FlowVB: Automatic determination of the number of mixture components in Flow Cytometry with Variational Bayes

H. Bretschneider[1,2] A. Roth[3]

[1]Department of Statistics, University of British Columbia

[2]School of Business and Economics, Humboldt-Universität zu Berlin

[2]Department Of Molecular Biology and Biochemistry, Simon Fraser University

FlowCAP Summit 2010
September 21, 2010, Bethesda, MD

Motivation: GMMs for clustering

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Advantages:

Motivation: GMMs for clustering

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Advantages:

- Probabilistic method (soft clustering)

Motivation: GMMs for clustering

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Advantages:

- Probabilistic method (soft clustering)
- Aware of covariance-structure

Motivation: GMMs for clustering

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Advantages:

- Probabilistic method (soft clustering)
- Aware of covariance-structure
- Mathematically convenient

Motivation: GMMs for clustering

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Advantages:

- Probabilistic method (soft clustering)
- Aware of covariance-structure
- Mathematically convenient
- Can be fit (relatively) easily with the *Expectation-Maximization* (EM) Algorithm

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Disadvantages:

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Disadvantages:

1. Sensitivity to noise (outliers)

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Disadvantages:

1. Sensitivity to noise (outliers)
2. Number of mixture components must be pre-specified

Starting point:

Gaussian Mixture Models (GMMS) are a popular method for clustering data

Disadvantages:

1. Sensitivity to noise (outliers)
2. Number of mixture components must be pre-specified

We are trying to solve these two problems.

Problem 1: Sensitivity to noise

$\Rightarrow$ **Use Student-t Mixture Model (SMM) instead**

Problem 1: Sensitivity to noise

$\Rightarrow$ **Use Student-t Mixture Model (SMM) instead**

- More heavy-tailed, less sensitive to outliers

Problem 1: Sensitivity to noise

$\Rightarrow$ **Use Student-t Mixture Model (SMM) instead**

- More heavy-tailed, less sensitive to outliers
- Can also be fit using the EM-algorithm

Problem 1: Sensitivity to noise

$\Rightarrow$ **Use Student-t Mixture Model (SMM) instead**

- More heavy-tailed, less sensitive to outliers
- Can also be fit using the EM-algorithm

But:

Closed form solutions don't exist for all parameters of the distribution

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

- The likelihood never decreases when increasing the number of mixture components M

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

- The likelihood never decreases when increasing the number of mixture components M
- Therefore optimizing over M is an degenerate (ill-posed) problem

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Solution 1: Cross-validation

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Solution 1: Cross-validation

- Requires that the algorithm is run many times

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Solution 1: Cross-validation

- Requires that the algorithm is run many times
- Computationally expensive

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Solution 1: Cross-validation

- Requires that the algorithm is run many times
- Computationally expensive

Solution 2: Variational Bayesian (VB) Inference

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Solution 1: Cross-validation

- Requires that the algorithm is run many times
- Computationally expensive

Solution 2: Variational Bayesian (VB) Inference

- Can determine the number of components in one pass

Problem 2: Finding the number of components

Maximum-likelihood cannot find the number of mixture components.

