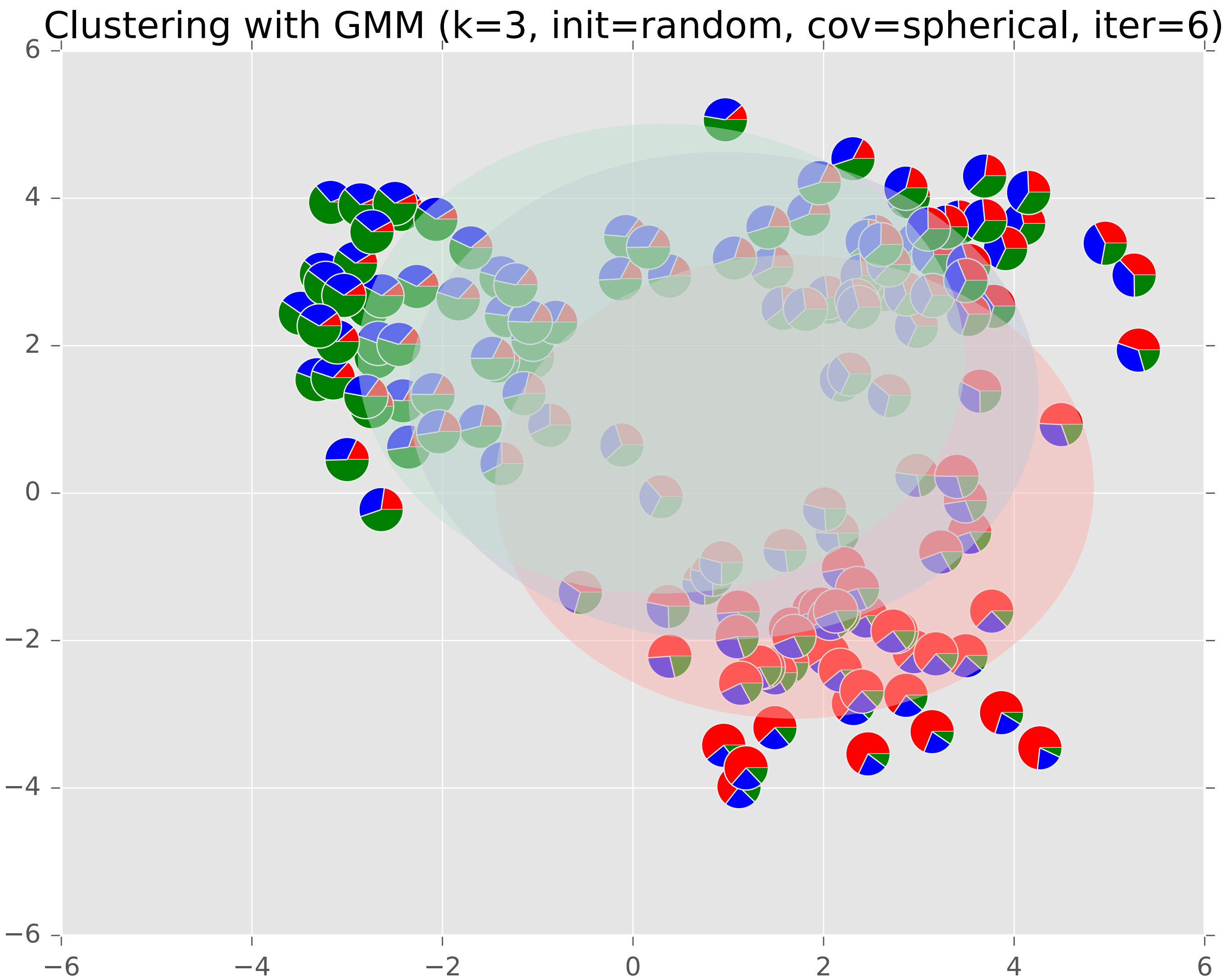
Clustering with GMM (k=3, init=random, cov=spherical, iter=4)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM



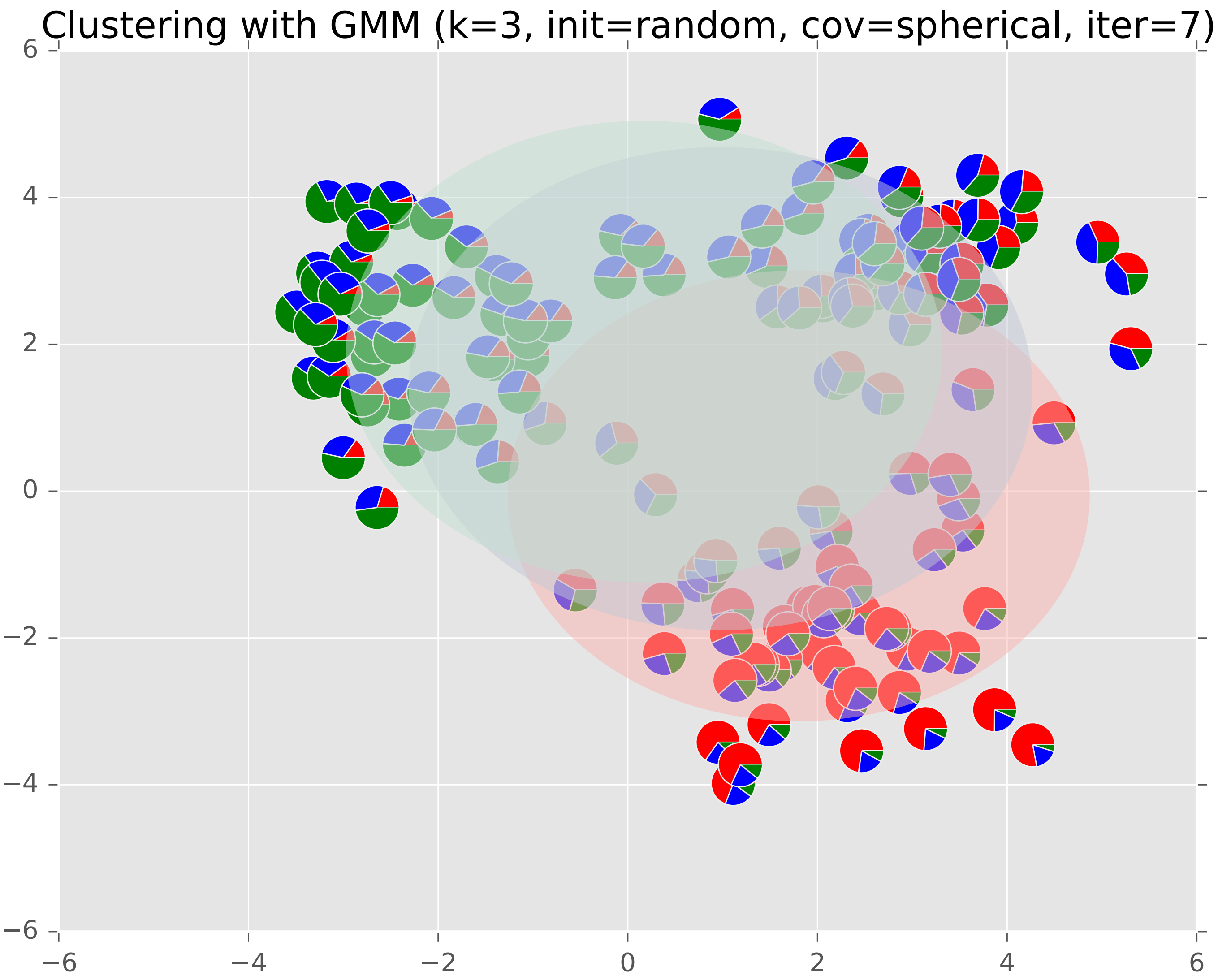
Clustering with GMM (k=3, init=random, cov=spherical, iter=5)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM



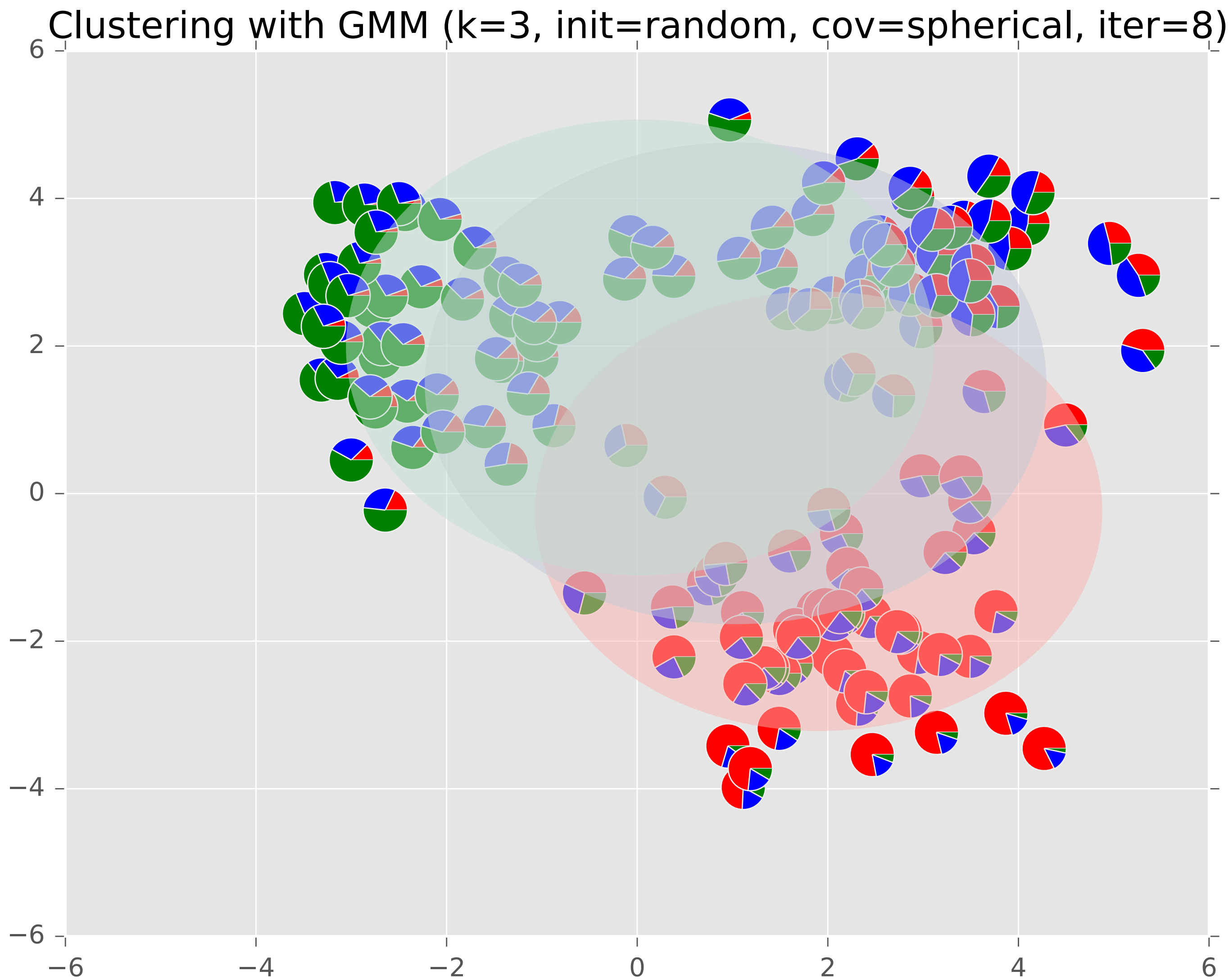
Clustering with GMM (k=3, init=random, cov=spherical, iter=6)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM

