CSC321 Tutorial 8: Assignment 3: Mixture of Gaussians

(K-means slides based on Maksims Volkovs's and many figures from Bishop 2006 textbook: Pattern recognition and machine learning)

Yue Li
Email: yueli@cs.toronto.edu

Wed 11-12 March 12
Fri 10-11 March 14

Outline

- Setting:
  - data: $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  - goal: partition the data into K clusters
  - objective function in K-means:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2 \tag{1}$$

- Setting:
  - data: $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  - goal: partition the data into K clusters
  - objective function in K-means:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2 \tag{1}$$

- Algorithm:
  1. initialize K cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
  2. assign each point $\mathbf{x}_n$ to the closest center k:

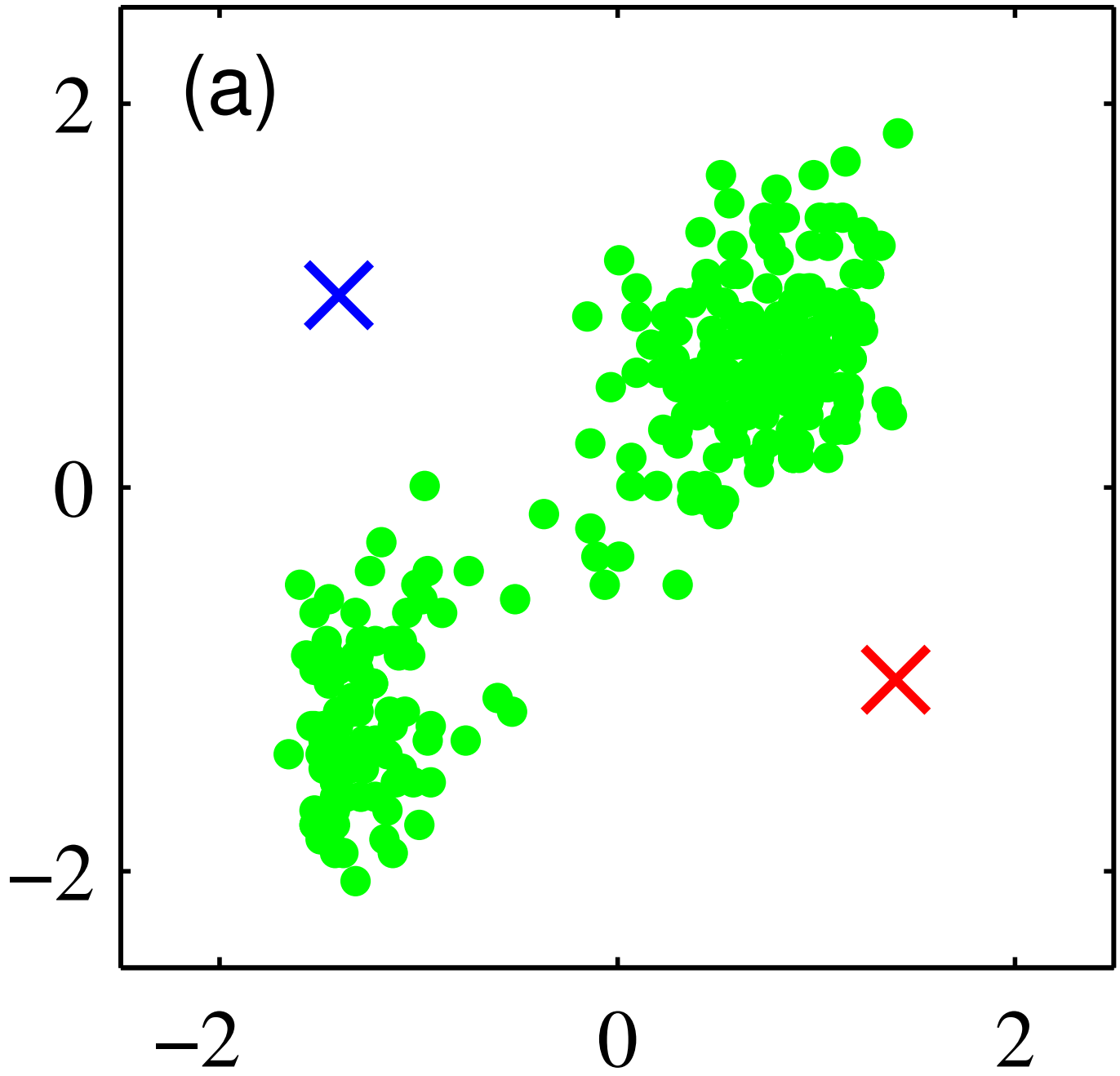
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||^2 \\ 0 & \text{if otherwise} \end{cases}$$