Solution 1: Cross-validation

- Requires that the algorithm is run many times
- Computationally expensive

Solution 2: Variational Bayesian (VB) Inference

- Can determine the number of components in one pass

Task

Develop an algorithm to fit SMMs using Variational Bayes

Demo - Robustness to noise

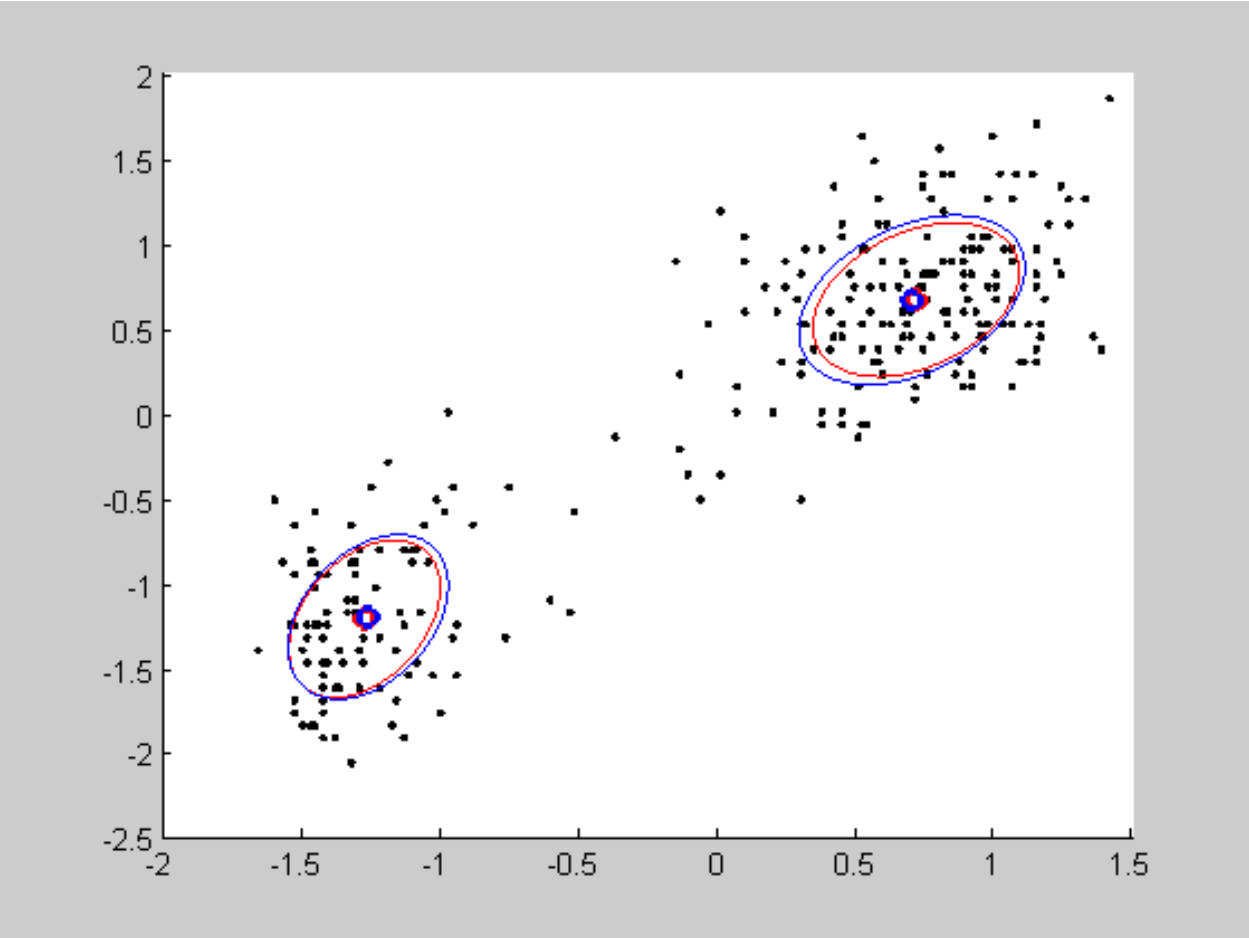


Figure: GMM (blue) vs. SMM(red) fit with VB without added noise

Demo - Robustness to noise

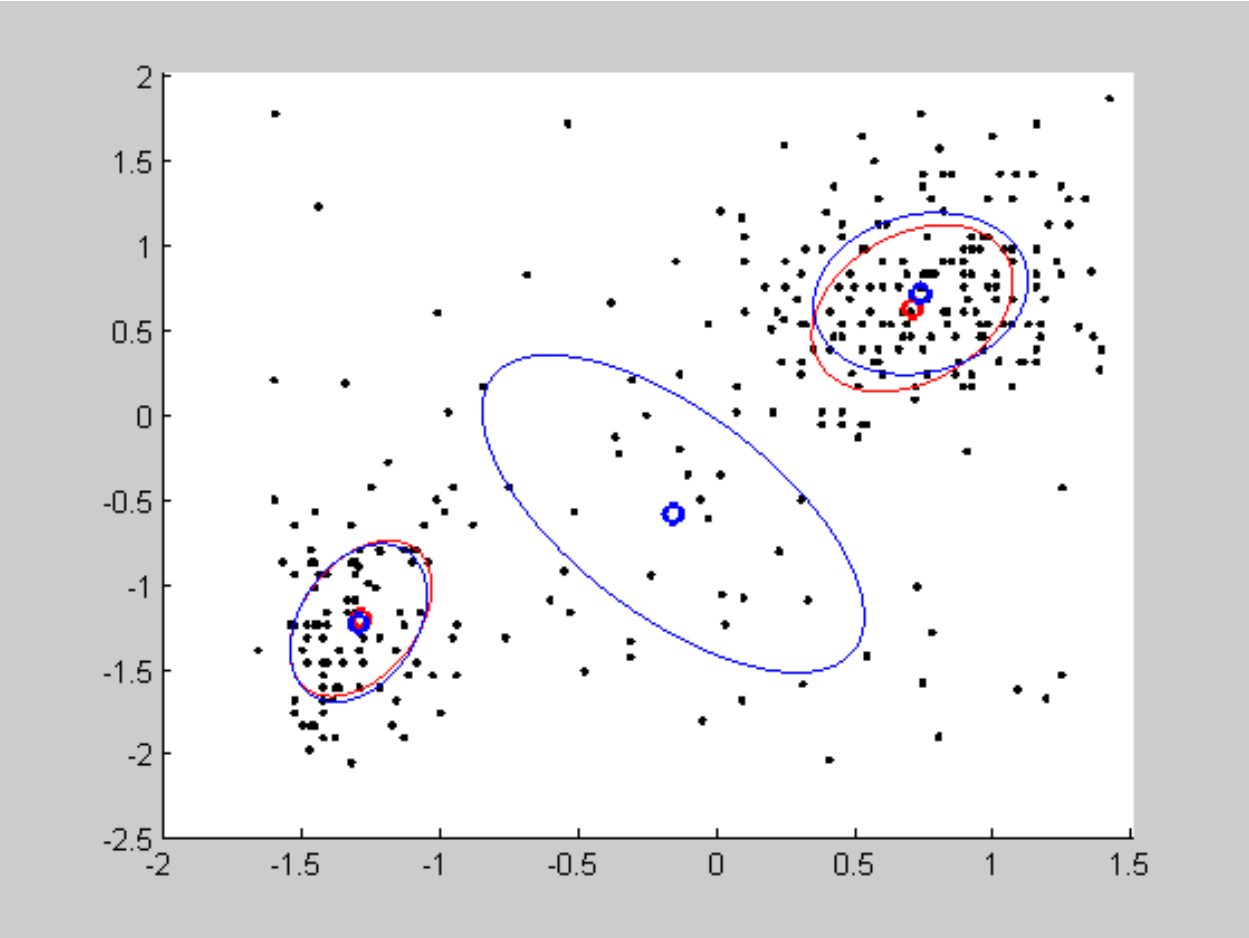
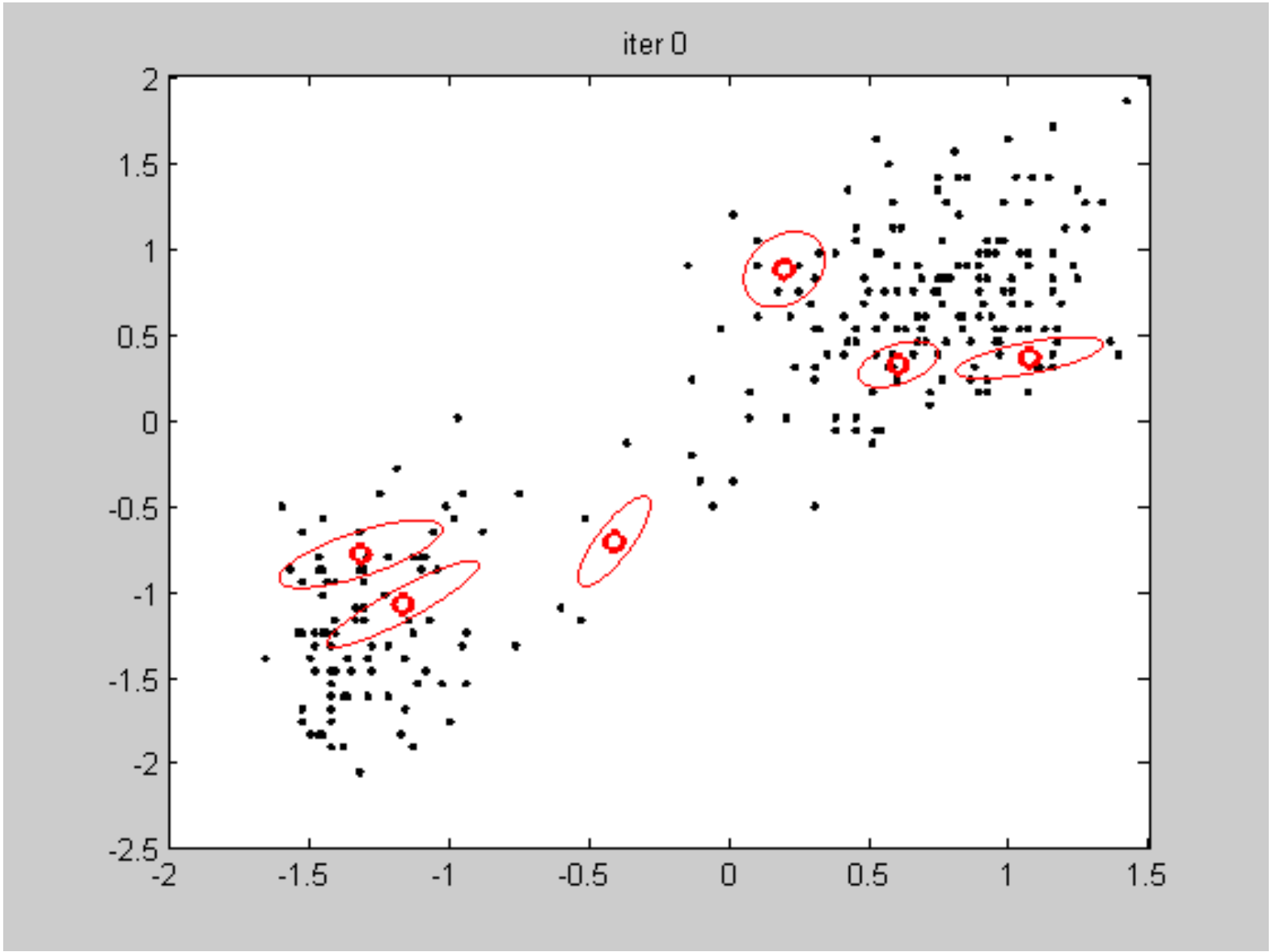
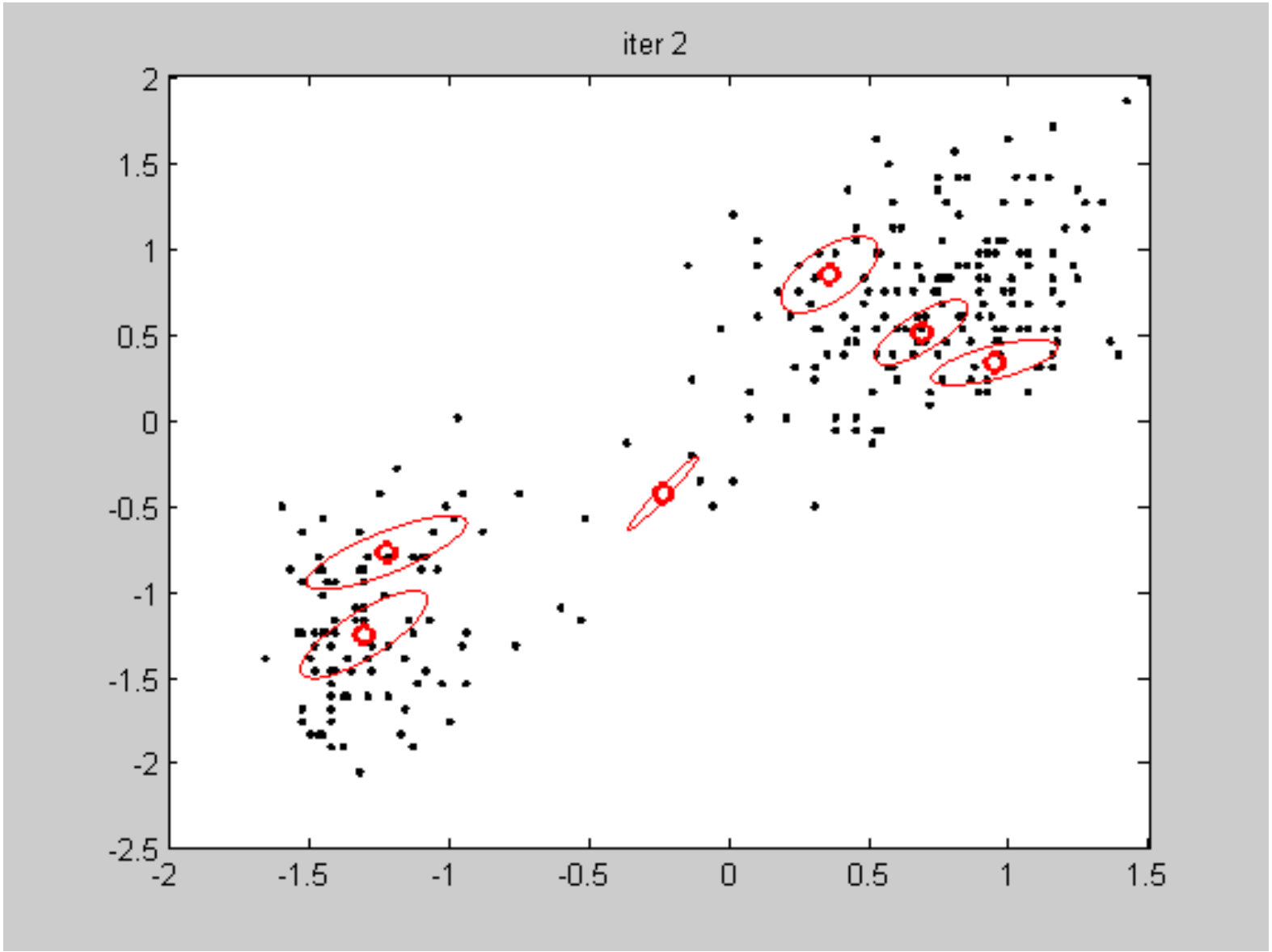