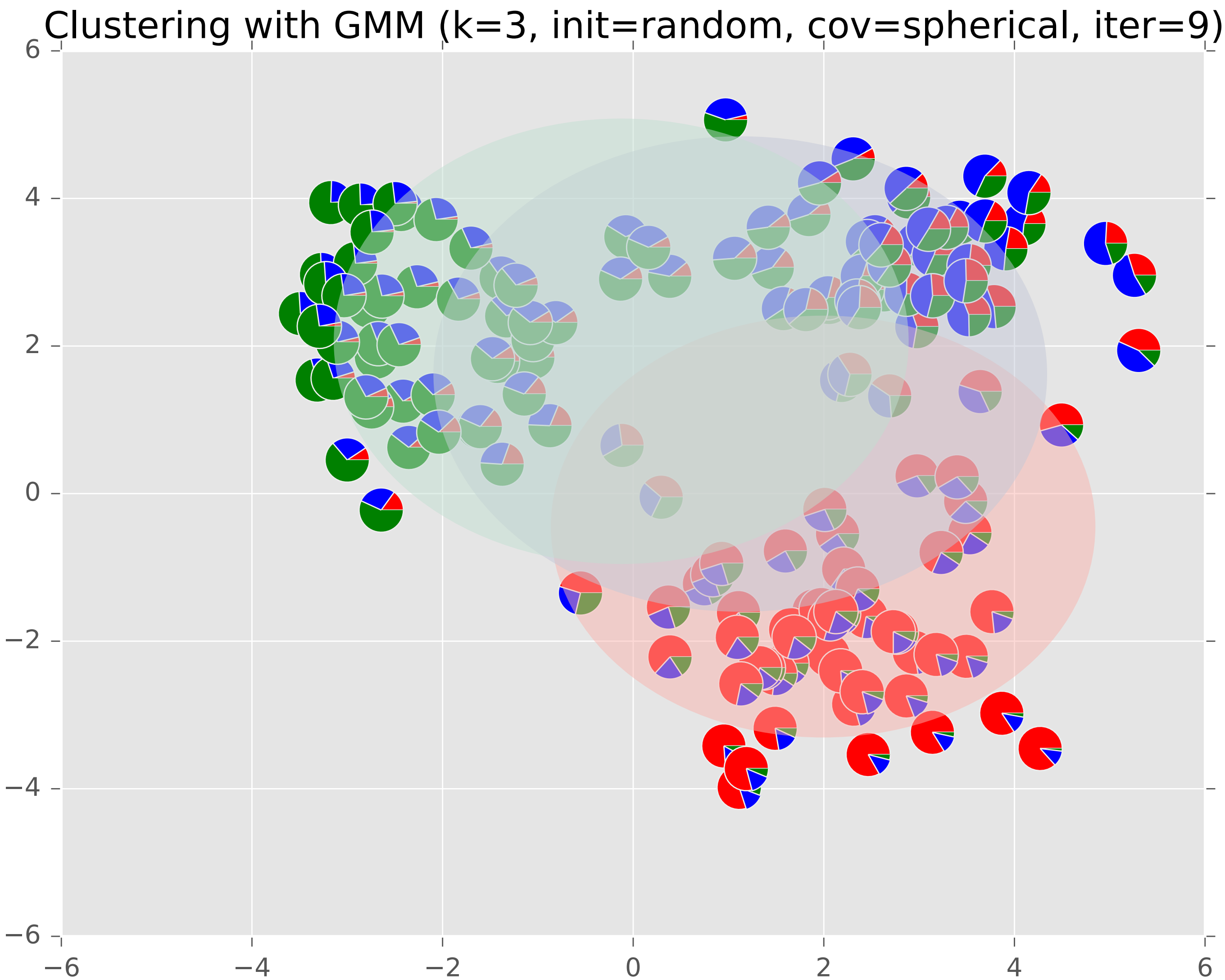


Clustering with GMM (k=3, init=random, cov=spherical, iter=7)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM

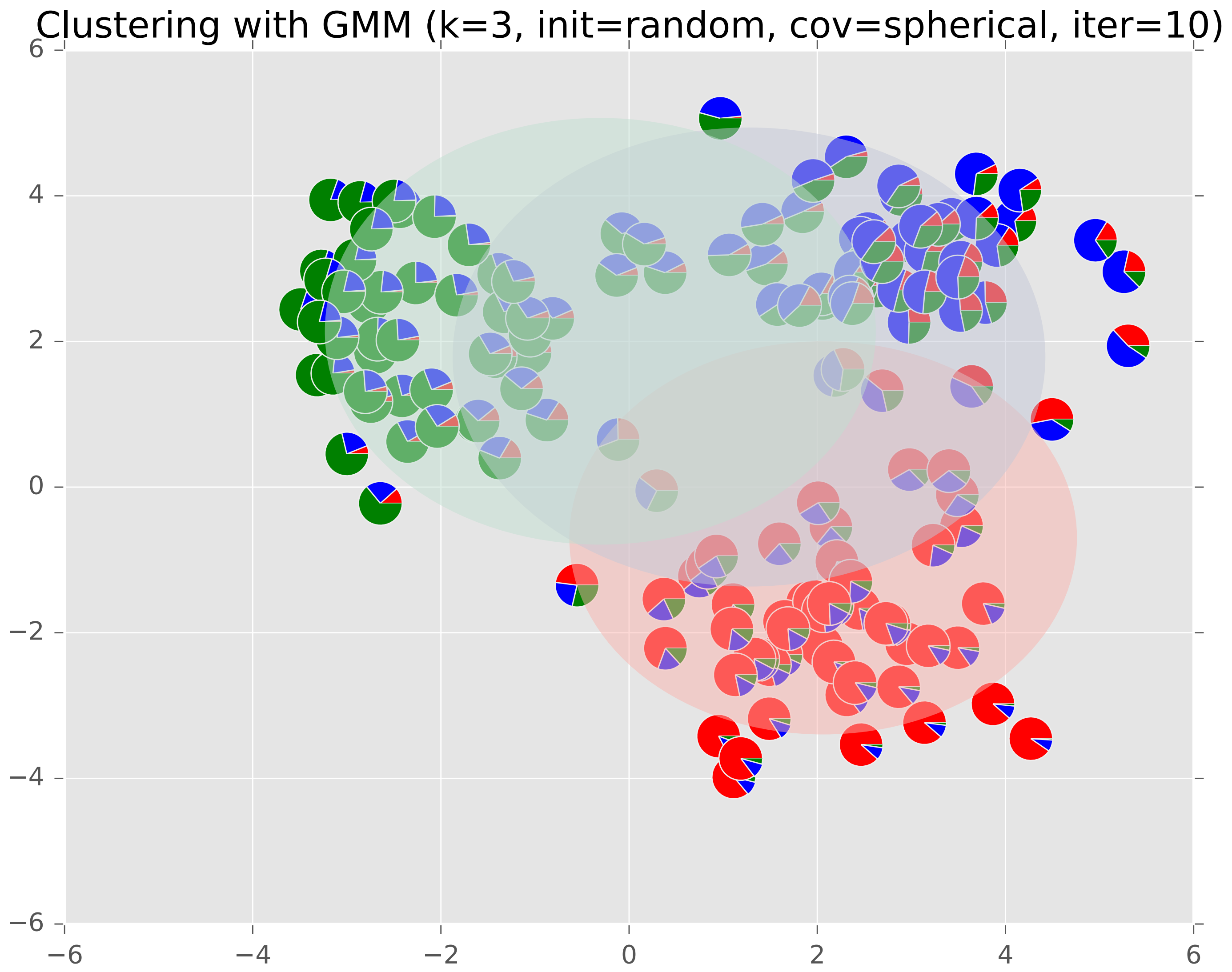

Clustering with GMM (k=3, init=random, cov=spherical, iter=8)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM