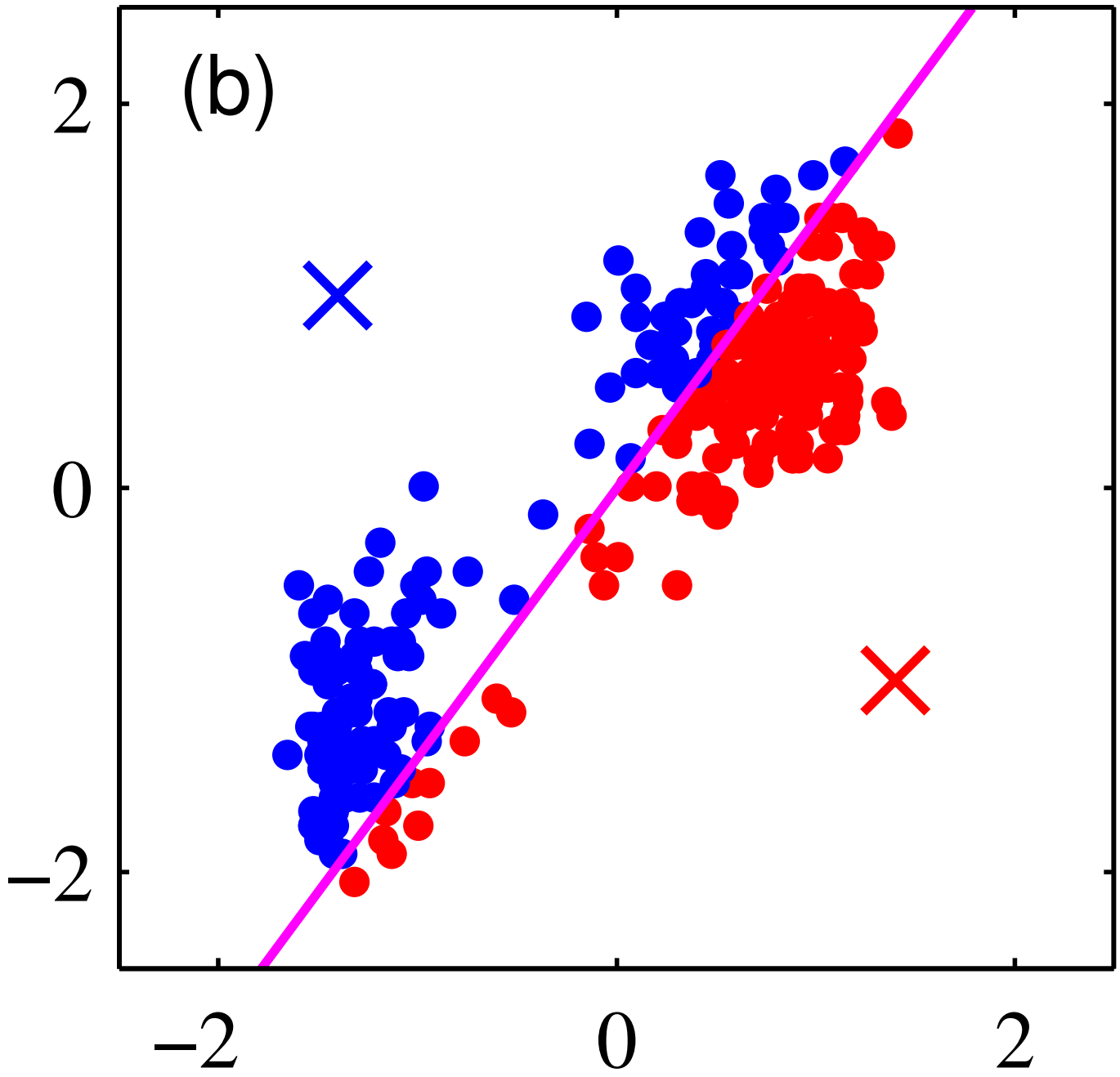
  3. update cluster centers:

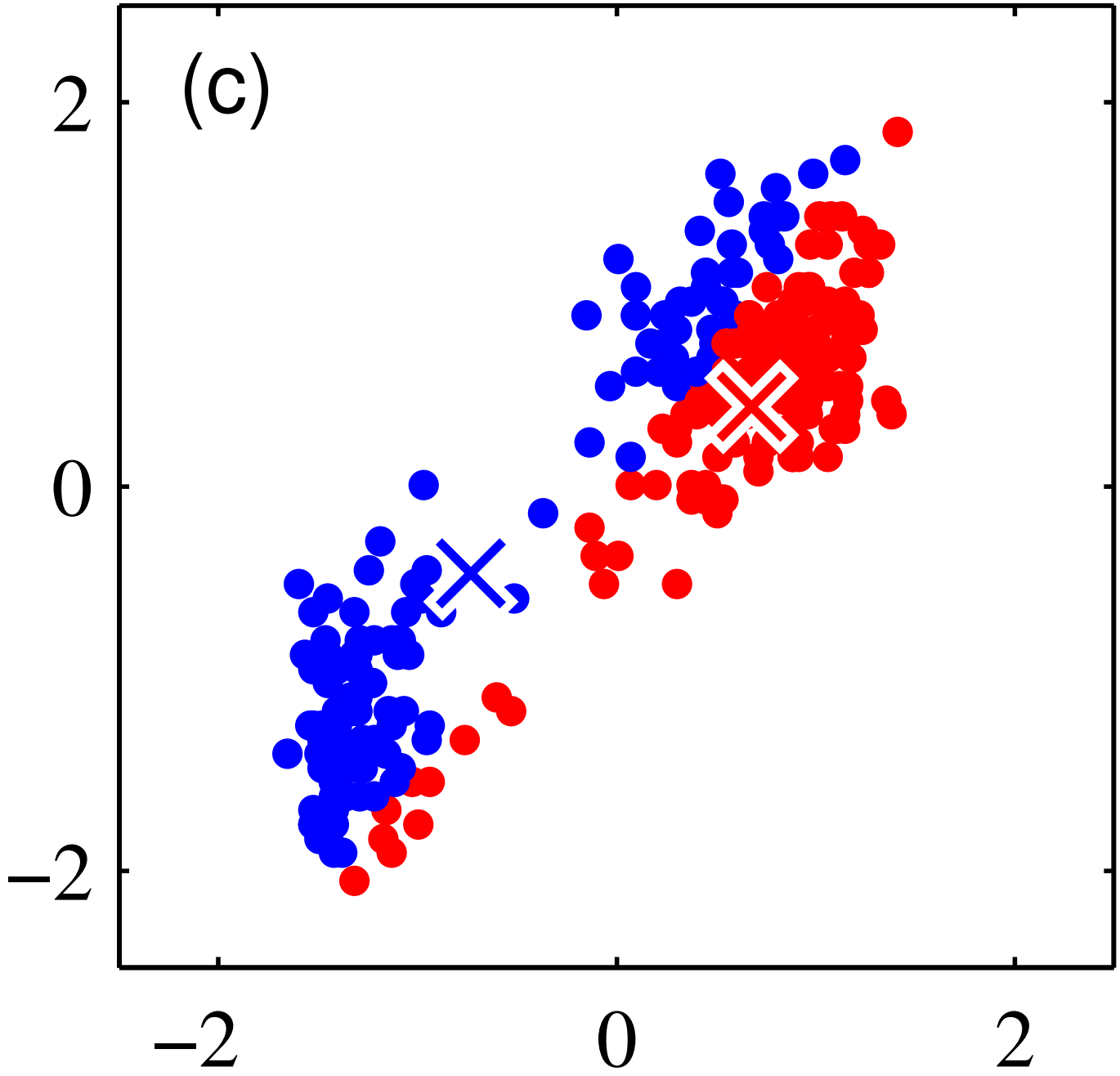
$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_k}{\sum_n r_{nk}}$$