Figure: GMM (blue) vs. SMM(red) fit with VB with 20% added noise

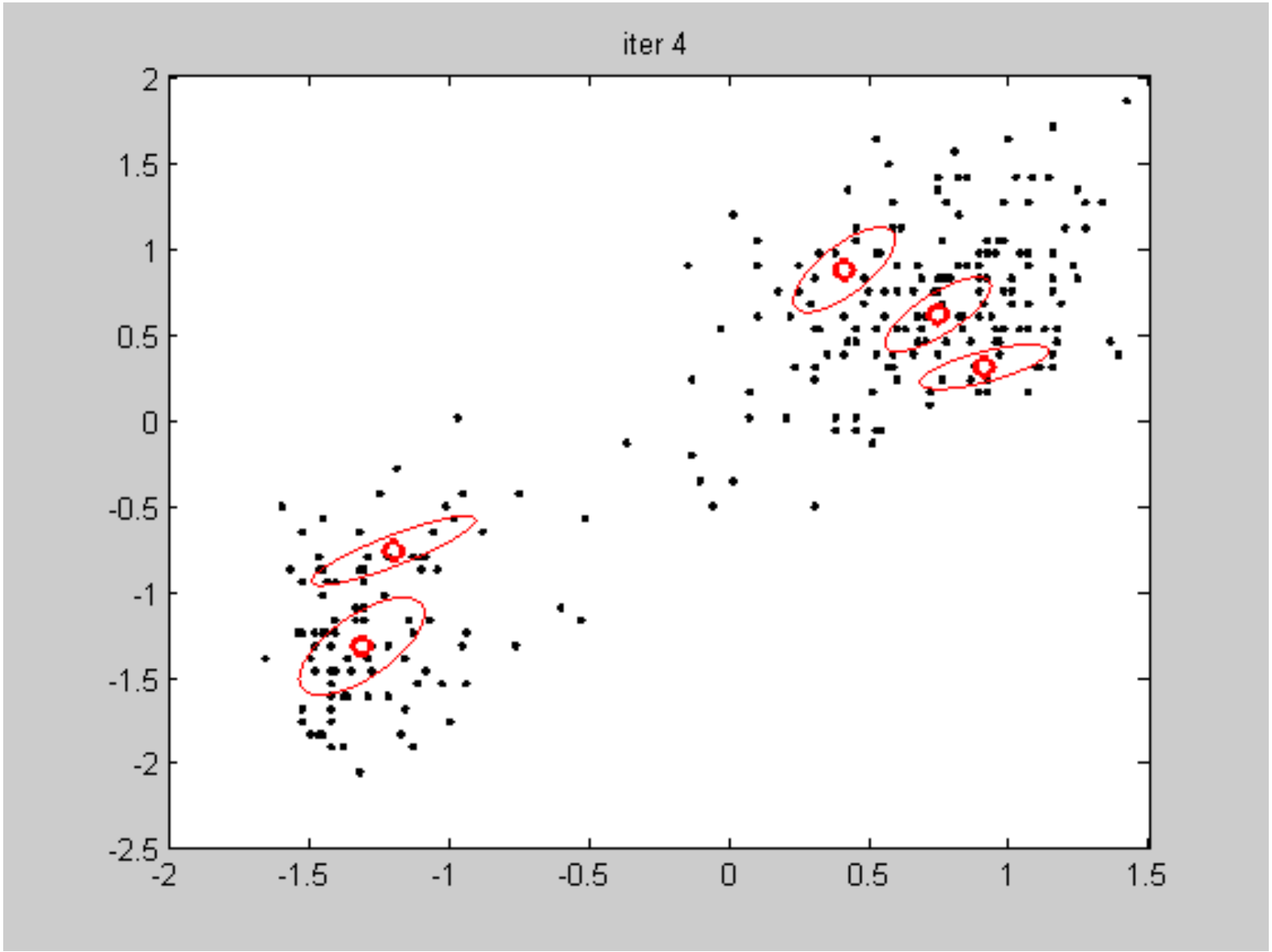
Demo - Automatic pruning of components



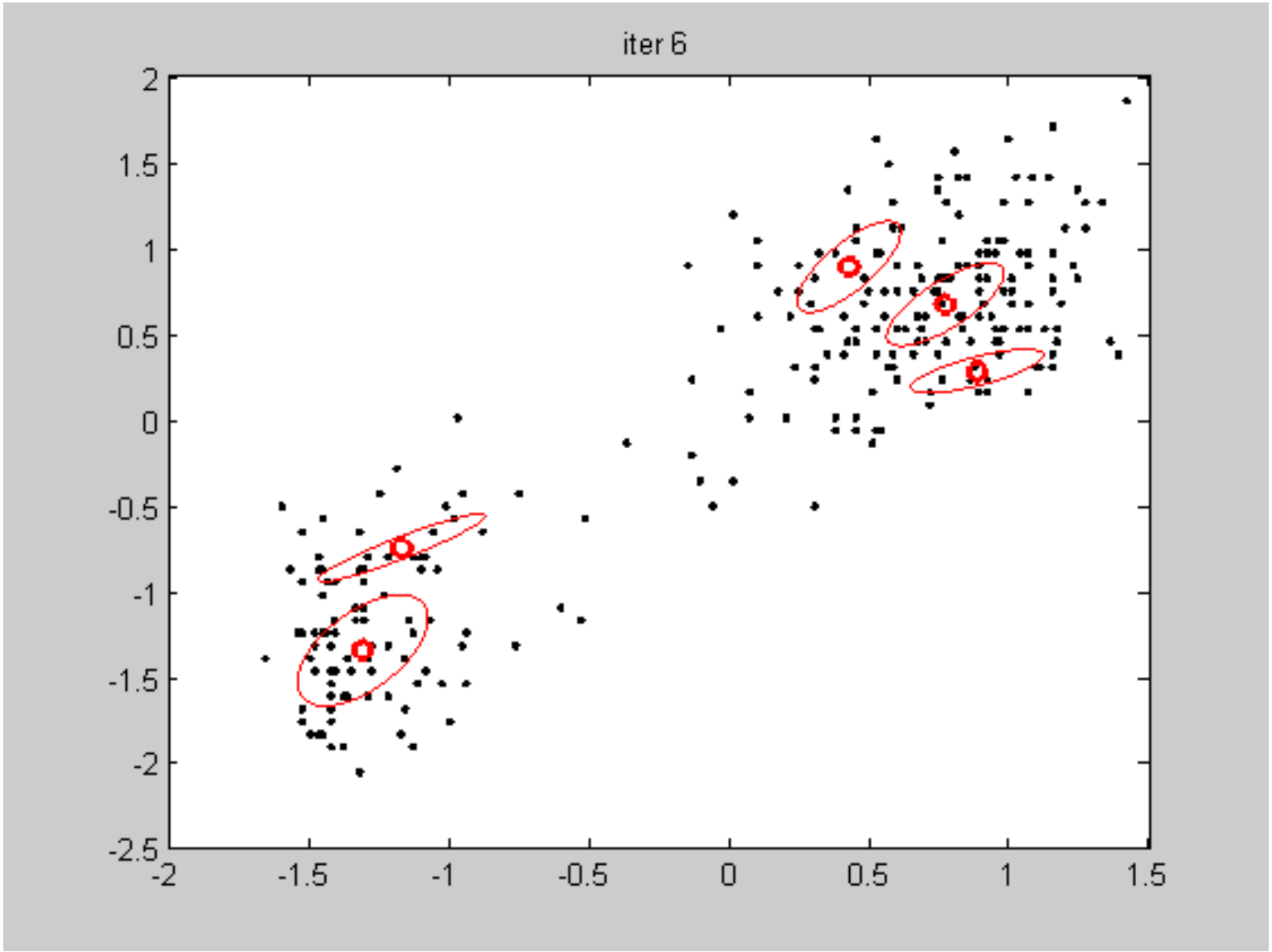
Demo - Automatic pruning of components



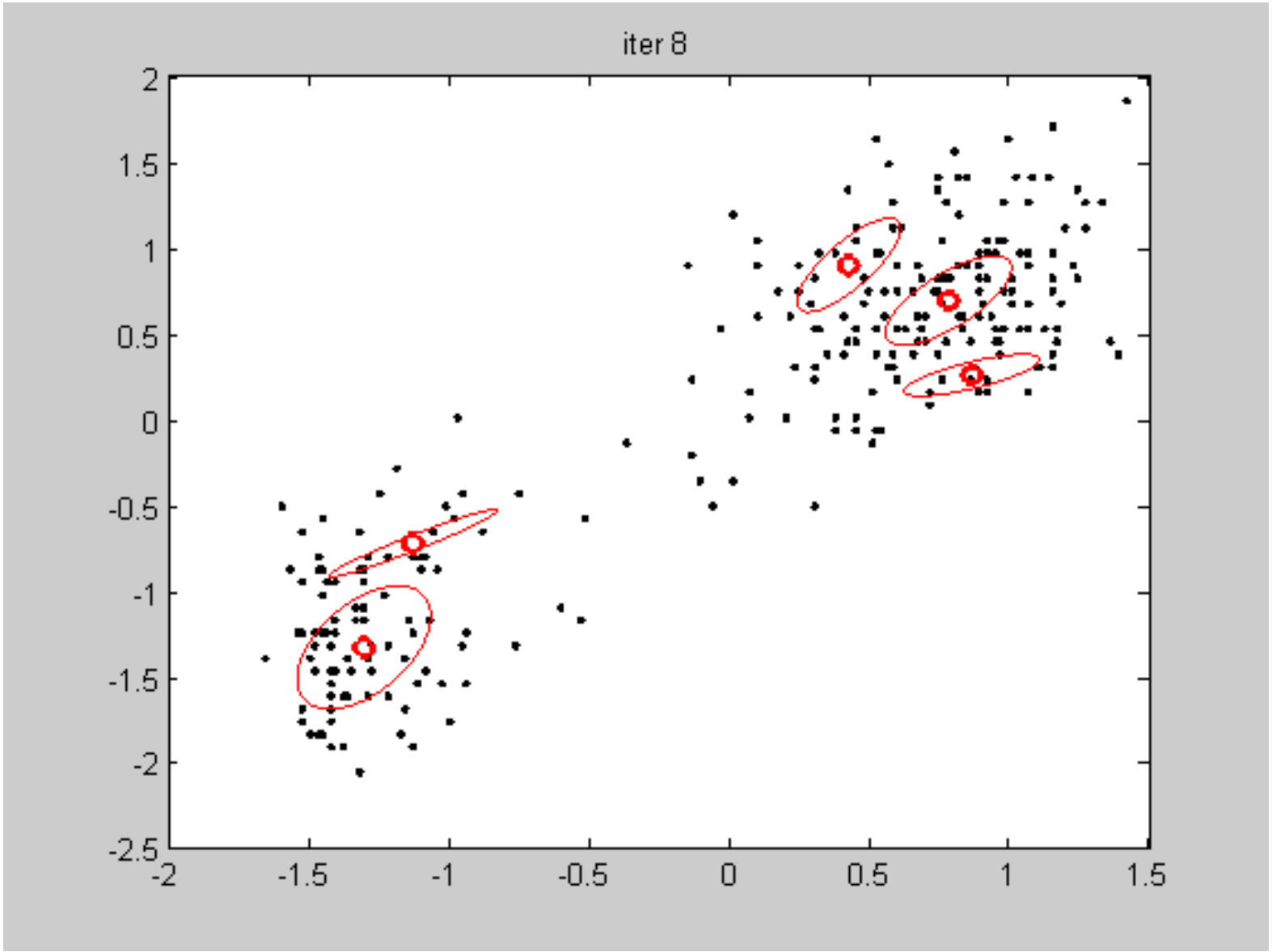
Demo - Automatic pruning of components



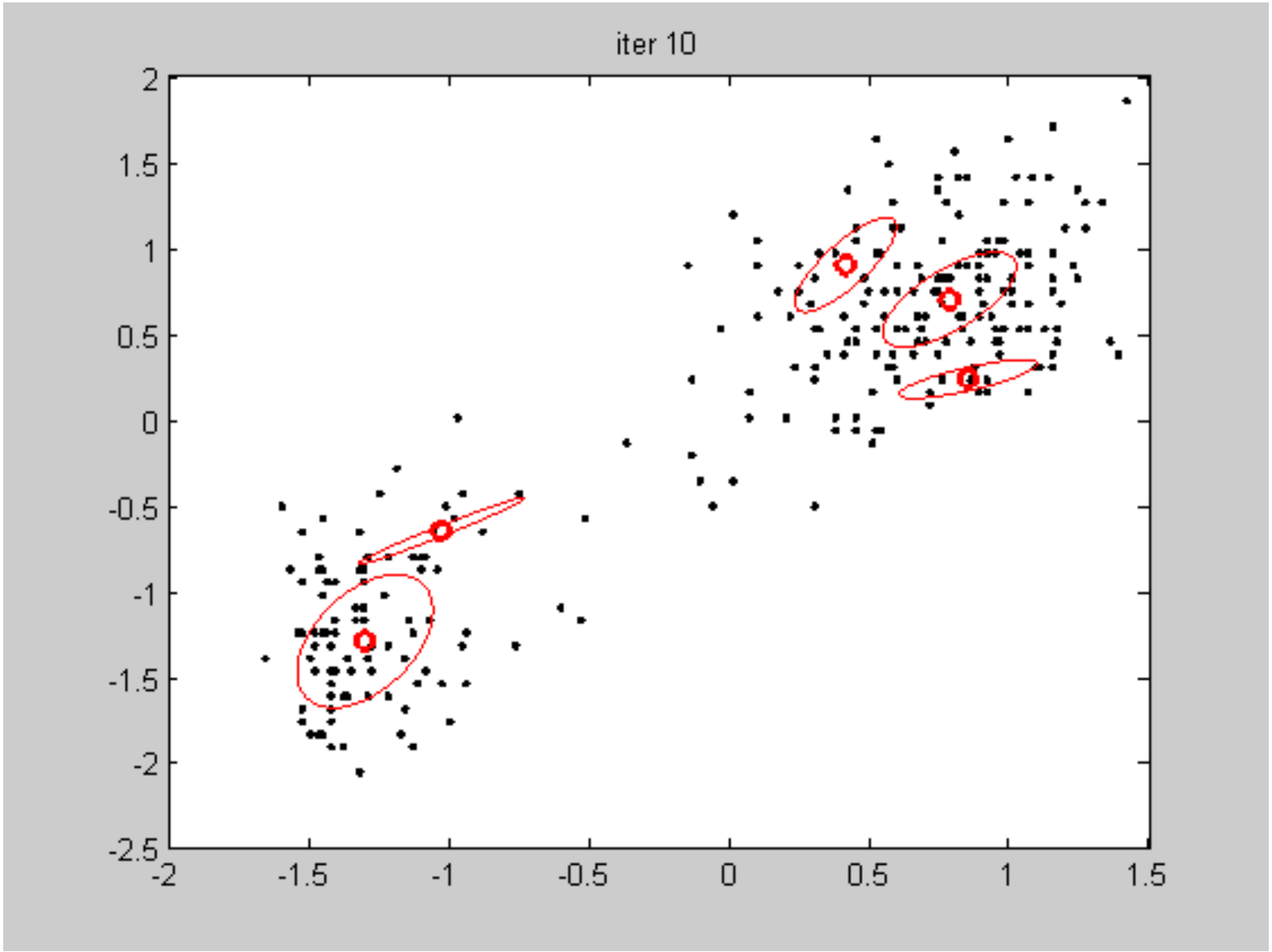
Demo - Automatic pruning of components



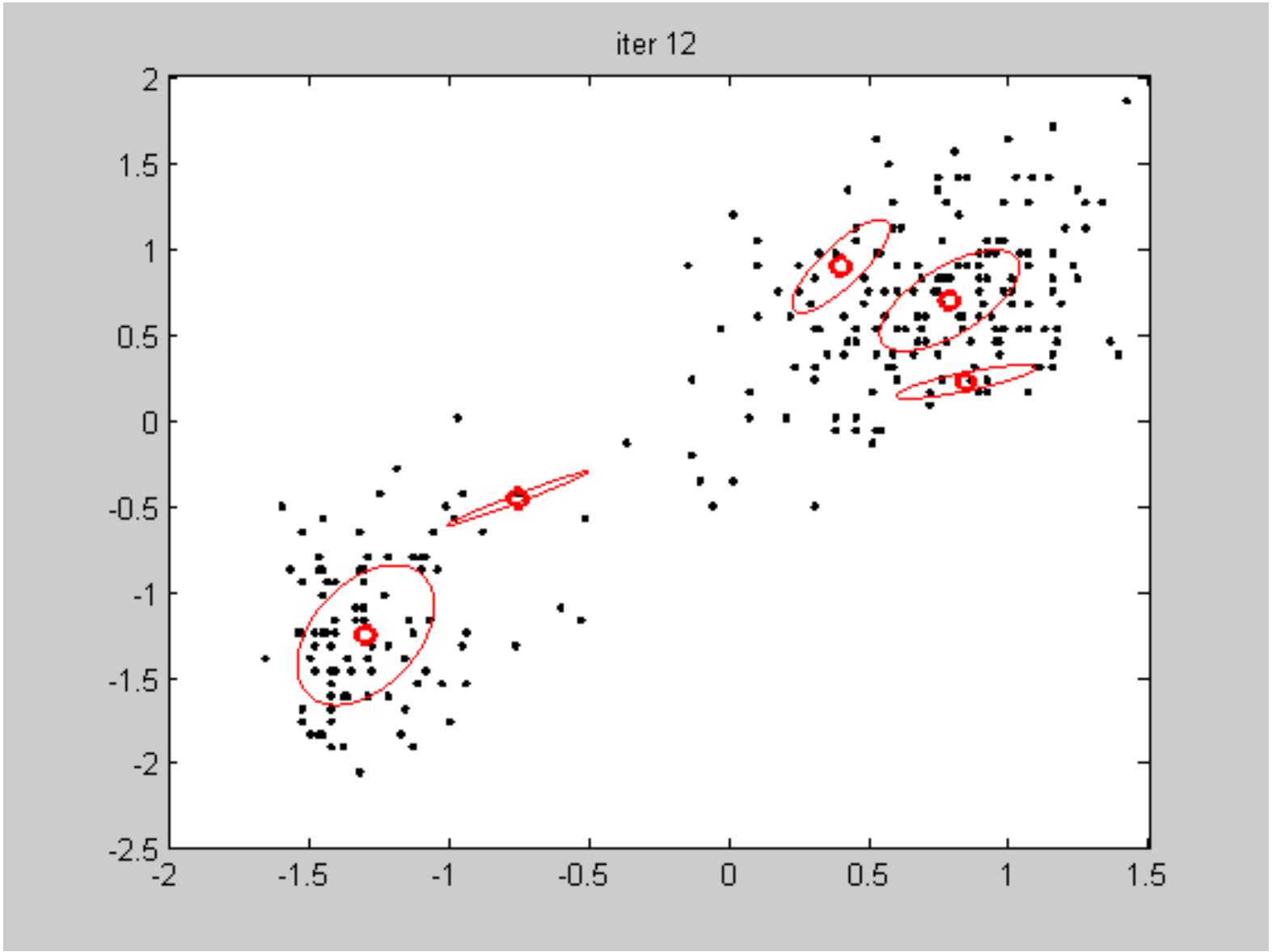
Demo - Automatic pruning of components



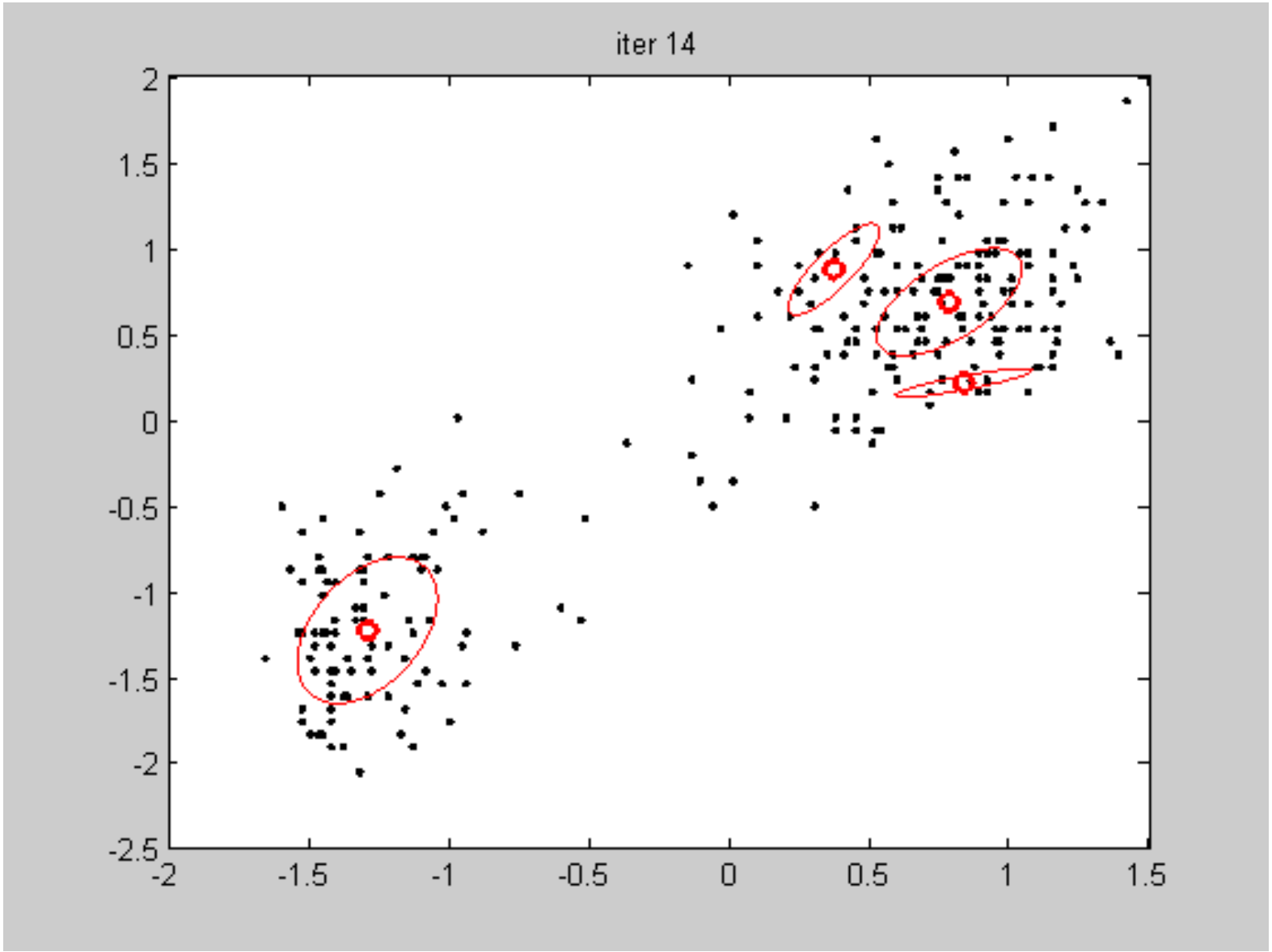
Demo - Automatic pruning of components



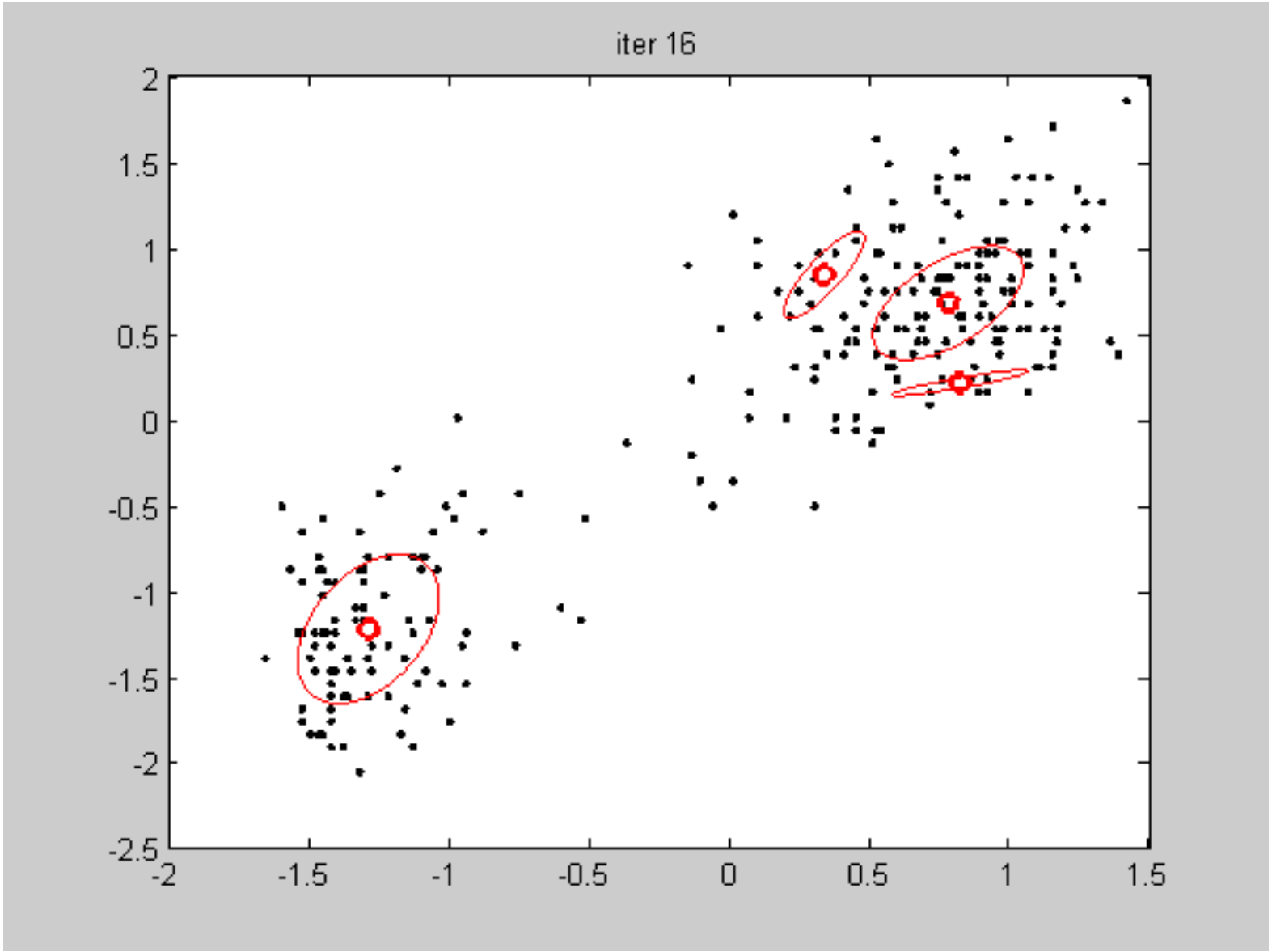
Demo - Automatic pruning of components