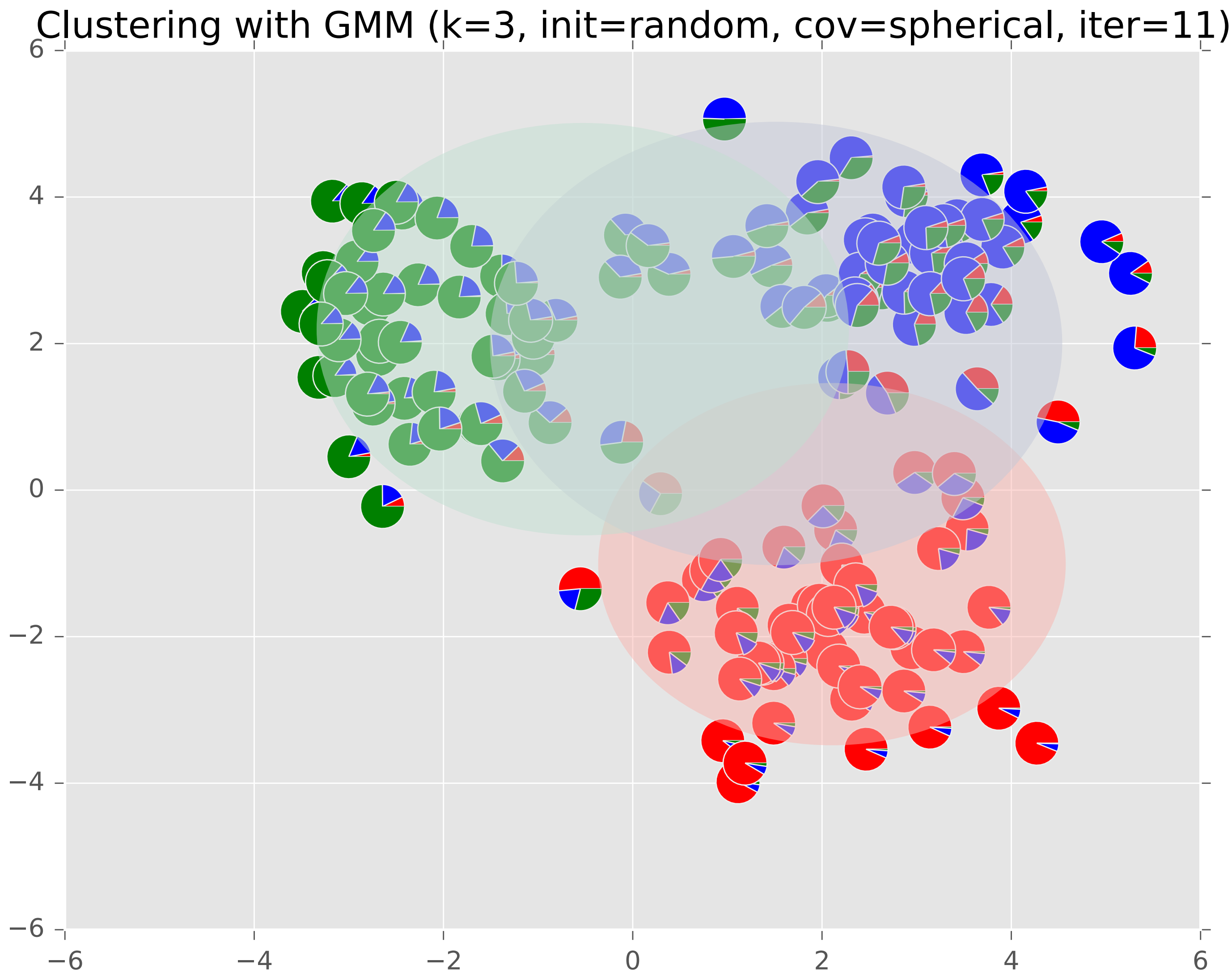


Clustering with GMM (k=3, init=random, cov=spherical, iter=9)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