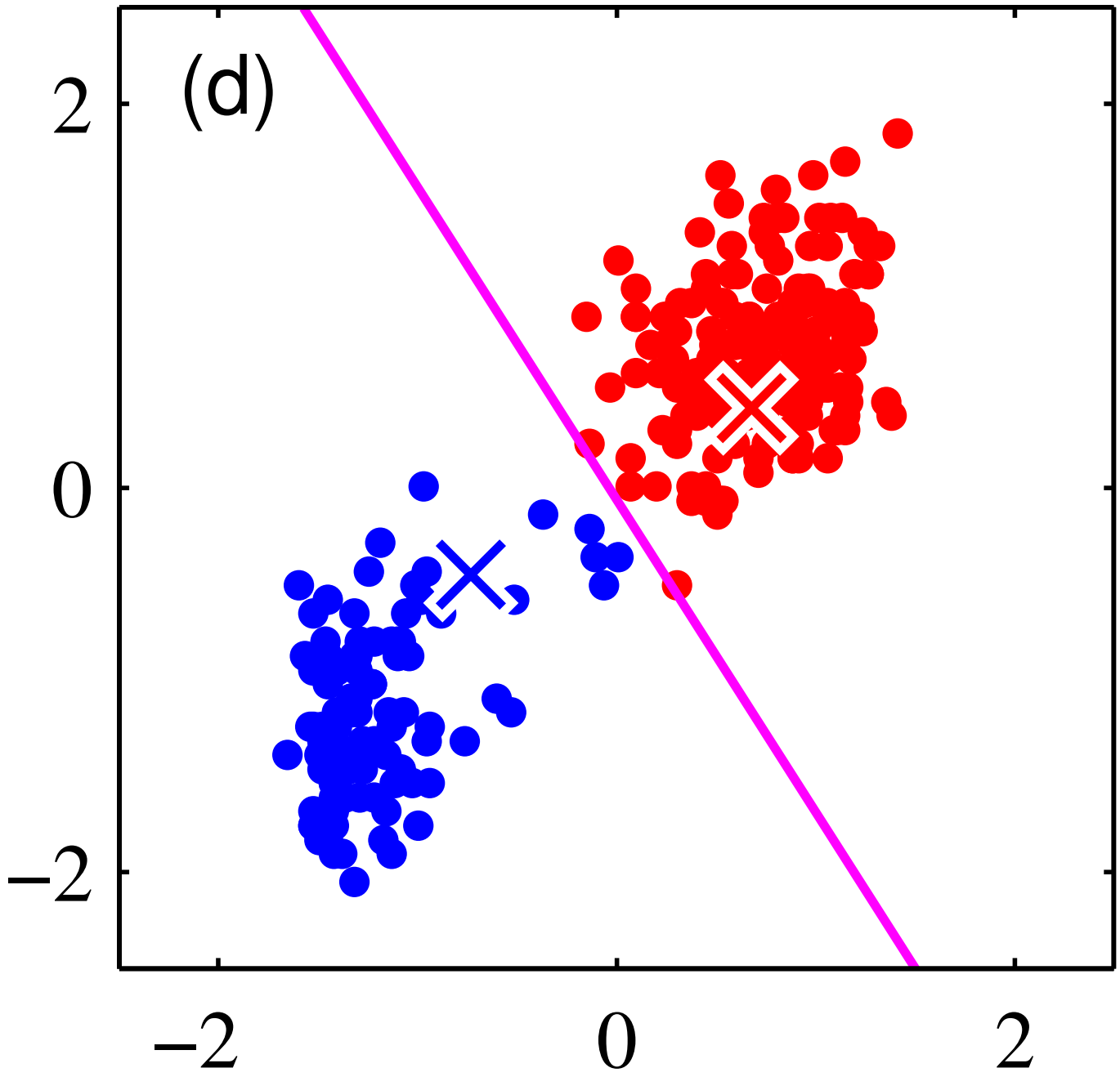
  4. repeat 2 & 3 until convergence (i.e. little change from (1))