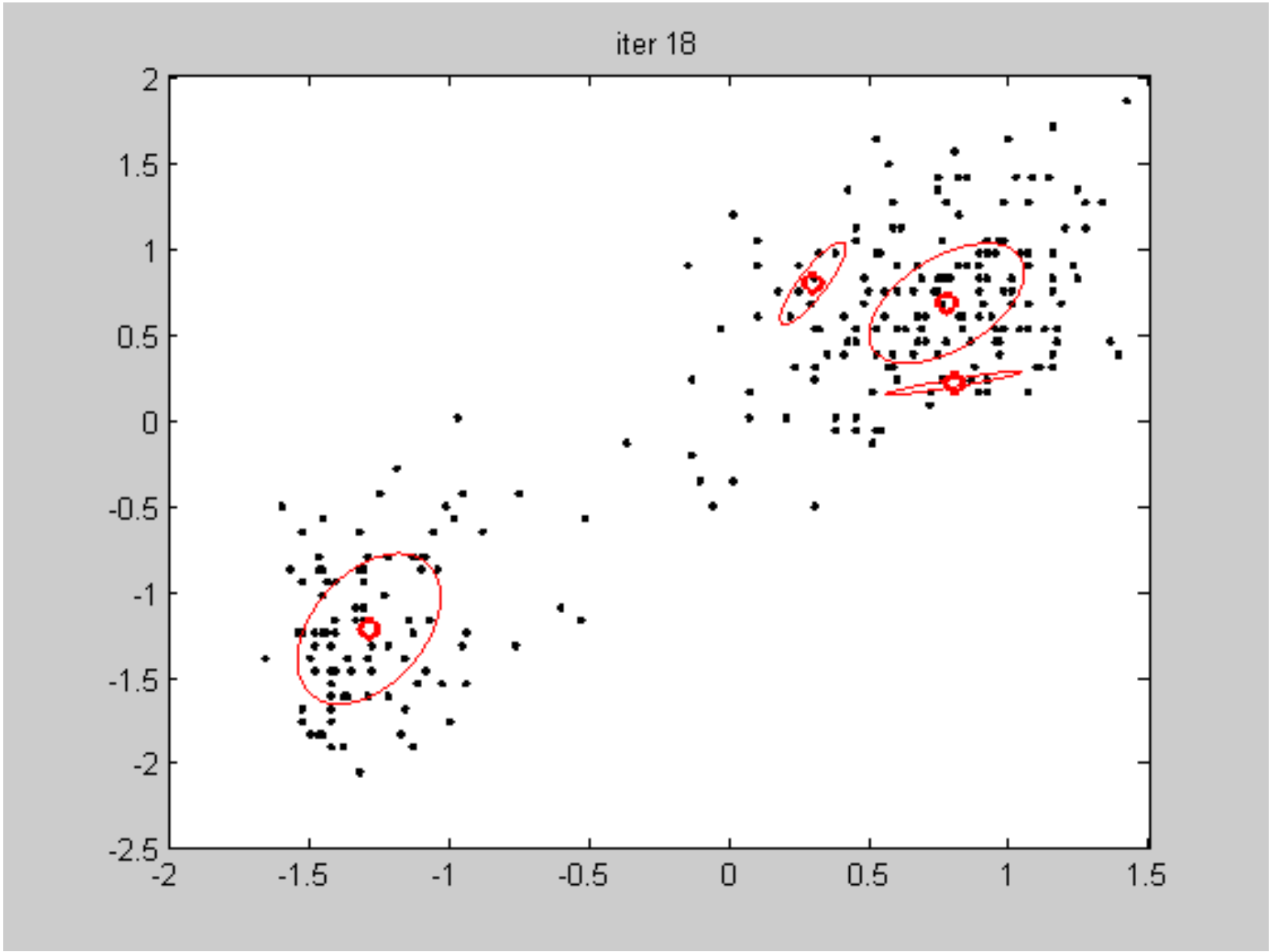
Demo - Automatic pruning of components



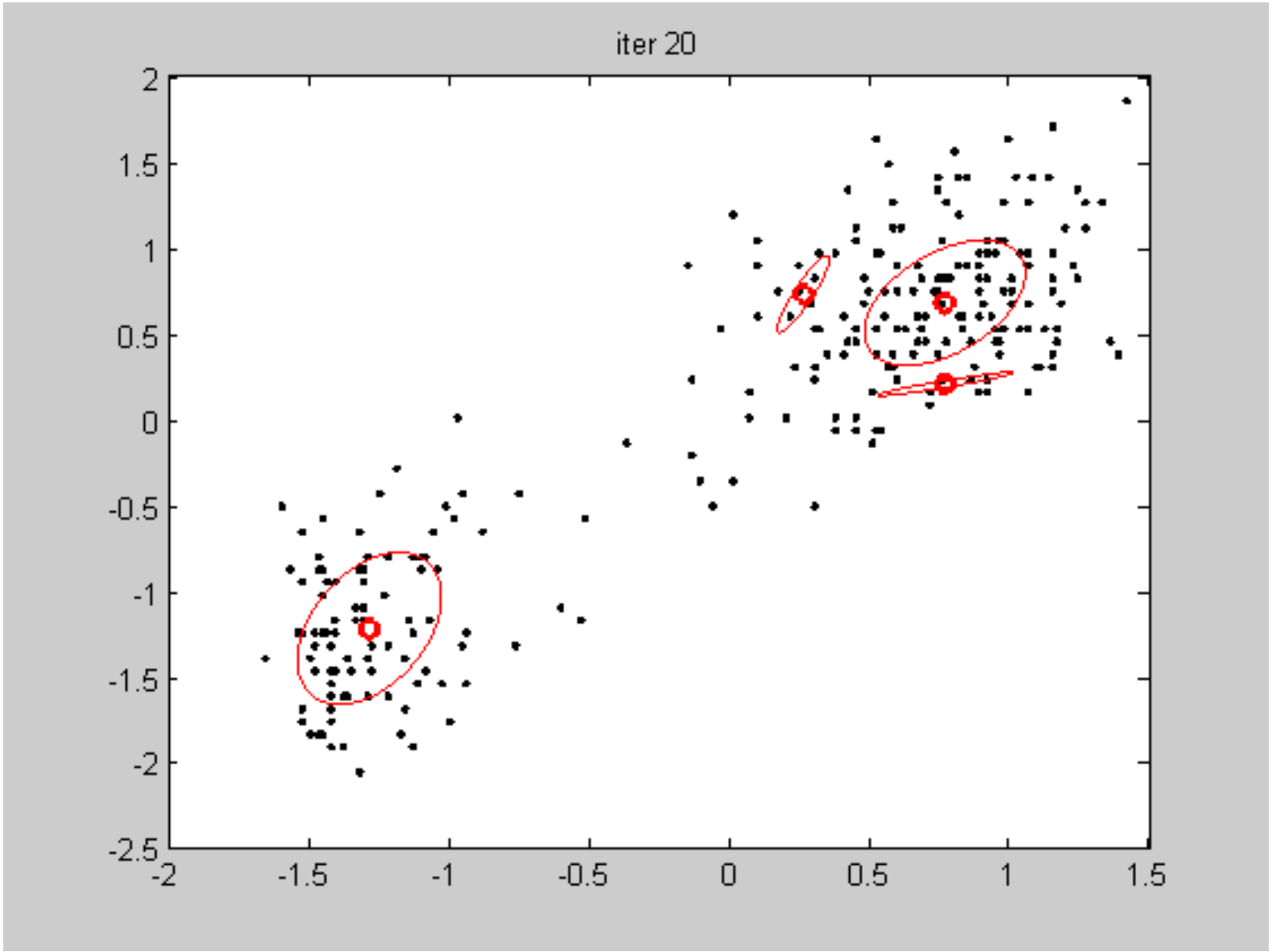
Demo - Automatic pruning of components



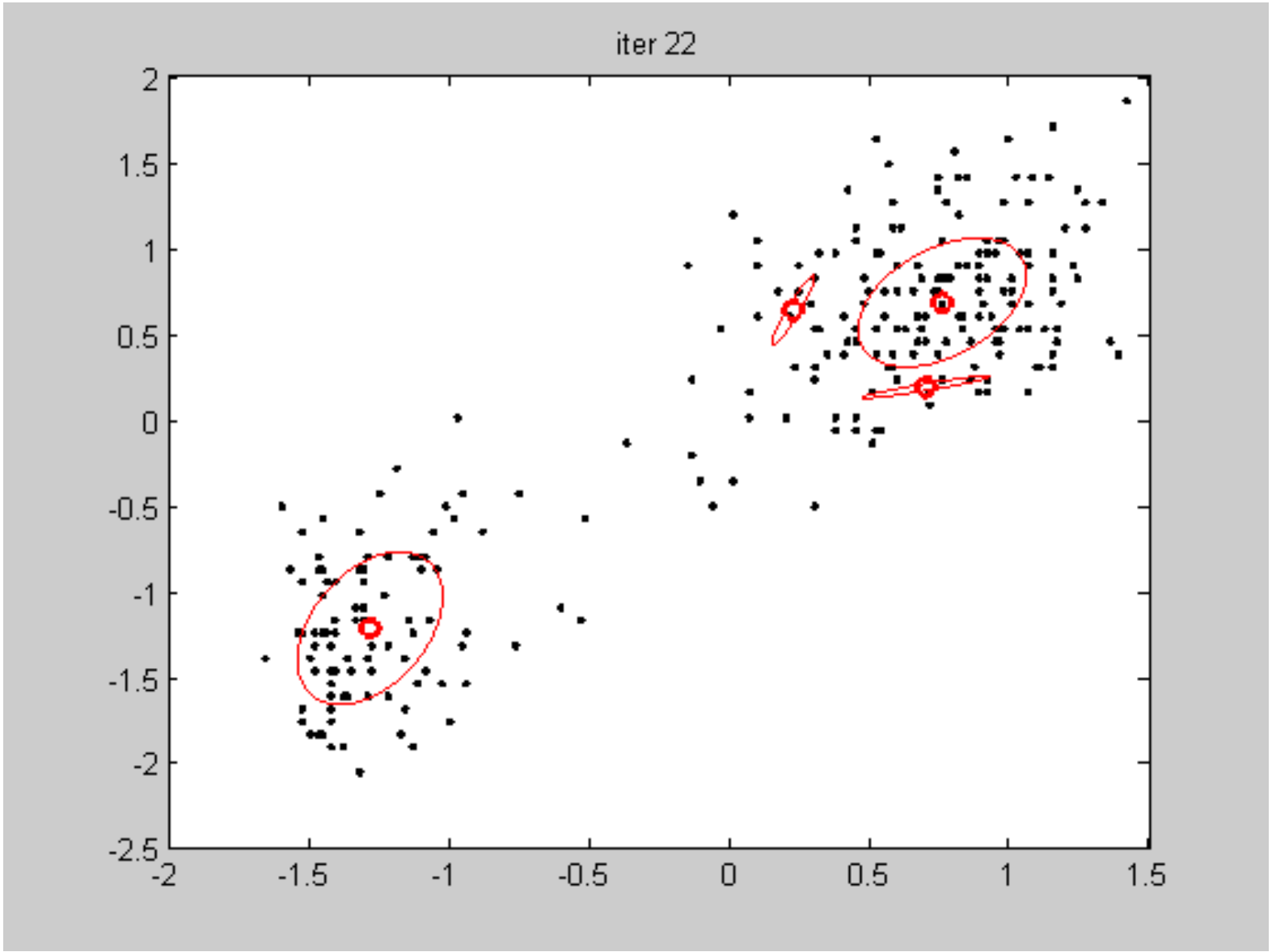
Demo - Automatic pruning of components



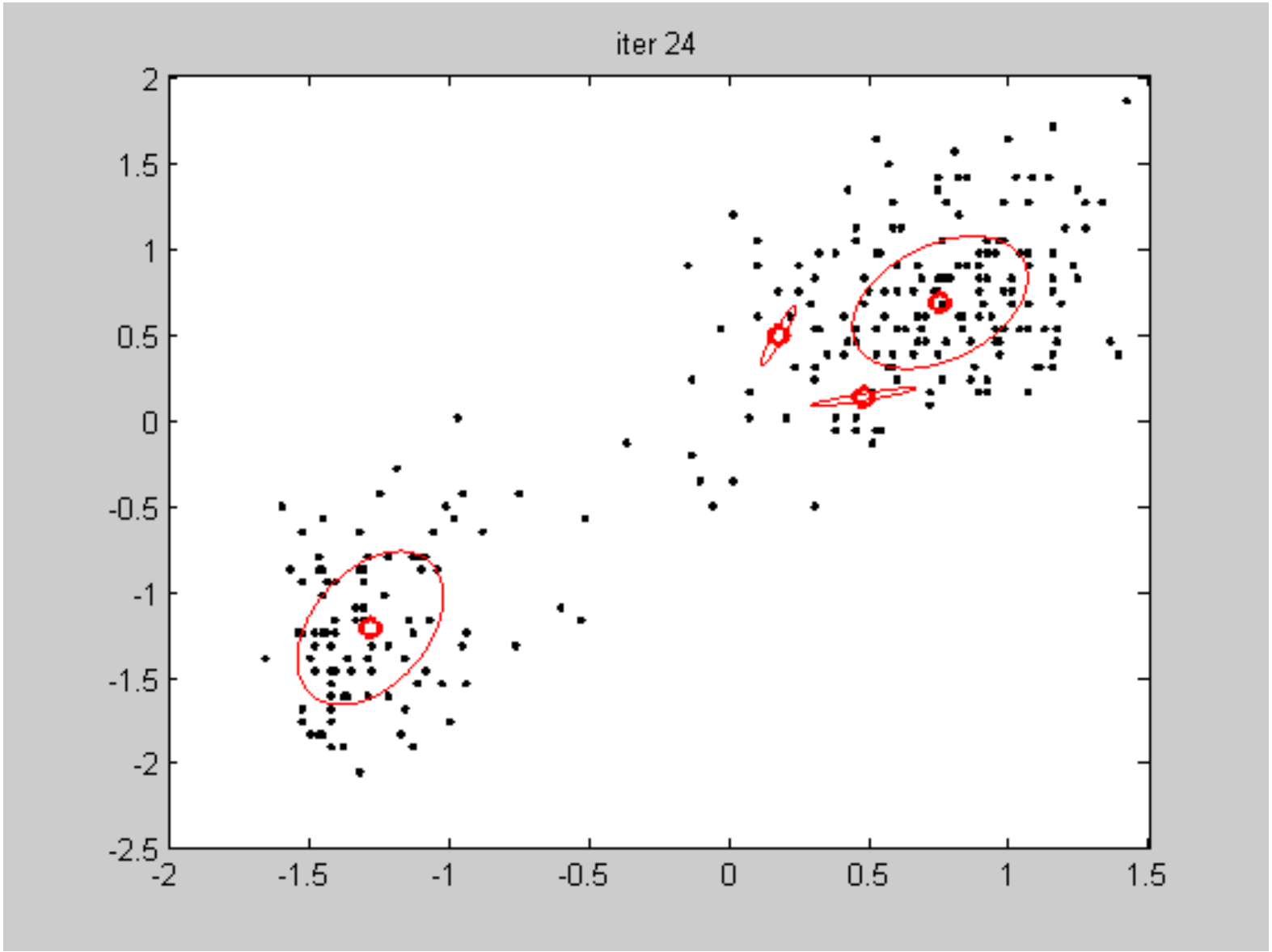
Demo - Automatic pruning of components



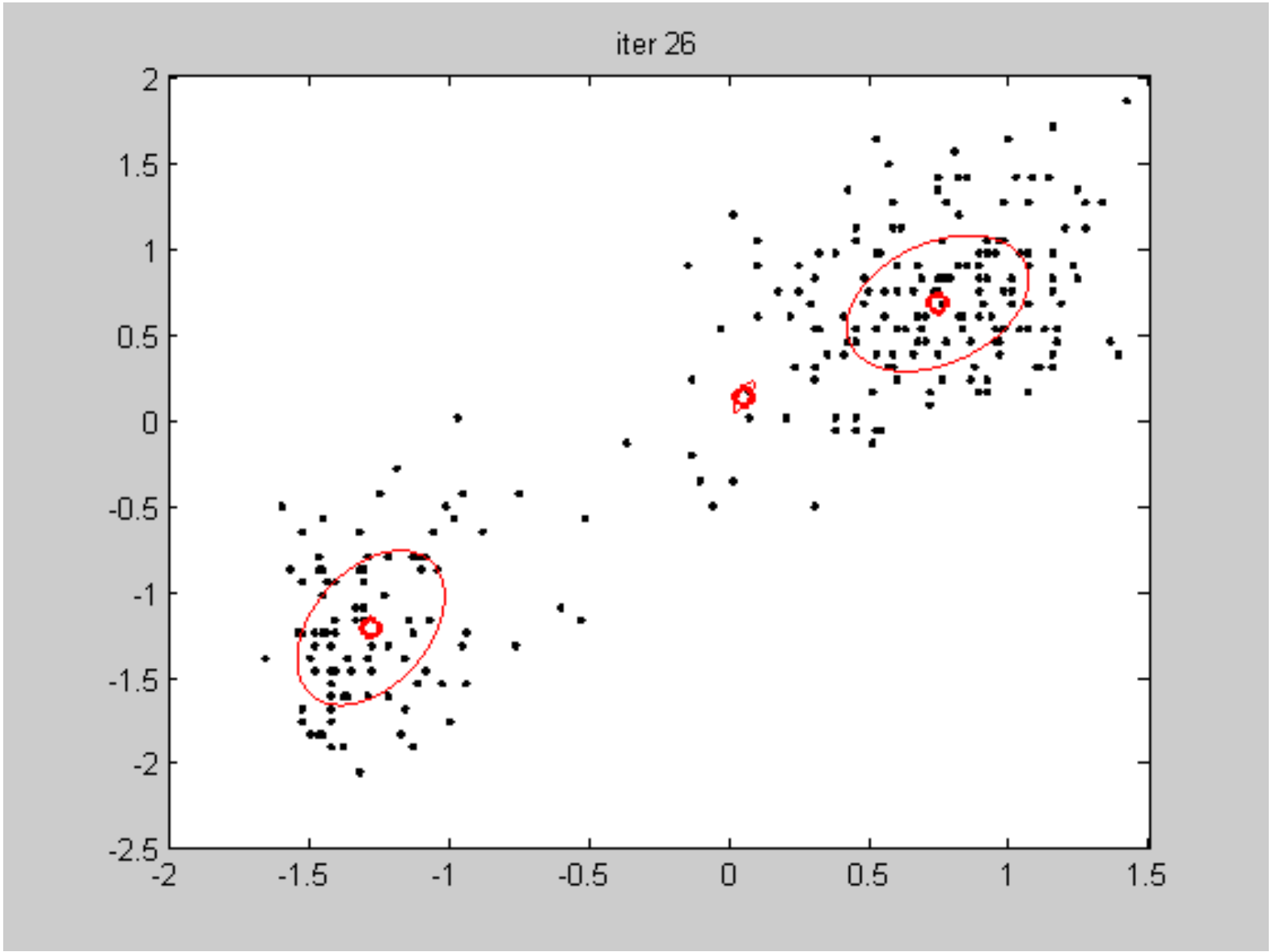
Demo - Automatic pruning of components



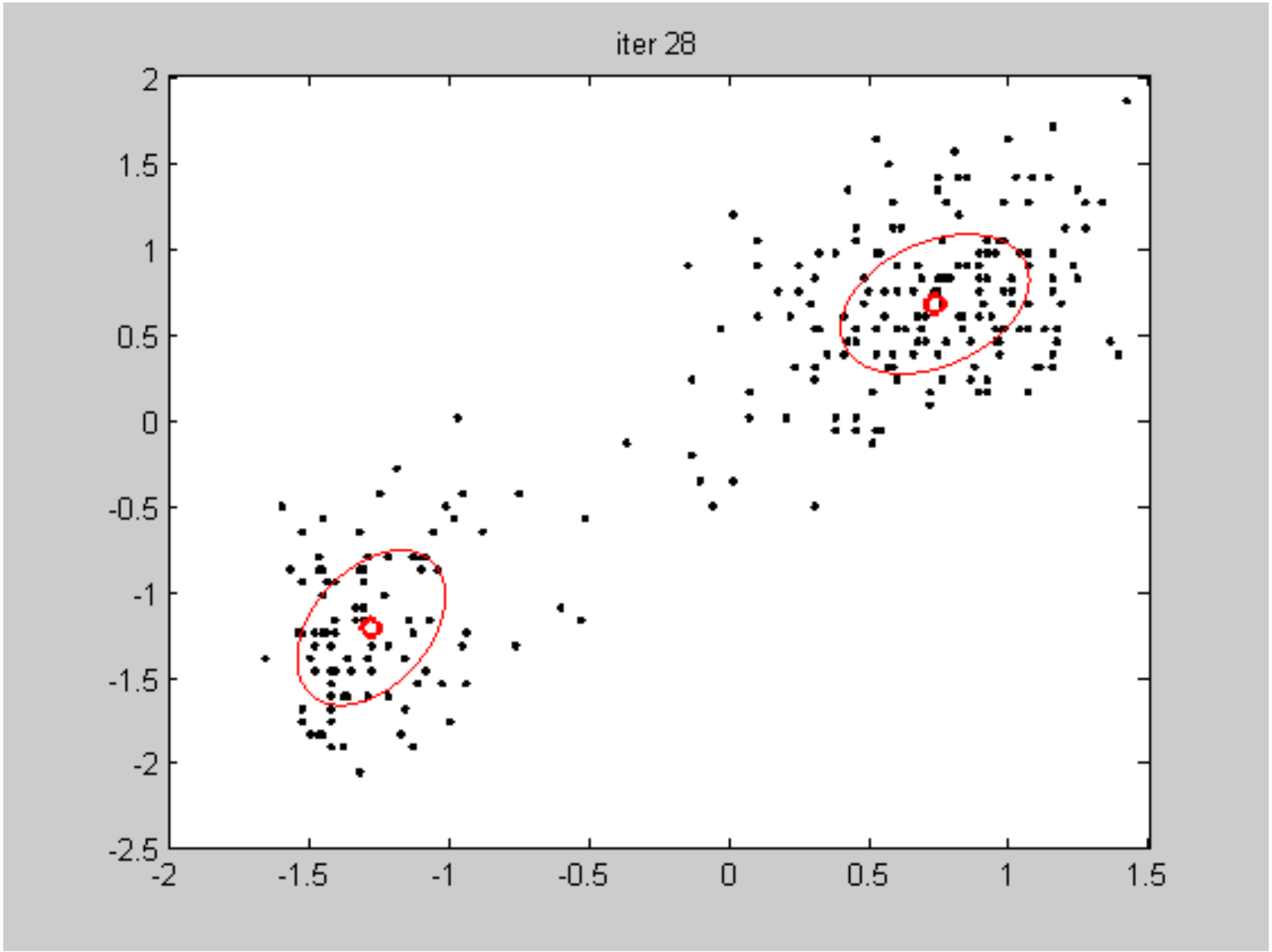
Demo - Automatic pruning of components



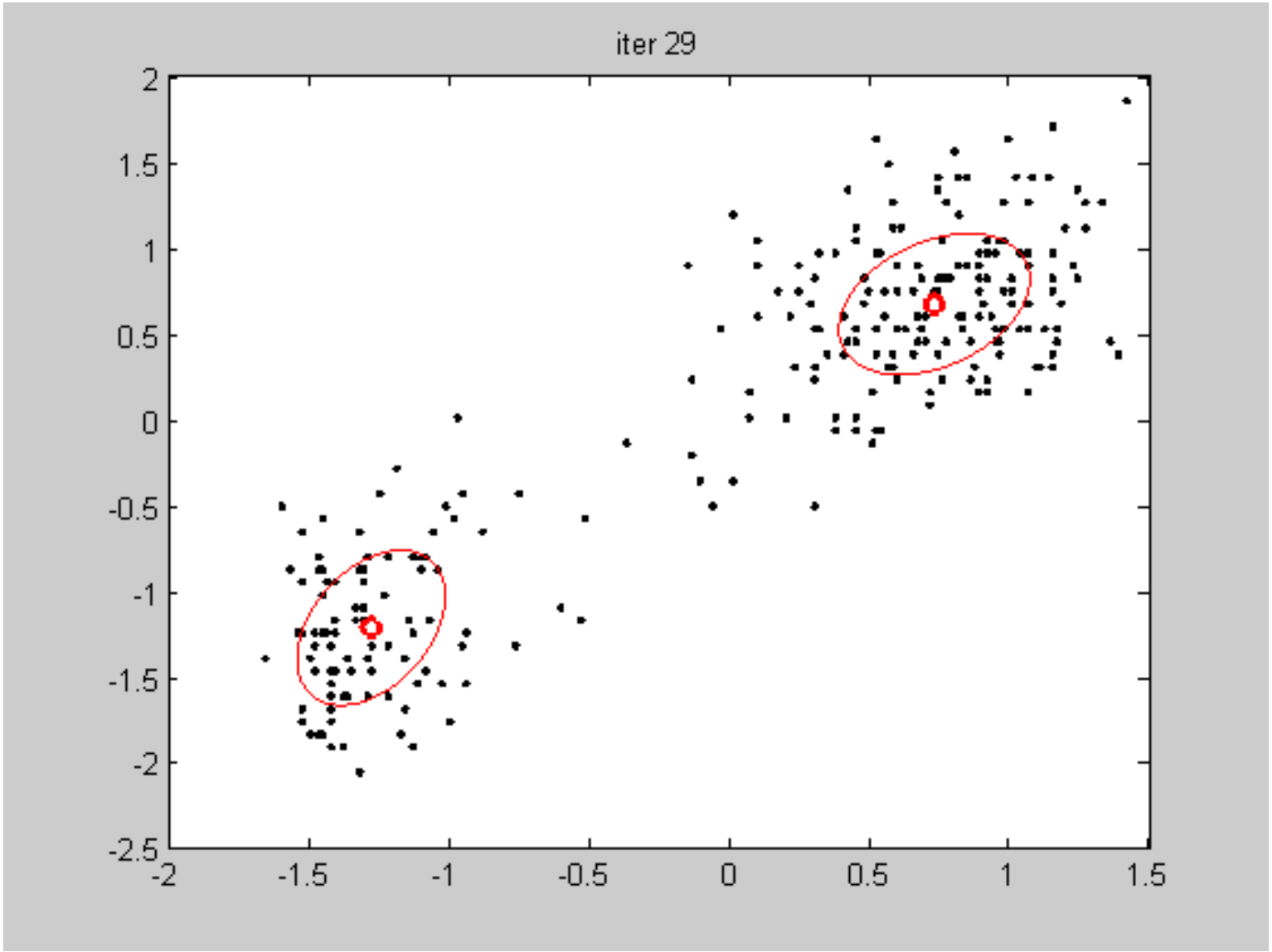
Demo - Automatic pruning of components



Demo - Automatic pruning of components



Demo - Automatic pruning of components