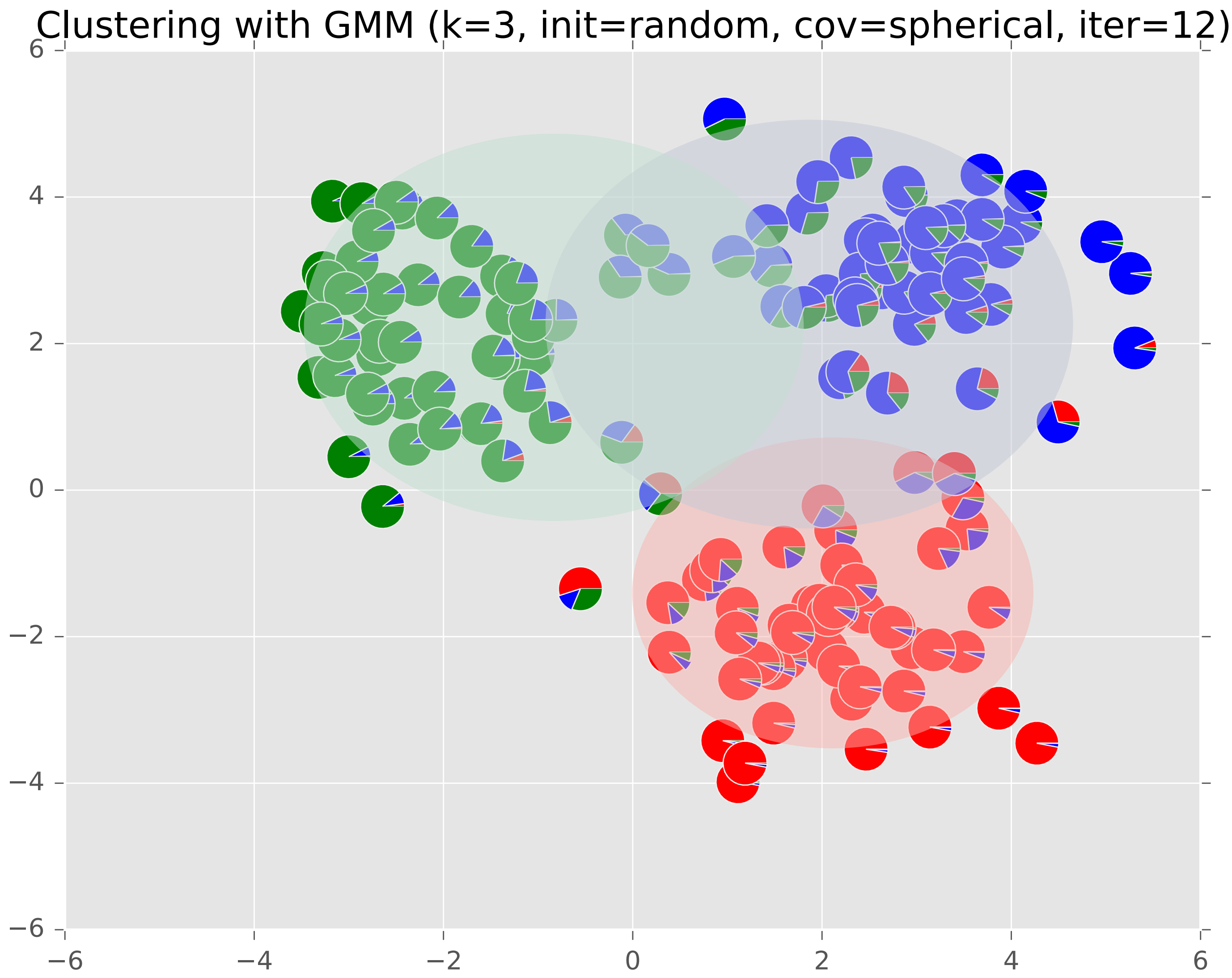
Example: GMM


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

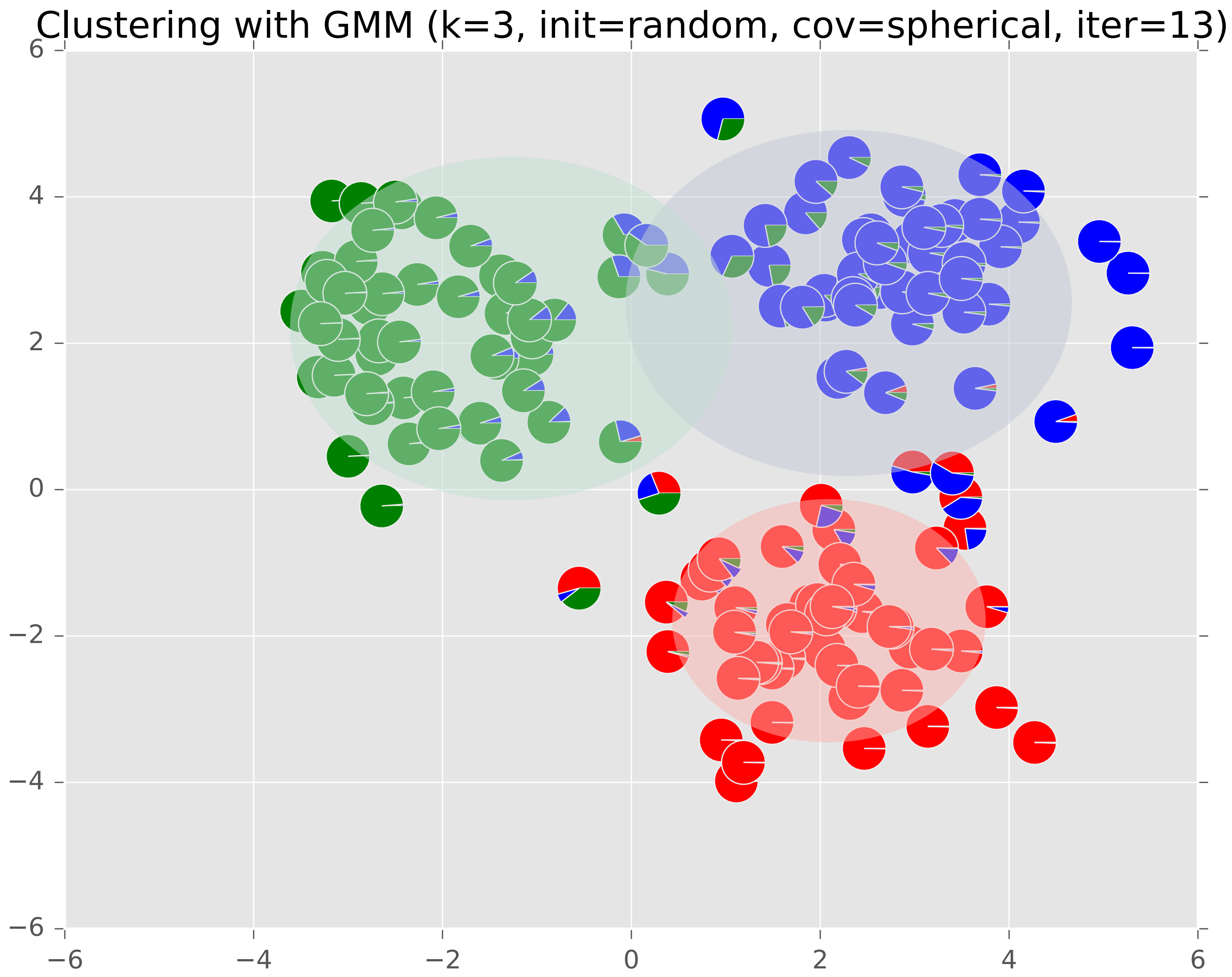
Example: GMM


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

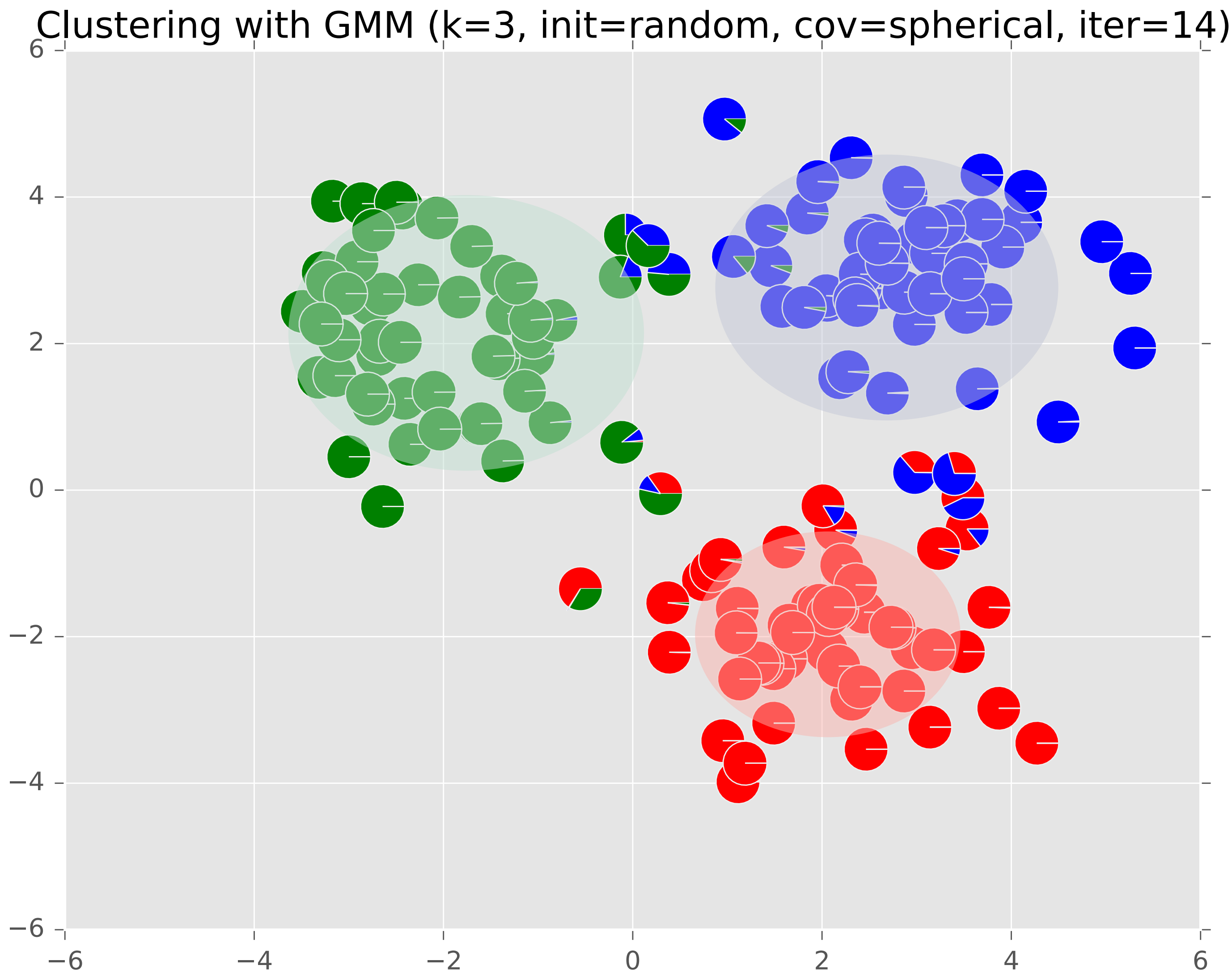

Clustering with GMM (k=3, init=random, cov=spherical, iter=12)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Example: GMM

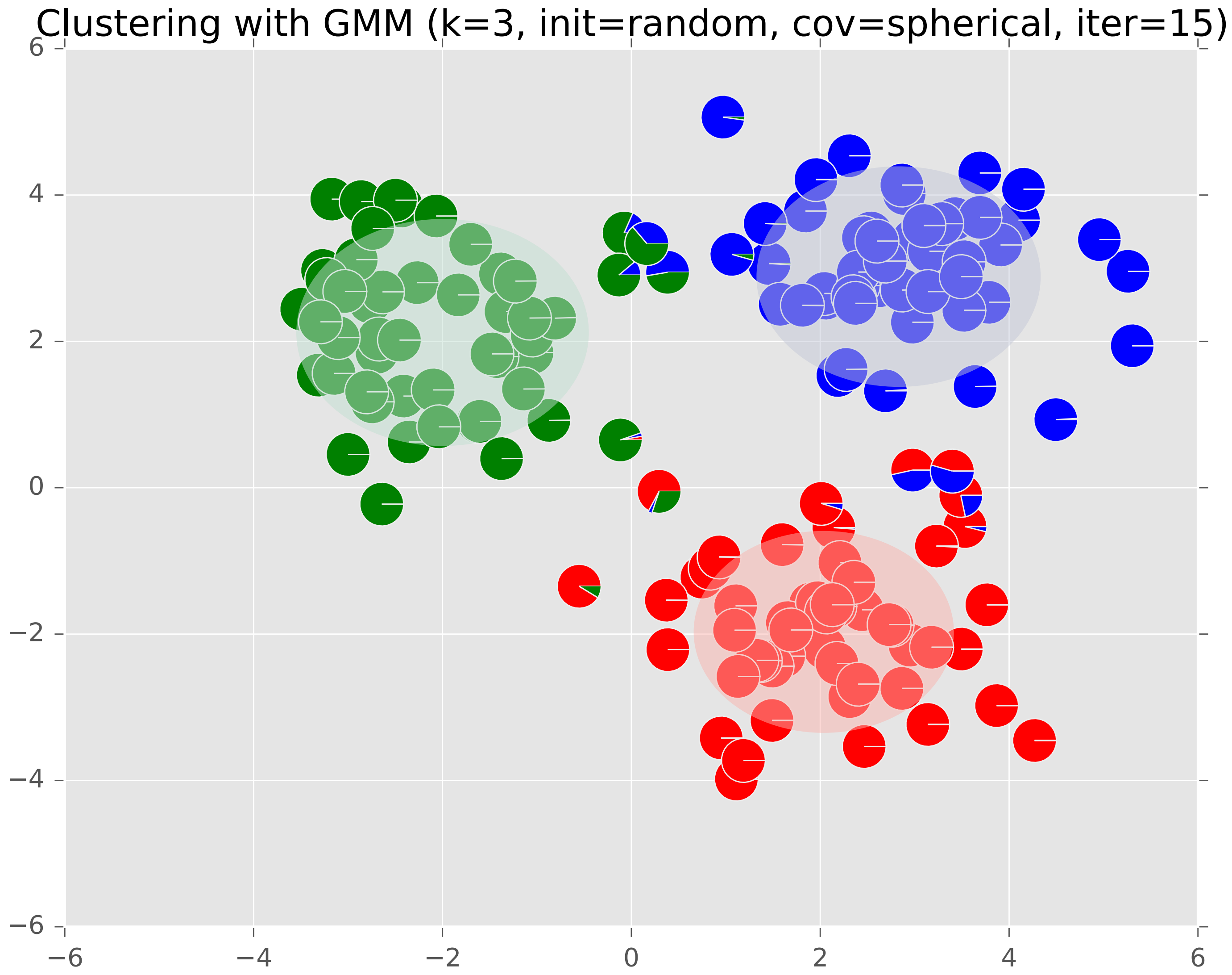


Clustering with GMM (k=3, init=random, cov=spherical, iter=13)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

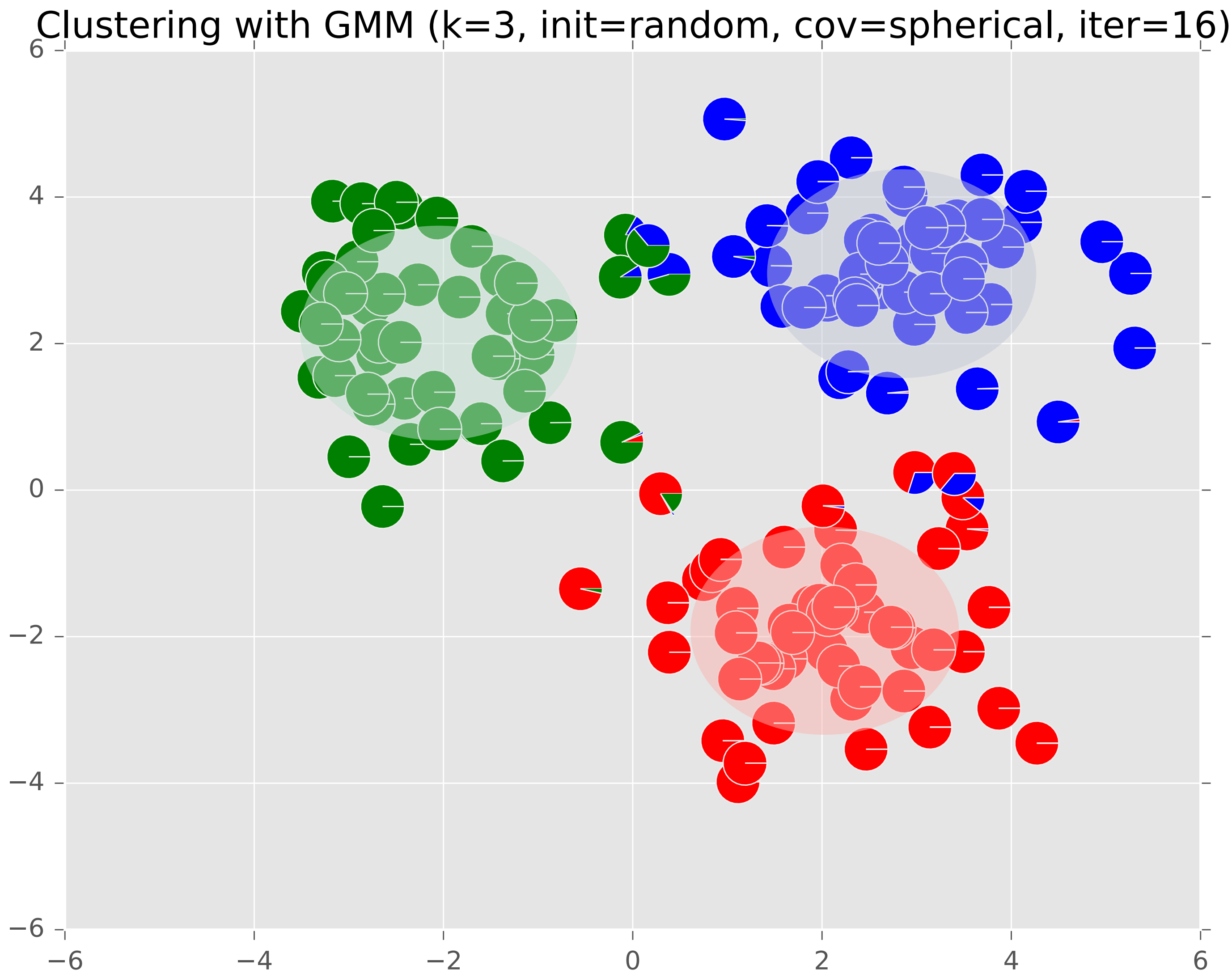
Example: GMM


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

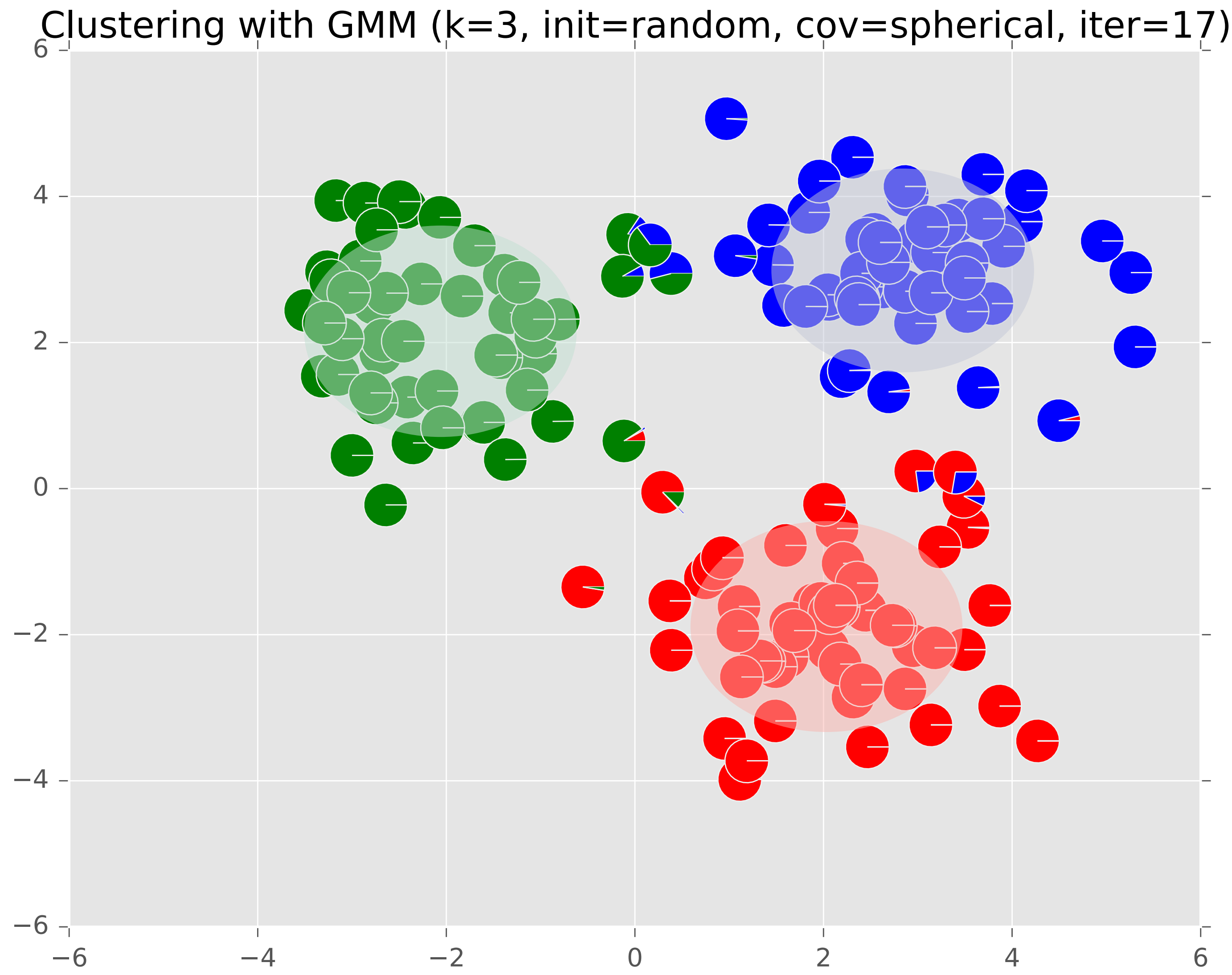
Example: GMM


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

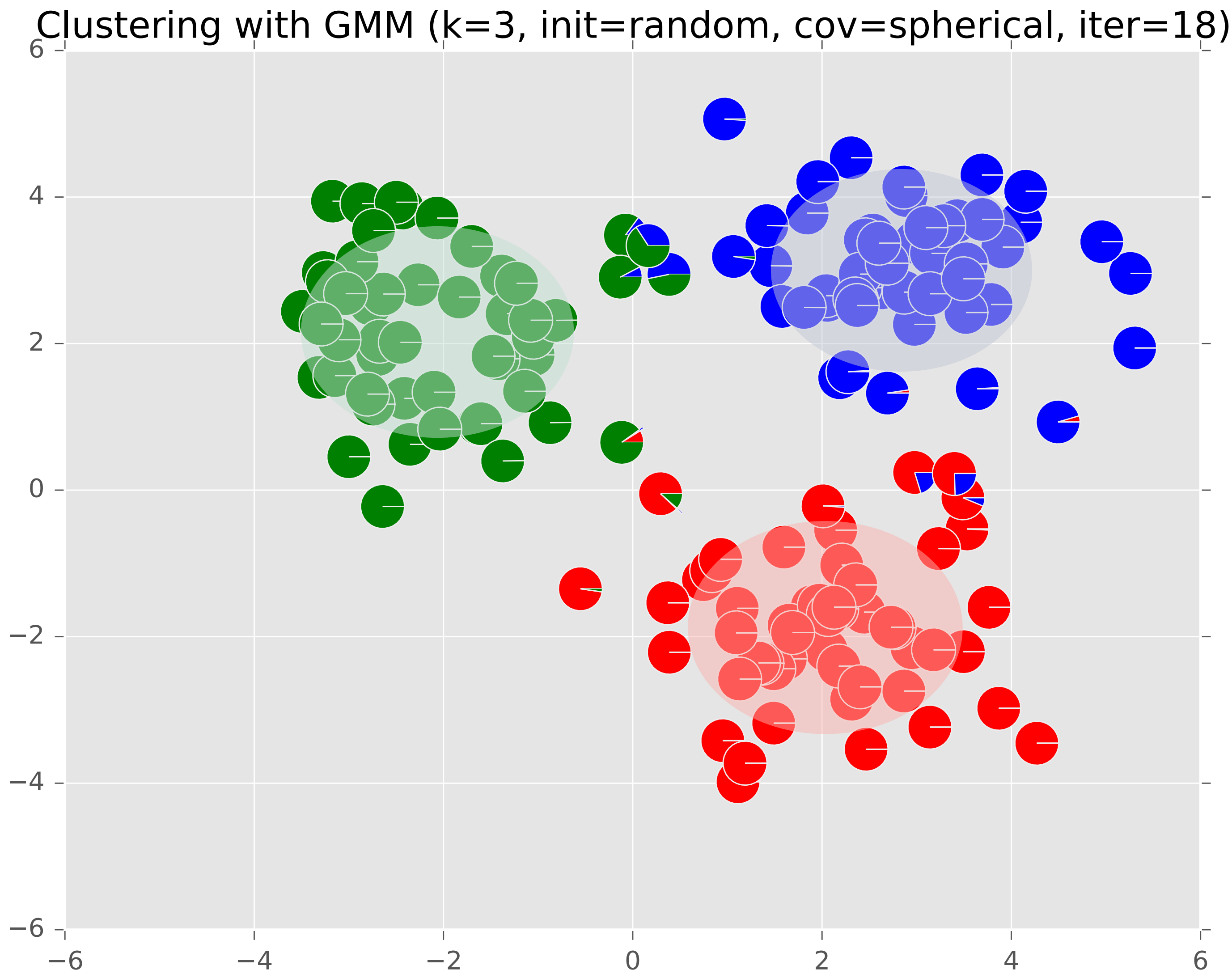
Example: GMM


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

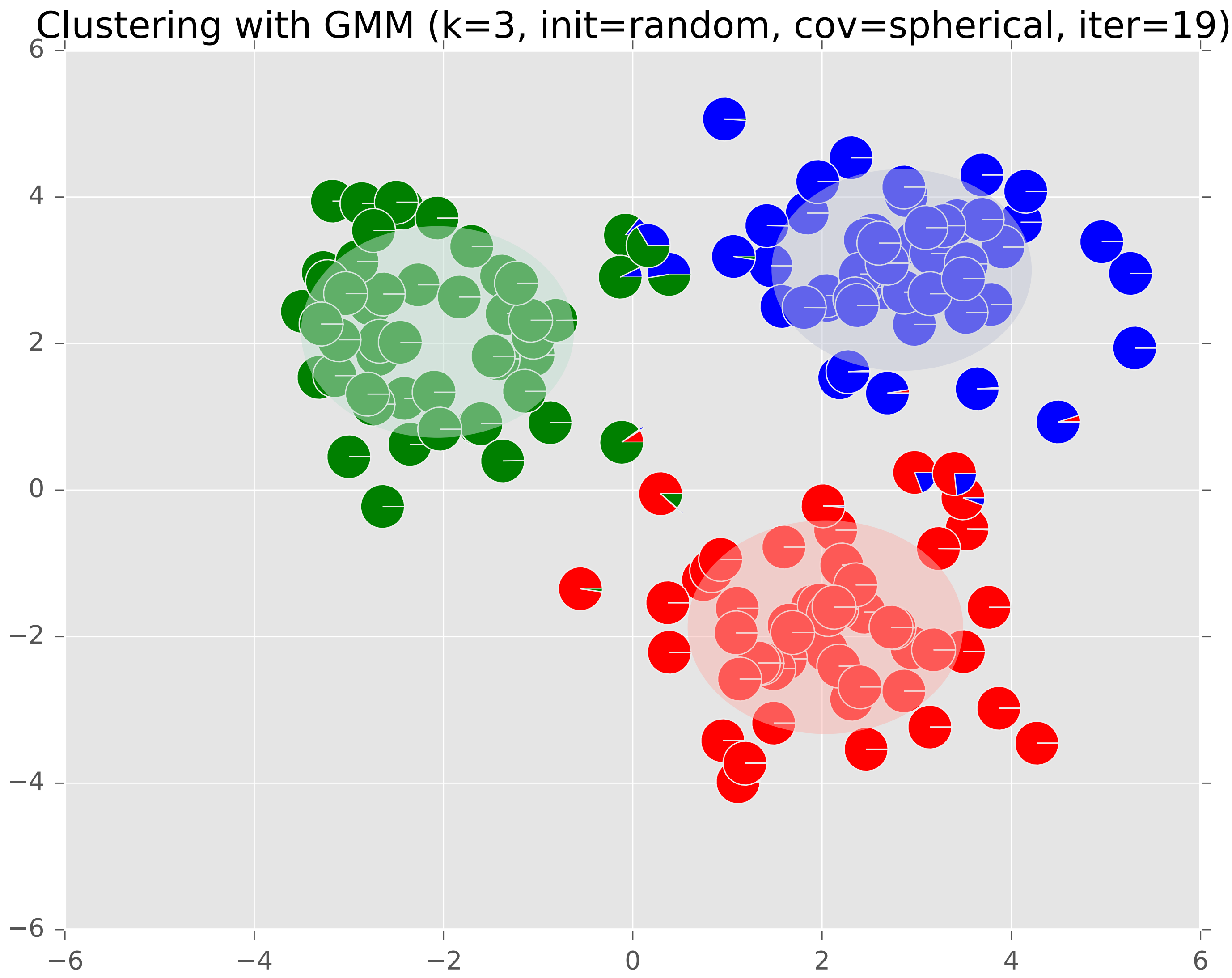
Example: GMM


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

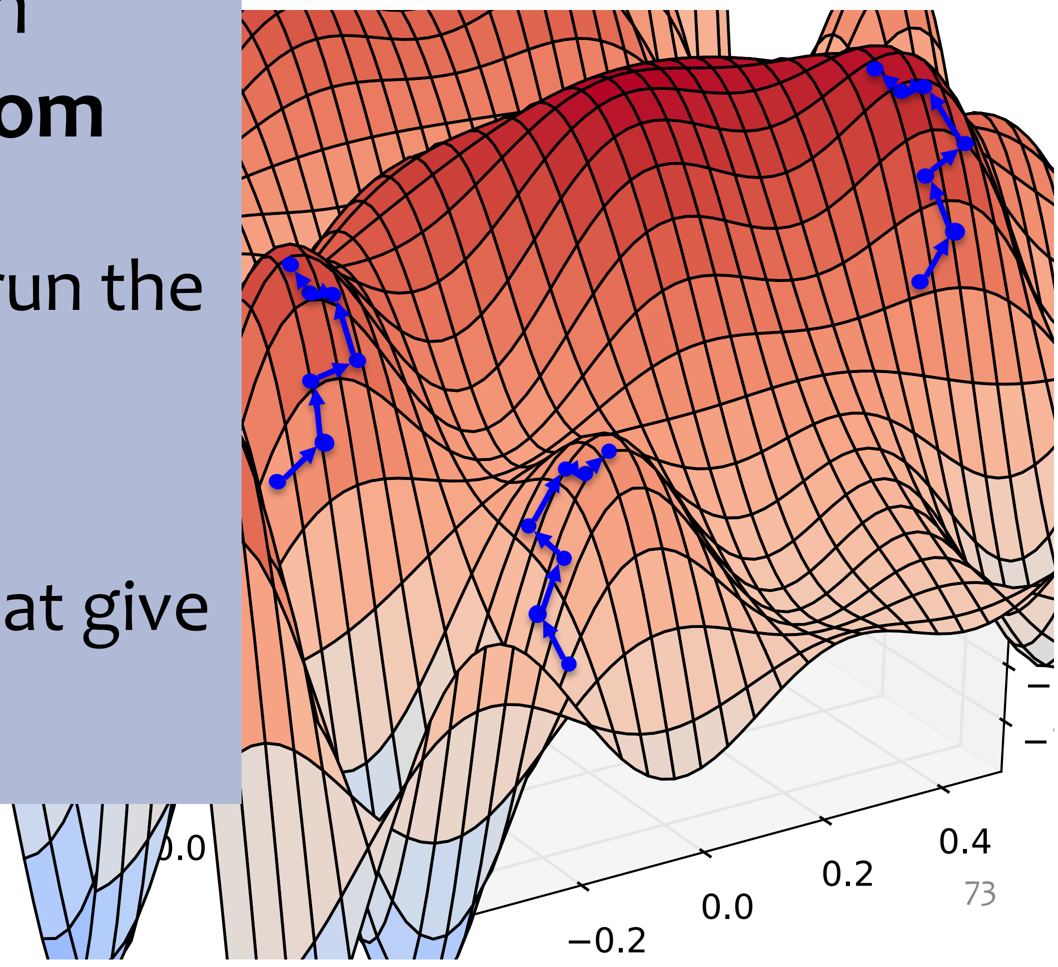
Output:

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

PROPERTIES OF EM

Properties of (Variational) 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



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

VARIATIONAL EM

Variational EM

Whiteboard

- Example: Unsupervised POS Tagging
- Variational Bayes
- Variational EM

Unsupervised POS Tagging

Bayesian Inference for HMMs

- **Task**: unsupervised POS tagging
- **Data:** 1 million words (i.e. unlabeled sentences) of WSJ text
- **Dictionary**: defines legal part-of-speech (POS) tags for each word type
- **Models**:
  - EM: standard HMM
  - VB: uncollapsed variational Bayesian HMM
  - Algo 1 (CVB): collapsed variational Bayesian HMM (strong indep. assumption)