FlowVB is largely based on the algorithm presented in

C Archambeau and M Verleysen. Robust Bayesian Clustering. *Neural Networks*, 20(1):129–138, 2007.

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Neural Networks 20 (2007) 129–138

Neural Networks

www.elsevier.com/locate/neunet

Robust Bayesian clustering

Cédric Archambeau*, Michel Verleysen[1]

Machine Learning Group, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Received 25 October 2005; accepted 12 June 2006

Abstract

A new variational Bayesian learning algorithm for Student-t mixture models is introduced. This algorithm leads to (i) robust density estimation, (ii) robust clustering and (iii) robust automatic model selection. Gaussian mixture models are learning machines which are based on a divide-and-conquer approach. They are commonly used for density estimation and clustering tasks, but are sensitive to outliers. The Student-t distribution has heavier tails than the Gaussian distribution and is therefore less sensitive to any departure of the empirical distribution from Gaussianity. As

The latent variable model

A finite Student-t mixture model (SMM) is of the form

$$p(\boldsymbol{x}|\boldsymbol{\theta}_s) = \sum_{m=1}^{M} \pi_m \, \mathcal{S}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m, \nu_m),$$

where

π_m	Mixing proportions, with $\sum_{m=1}^{M} \pi_m = 1$
$\boldsymbol{\mu}_m$	Expected value of component m
$\boldsymbol{\Lambda}_m$	Precision matrix of component m
ν_m	Degrees of freedom of component m
$\boldsymbol{\theta}_s$	Ensemble of parameters

The latent variable model

Let us introduce a set of indicator variables $Z = \{\boldsymbol{z}\}_{n=1}^{N}$, where $z_{nm} = 1$ iff datapoint n belongs to component m (and $z_{nm} = 0$ otherwise).

The latent variable model

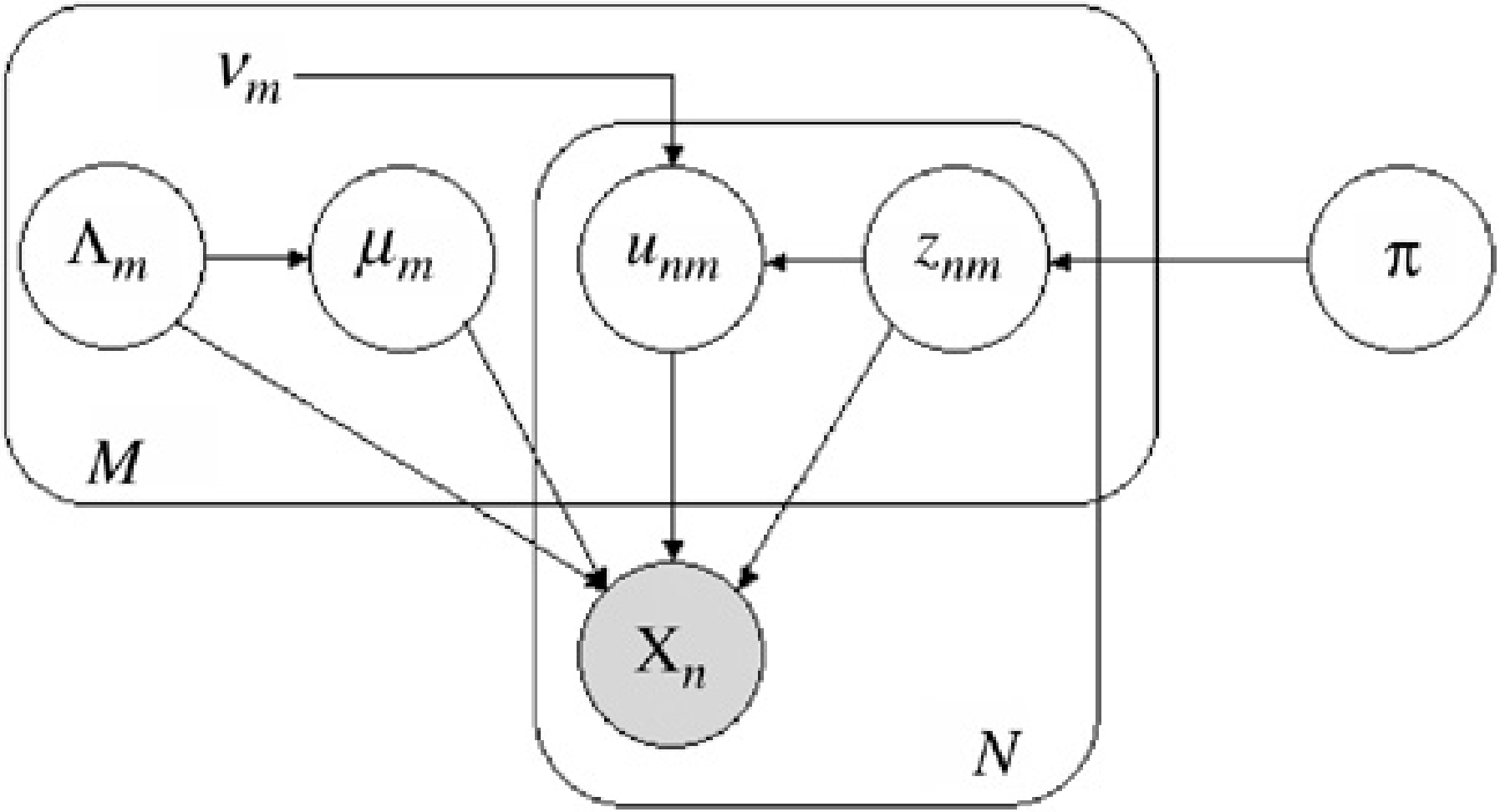
Let us introduce a set of indicator variables $Z = \{\boldsymbol{z}\}_{n=1}^{N}$, where $z_{nm} = 1$ iff datapoint n belongs to component m (and $z_{nm} = 0$ otherwise).