  - Algo 2 (CVB): collapsed variational Bayesian HMM (weaker indep. assumption)
  - CGS: collapsed Gibbs Sampler for Bayesian HMM

Algo 1 mean field update:

$$q(z_t = k) \propto \frac{\mathbb{E}_{q(\mathbf{z}^{\neg t})}[C^{\neg t}_{k,w}] + \beta}{\mathbb{E}_{q(\mathbf{z}^{\neg t})}[C^{\neg t}_{k,\cdot}] + W\beta} \cdot \frac{\mathbb{E}_{q(\mathbf{z}^{\neg t})}[C^{\neg t}_{z_{t-1},k}] + \alpha}{\mathbb{E}_{q(\mathbf{z}^{\neg t})}[C^{\neg t}_{z_{t-1},\cdot}] + K\alpha} \cdot \frac{\mathbb{E}_{q(\mathbf{z}^{\neg t})}[C^{\neg t}_{k,z_{t+1}}] + \alpha + \mathbb{E}_{q(\mathbf{z}^{\neg t})}[\delta(z_{t-1} = k = z_{t+1})]}{\mathbb{E}_{q(\mathbf{z}^{\neg t})}[C^{\neg t}_{k,\cdot}] + K\alpha + \mathbb{E}_{q(\mathbf{z}^{\neg t})}[\delta(z_{t-1} = k)]}$$

CGS full conditional:

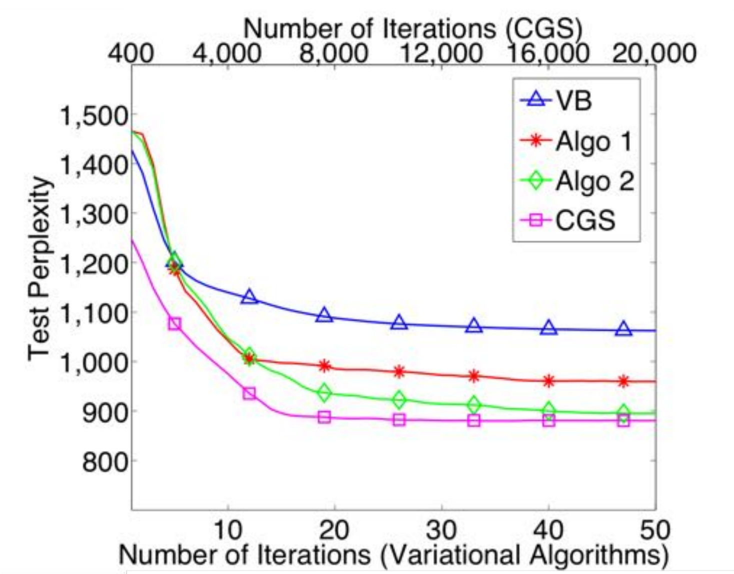
$$p(z_t = k|\mathbf{x}, \mathbf{z}^{\neg t}, \alpha, \beta) \propto \frac{C^{\neg t}_{k,w} + \beta}{C^{\neg t}_{k,\cdot} + W\beta} \cdot \frac{C^{\neg t}_{z_{t-1},k} + \alpha}{C^{\neg t}_{z_{t-1},\cdot} + K\alpha} \cdot \frac{C^{\neg t}_{k,z_{t+1}} + \alpha + \delta(z_{t-1} = k = z_{t+1})}{C^{\neg t}_{k,\cdot} + K\alpha + \delta(z_{t-1} = k)}$$

Figure from Wang & Blunsom (2013)

Unsupervised POS Tagging

Bayesian Inference for HMMs

- **Task:** unsupervised POS tagging
- **Data:** 1 million words (i.e. unlabeled sentences) of WSJ text
- **Dictionary:** defines legal part-of-speech (POS) tags for each word type
- **Models:**
  - EM: standard HMM
  - VB: uncollapsed variational Bayesian HMM
  - Algo 1 (CVB): collapsed variational Bayesian HMM (strong indep. assumption)
  - Algo 2 (CVB): collapsed variational Bayesian HMM (weaker indep. assumption)
  - CGS: collapsed Gibbs Sampler for Bayesian HMM



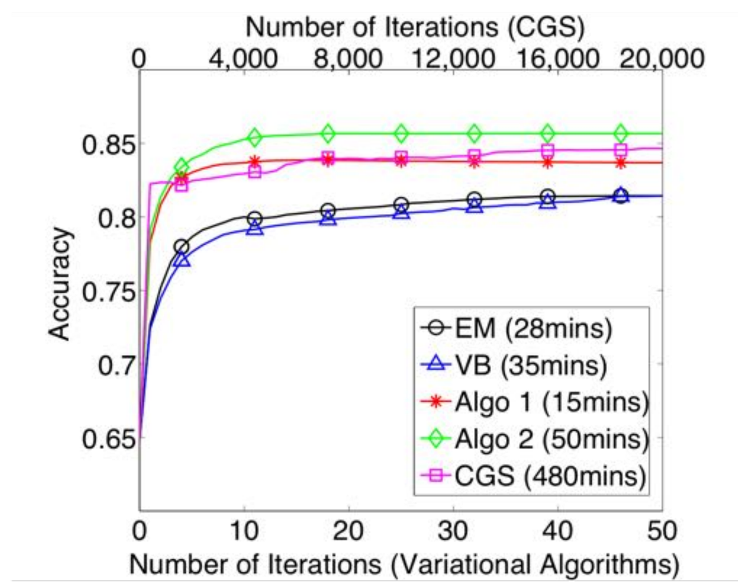


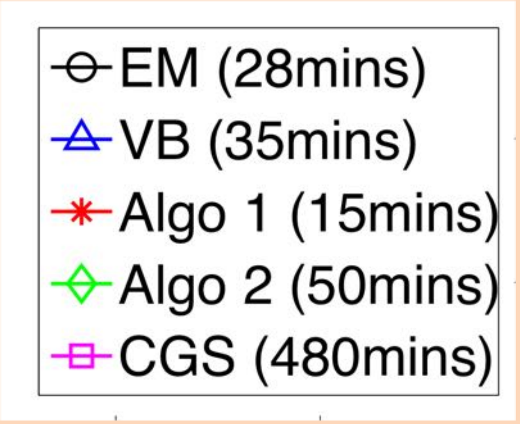
Figure from Wang & Blunsom (2013)

Unsupervised POS Tagging

Bayesian Inference for HMMs

- **Task:** unsupervised POS tagging
- **Data:** 1 million words (i.e. unlabeled sentences) of WSJ text
- **Dictionary:** defines legal part-of-speech (POS) tags for each word type
- **Models:**
  - EM: standard HMM
  - VB: uncollapsed variational Bayesian HMM
  - Algo 1 (CVB): collapsed variational Bayesian HMM (strong indep. assumption)
  - Algo 2 (CVB): collapsed variational Bayesian HMM (weaker indep. assumption)
  - CGS: collapsed Gibbs Sampler for Bayesian HMM

Speed:



- EM is slow b/c of log-space computations
- VB is slow b/c of digamma computations
- Algo 1 (CVB) is the fastest!
- Algo 2 (CVB) is slow b/c it computes dynamic parameters
- CGS: an order of magnitude slower than any deterministic algorithm

Figure from Wang & Blunsom (2013)

Stochastic Variational Bayesian HMM

- **Task:** Human Chromatin Segmentation
- **Goal:** unsupervised segmentation of the genome
- **Data:** from ENCODE, "250 million observations consisting of twelve assays carried out in the chronic myeloid leukemia cell line K562"
- **Metric:** "the false discovery rate (FDR) of predicting active promoter elements in the sequence"
- **Models:**
  - DBN HMM: dynamic Bayesian HMM trained with standard EM
  - SVIHMM: stochastic variational inference for a Bayesian HMM
- **Main Takeaway:**
  - the two models perform at similar levels of FDR
  - SVIHMM takes **one hour**
  - DBNHMM takes **days**

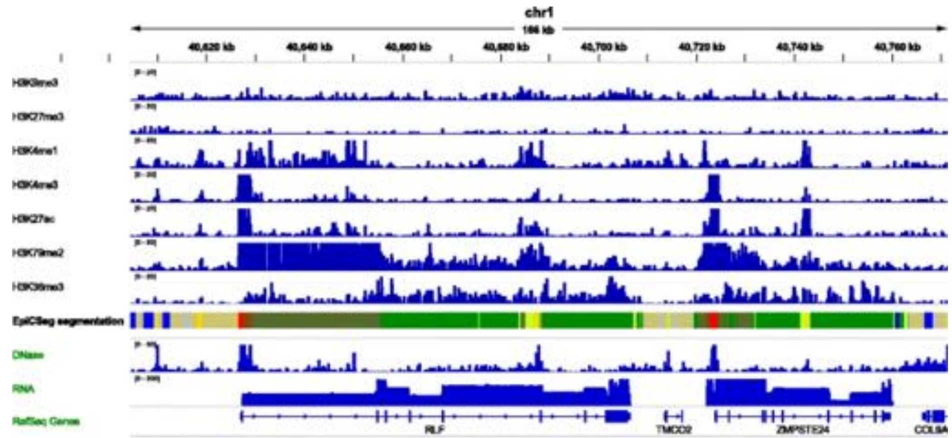


Figure from Foti et al. (2014)

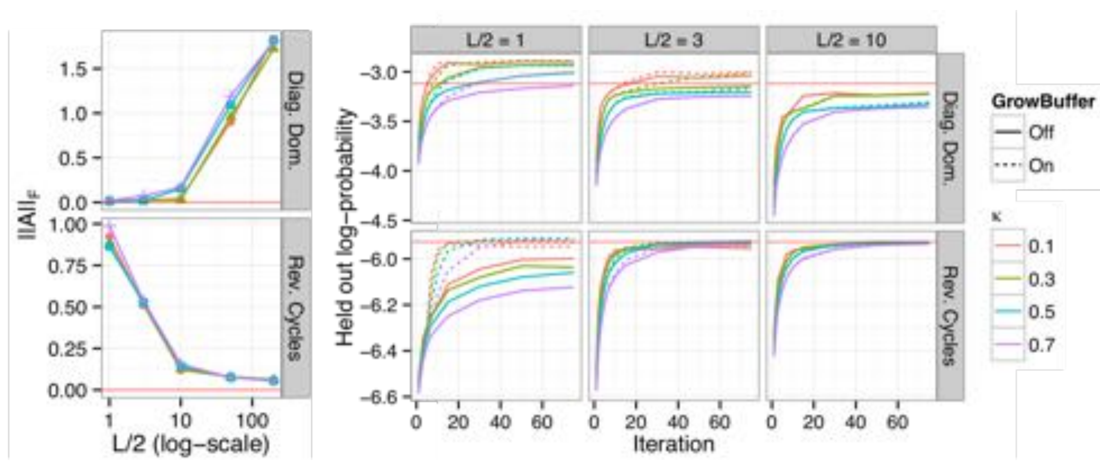


Figure from Mammana & Chung (2015)

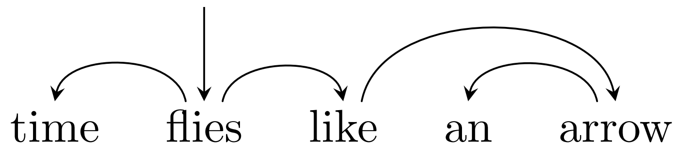
Grammar Induction

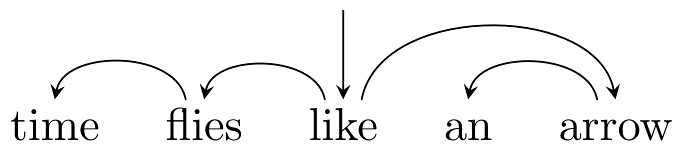
Question: Can maximizing (unsupervised) marginal likelihood produce useful results?

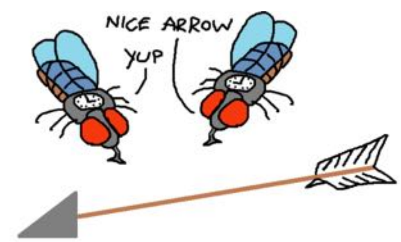
Answer: Let's look at an example…

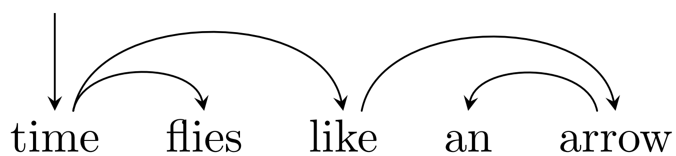
- **Babies** learn the syntax of their **native language** (e.g. English) just by **hearing** many sentences
- Can a **computer** similarly learn syntax of a **human language** just by looking at lots of example sentences?
  - This is the problem of Grammar Induction!
  - It's an unsupervised learning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