In contrast to GMM we need a further latent variable. Note that the Student-t density can be written as an *infinite scale mixture*:

$$\mathcal{S}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}, \nu) = \int_0^{+\infty} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, u\boldsymbol{\Lambda})\, \mathcal{G}\left(u \middle| \frac{\nu}{2}, \frac{\nu}{2}\right) \boldsymbol{d}u$$

We therefore have unobserved variables z_{nm} and u_{nm} for every observation and component.

Representation as Bayesian Graphical Model



from Archambeau & Verleysen (2007)

Variational Bayesian Inference

The evidence of the data

$$p(X|\mathcal{H}_M),$$

given the model structure $\mathcal{H}_M$, is untractable when using the real posterior

$$p(U, Z, \theta_s|X, \mathcal{H}_M).$$

Instead use an approximation and assume that it factorizes as

$$q(U, Z, \theta_s) = q(U, Z)\, q(\theta_s).$$

It can be shown that this leads to a *lower bound* for the real evidence.

Variational Bayesian Inference (cont.)

The VBEM iteration consists of two steps:

Variational Bayesian Inference (cont.)

The VBEM iteration consists of two steps:

VBE-step: Update the latent variables U and Z to maximize $q(U, Z)$.

Variational Bayesian Inference (cont.)

The VBEM iteration consists of two steps:

VBE-step: Update the latent variables U and Z to maximize $q(U, Z)$.

VBM-step: Update the parameters θ_s to maximize $q(\theta_s)$.

Variational Bayesian Inference (cont.)

The VBEM iteration consists of two steps:

VBE-step: Update the latent variables U and Z to maximize $q(U, Z)$.

VBM-step: Update the parameters θ_s to maximize $q(\theta_s)$.

And repeat...

Variational Bayesian Inference (cont.)

The VBEM iteration consists of two steps:

VBE-step: Update the latent variables U and Z to maximize $q(U, Z)$.

VBM-step: Update the parameters θ_s to maximize $q(\theta_s)$.

And repeat...

Stop the iteration when the lower bound stops increasing.

Variational Bayesian Inference (cont.)

The VBEM iteration consists of two steps:

VBE-step: Update the latent variables U and Z to maximize $q(U, Z)$.

VBM-step: Update the parameters θ_s to maximize $q(\theta_s)$.

And repeat...

Stop the iteration when the lower bound stops increasing.

During the iteration we remove clusters where π_m is close to zero.

Initialization

- We use k-means to initialize starting points for our algorithm

Initialization

- We use k-means to initialize starting points for our algorithm
- Unfortunately, EM-type algorithms don't guarantee the global optimum

Initialization

- We use k-means to initialize starting points for our algorithm
- Unfortunately, EM-type algorithms don't guarantee the global optimum
- To increase the chances of finding the best solution, we perform random restarts, where we

Initialization

- We use k-means to initialize starting points for our algorithm
- Unfortunately, EM-type algorithms don't guarantee the global optimum
- To increase the chances of finding the best solution, we perform random restarts, where we
  - Initialize k-means with $3 \times M$ clusters

Initialization

- We use k-means to initialize starting points for our algorithm
- Unfortunately, EM-type algorithms don't guarantee the global optimum
- To increase the chances of finding the best solution, we perform random restarts, where we
  - Initialize k-means with $3 \times M$ clusters
  - For every restart randomly select M points from this solution

Experiences with FlowCAP

- Started as class project at UBC

Experiences with FlowCAP

- Started as class project at UBC
- Tuned our algorithm to a (different) GvHD dataset

Experiences with FlowCAP

- Started as class project at UBC
- Tuned our algorithm to a (different) GvHD dataset
- Good results with GvHD and DLBCL

Experiences with FlowCAP

- Started as class project at UBC
- Tuned our algorithm to a (different) GvHD dataset
- Good results with GvHD and DLBCL
- Not so good results with HSCT, WNV, ND

Experiences with FlowCAP

- Started as class project at UBC
- Tuned our algorithm to a (different) GvHD dataset
- Good results with GvHD and DLBCL
- Not so good results with HSCT, WNV, ND
- Found that some Pre- and Post-processing improved results with GvHD

Pre-processing

Pre-processing: Principal components analysis (PCA)

Pre-processing

Pre-processing: Principal components analysis (PCA)

- includes centering and whitening data

Pre-processing

Pre-processing: Principal components analysis (PCA)

- includes centering and whitening data
- keep only factors with eigenvalues > 1

Pre-processing

Pre-processing: Principal components analysis (PCA)

- includes centering and whitening data
- keep only factors with eigenvalues > 1
- improves signal/noise ratio

Pre-processing

Pre-processing: Principal components analysis (PCA)

- includes centering and whitening data
- keep only factors with eigenvalues > 1
- improves signal/noise ratio
- since real-world flow-cytometry data is inherently noisy, this should improve clustering results

Pre-processing

Pre-processing: Principal components analysis (PCA)

- includes centering and whitening data
- keep only factors with eigenvalues > 1
- improves signal/noise ratio
- since real-world flow-cytometry data is inherently noisy, this should improve clustering results
- some performance gains

Post-processing

Post-processing: FlowMerge

Post-processing

Post-processing: FlowMerge

- as with GMMs, algorithm favors more clusters than manual analysis

Post-processing

Post-processing: FlowMerge

- as with GMMs, algorithm favors more clusters than manual analysis
- Finak et al. Merging Mixture Components for Cell Population Identification in Flow Cytometry. *Advances in Bioinformatics*, 2009.

Post-processing

Post-processing: FlowMerge

- as with GMMs, algorithm favors more clusters than manual analysis
- Finak et al. Merging Mixture Components for Cell Population Identification in Flow Cytometry. *Advances in Bioinformatics*, 2009.
- wrote our own implementation of FlowMerge

Summary

Keypoints of our FlowVB-algorithm

Summary

Keypoints of our FlowVB-algorithm

- Robust inference by using Student-t distribution

Summary

Keypoints of our FlowVB-algorithm

- Robust inference by using Student-t distribution
- One-pass determination of number of mixture components

Summary

Keypoints of our FlowVB-algorithm

- Robust inference by using Student-t distribution
- One-pass determination of number of mixture components
- Optional random restarts to improve results

Summary

Keypoints of our FlowVB-algorithm

- Robust inference by using Student-t distribution
- One-pass determination of number of mixture components
- Optional random restarts to improve results
- Improve performance by using PCA and flowMerge

Further developement

We are porting the algorithm to Python for further developement.

We are looking for additional developers and feedback from users.

Get involved:

Project page: **http://github.com/FlowVB/FlowVB**
Mailing list: **http://groups.google.com/group/flowvb**

Appendix

Variational Bayesian Inference

For a given model structure $\mathcal{H}_M$, the evidence is given by:

$$p(X|\mathcal{H}_M) = \int_{\theta_s} \int_U \sum_Z p(X, U, Z, \theta_s|\mathcal{H}_M) \boldsymbol{d}U \boldsymbol{d}\theta_s$$

This quantity is untractable, but it can be lower-bounded by

$$\begin{aligned} \log p(X|\mathcal{H}_M) \geq & \log p(X|\mathcal{H}_M) \\ & - KL[q(U, Z, \theta_s)||p(U, Z, \theta_s|X, \mathcal{H}_M)], \end{aligned}$$

where $KL(\cdot)$ is the Kullback-Leibler divergence, and $q(U, Z, \theta_s)$ is an approximation to $p(U, Z, \theta_s|X, \mathcal{H}_M)$.
By assuming that q factorizes as $q(U, Z, \theta_s) = q(U, Z)\, q(\theta_s)$, this quantity becomes tractable.

The latent variable model (cont.)

Putting the above together we get the following latent variable model:

$$p(\boldsymbol{z}_n|\boldsymbol{\theta}_s, \mathcal{H}_M) = \prod_{m=1}^{M} \pi_m^{z_{nm}}$$

$$p(\boldsymbol{u}_n|\boldsymbol{\theta}_s, \mathcal{H}_M) = \prod_{m=1}^{M} \mathcal{G}\left(u_{nm} \middle| \frac{\nu_m}{2}, \frac{\nu_m}{2}\right)^{z_{nm}}$$

$$p(\boldsymbol{x}_n|\boldsymbol{\theta}_s, \mathcal{H}_M) = \prod_{m=1}^{M} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_m, u_{nm}\boldsymbol{\Lambda}_m)^{z_{nm}}$$

Priors

We introduce the following conjugate priors for our parameters:

Parameter	*Prior*
$\boldsymbol{\pi} = \{\pi_m\}_{m=1}^{M}$	$\mathcal{D}(\boldsymbol{\pi}\|\varkappa_0)$ (Dirichlet)
$\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m$	$\mathcal{NW}(\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m\|\boldsymbol{\theta}_{\mathcal{NW}_0}) = \mathcal{N}(\boldsymbol{\mu}_m\|\boldsymbol{m}_0, \eta_0\boldsymbol{\Lambda}_m)\mathcal{W}(\boldsymbol{\Lambda}_m\|\gamma_0, \boldsymbol{S}_0)$ (Joint Gaussian-Wishart)
ν_m	no prior, since no conjugate prior exists