Grammar Induction

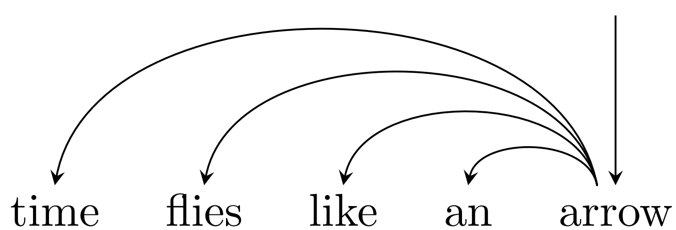








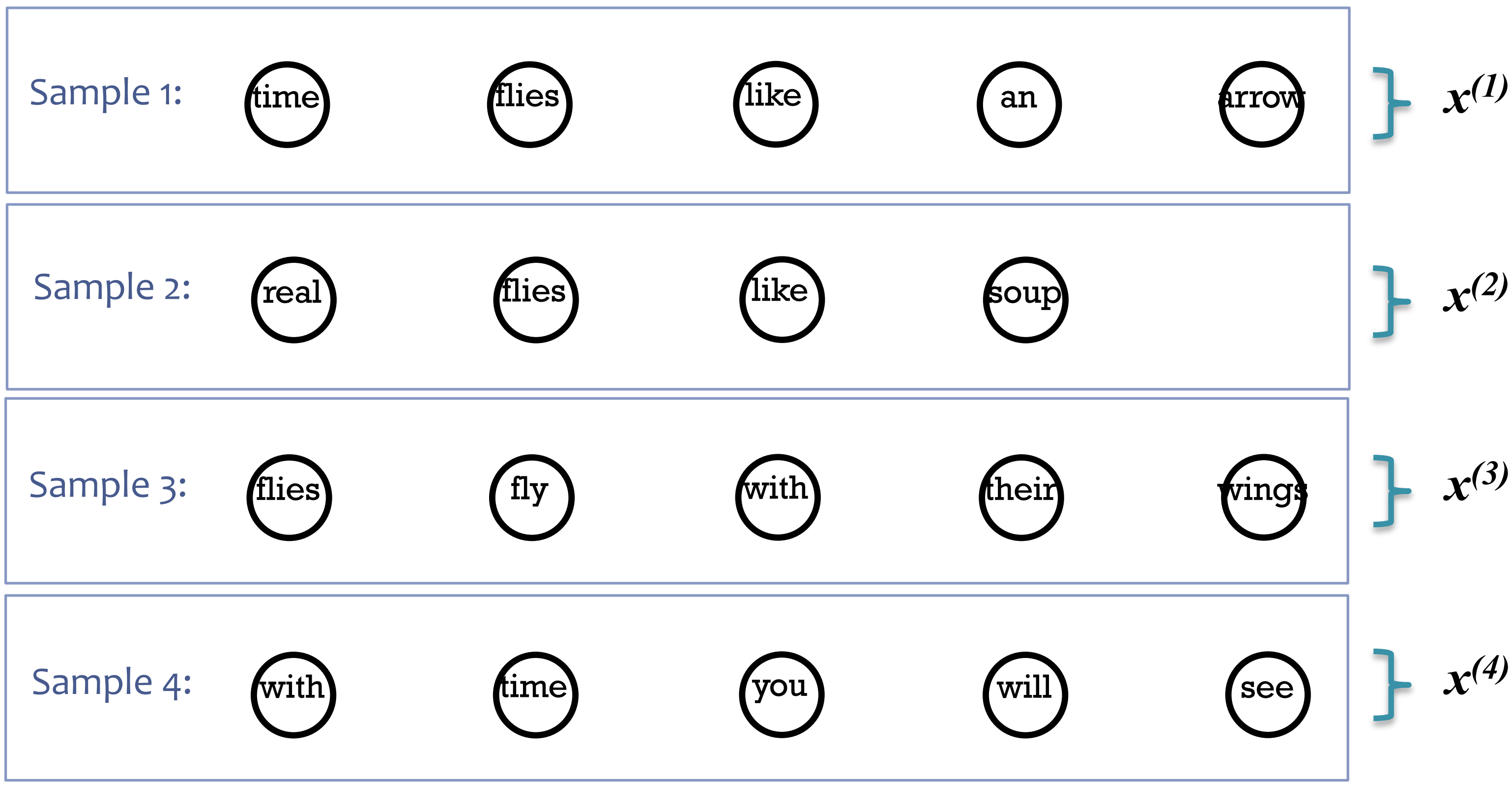
• • •



No semantic interpretation

Grammar Induction

Training Data: Sentences only, without parses



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

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)

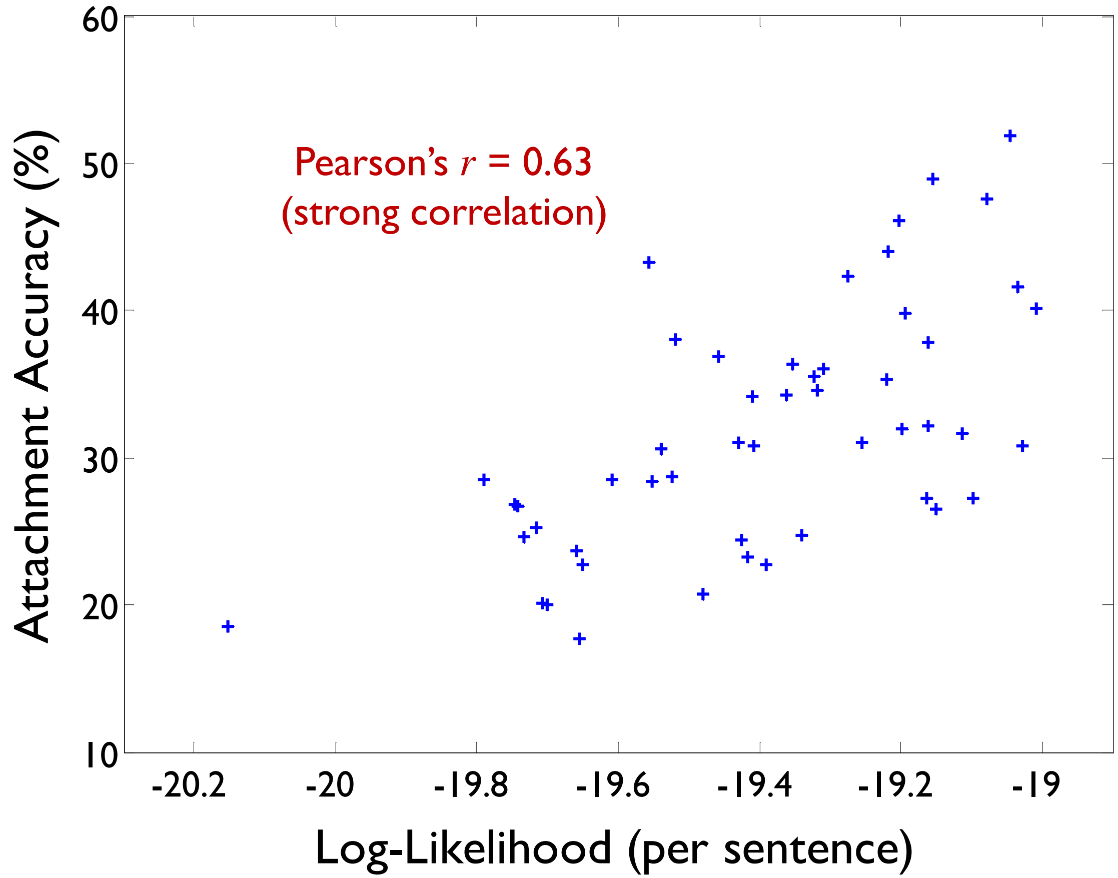
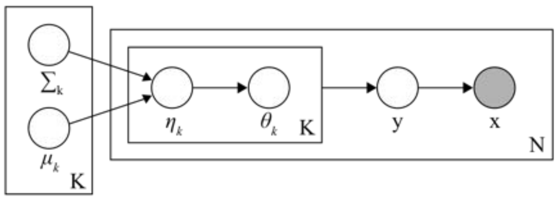


Figure from Gimpel & Smith (NAACL 2012) - slides

Figures from Cohen et al. (2009)

Grammar Induction

Graphical Model for Logistic Normal Probabilistic Grammar



y = syntactic parse
x = observed sentence

Settings:

EM Maximum likelihood estimate of $\boldsymbol{\theta}$ using the EM algorithm to optimize $p(\mathbf{x} \mid \boldsymbol{\theta})$ [14].

EM-MAP Maximum *a posteriori* estimate of $\boldsymbol{\theta}$ using the EM algorithm and a fixed symmetric Dirichlet prior with $\alpha > 1$ to optimize $p(\mathbf{x}, \boldsymbol{\theta} \mid \alpha)$. Tune α to maximize the likelihood of an unannotated development dataset, using grid search over $[1.1, 30]$.

VB-Dirichlet Use variational Bayes inference to estimate the posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{x}, \alpha)$, which is a Dirichlet. Tune the symmetric Dirichlet prior's parameter α to maximize the likelihood of an unannotated development dataset, using grid search over $[0.0001, 30]$. Use the mean of the posterior Dirichlet as a point estimate for $\boldsymbol{\theta}$.

VB-EM-Dirichlet Use variational Bayes EM to optimize $p(\mathbf{x} \mid \boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$. Use the mean of the learned Dirichlet as a point estimate for $\boldsymbol{\theta}$ (similar to [5]).

VB-EM-Log-Normal Use variational Bayes EM to optimize $p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Use the (exponentiated) mean of this Gaussian as a point estimate for $\boldsymbol{\theta}$.

Results:

	attachment accuracy (%)					
	Viterbi decoding			MBR decoding		
	$\lvert\mathbf{x}\rvert \le 10$	$\lvert\mathbf{x}\rvert \le 20$	all	$\lvert\mathbf{x}\rvert \le 10$	$\lvert\mathbf{x}\rvert \le 20$	all
Attach-Right	38.4	33.4	31.7	38.4	33.4	31.7
EM	45.8	39.1	34.2	46.1	39.9	35.9
EM-MAP, $\boldsymbol{\alpha} = 1.1$	45.9	39.5	34.9	46.2	40.6	36.7
VB-Dirichlet, $\boldsymbol{\alpha} = 0.25$	46.9	40.0	35.7	47.1	41.1	37.6
VB-EM-Dirichlet	45.9	39.4	34.9	46.1	40.6	36.9
VB-EM-Log-Normal, $\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}$	56.6	43.3	37.4	59.1	**45.9**	39.9
VB-EM-Log-Normal, families	**59.3**	**45.1**	**39.0**	**59.4**	**45.9**	**40.5**

Table 1: Attachment accuracy of different learning methods on unseen test data from the Penn Treebank of varying levels of difficulty imposed through a length filter. Attach-Right attaches each word to the word on its right and the last word to \$. EM and EM-MAP with a Dirichlet prior ($\alpha > 1$) are reproductions of earlier results [14, 18].