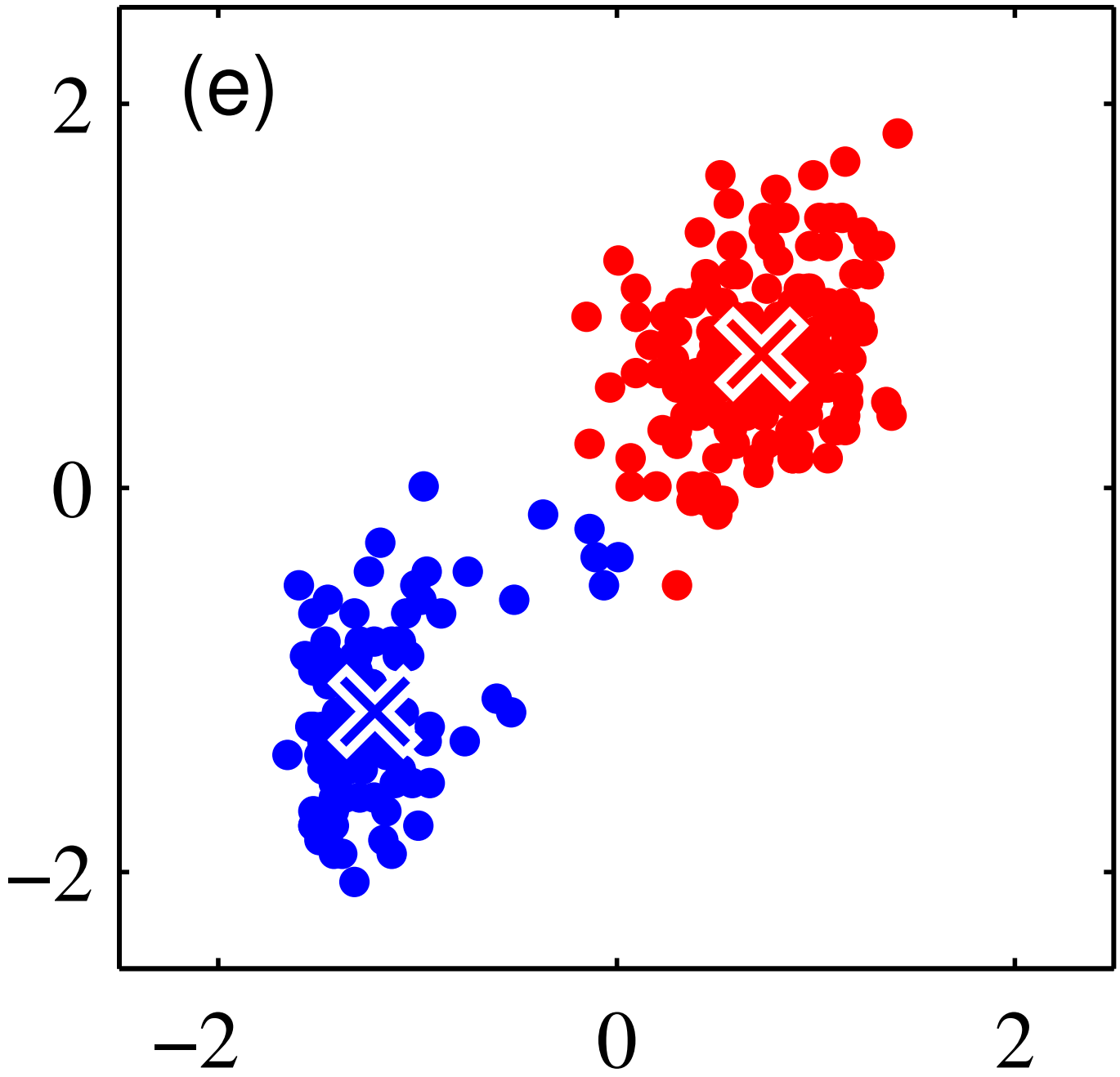
Pattern recognition and machine learning (Bishop, 2006)

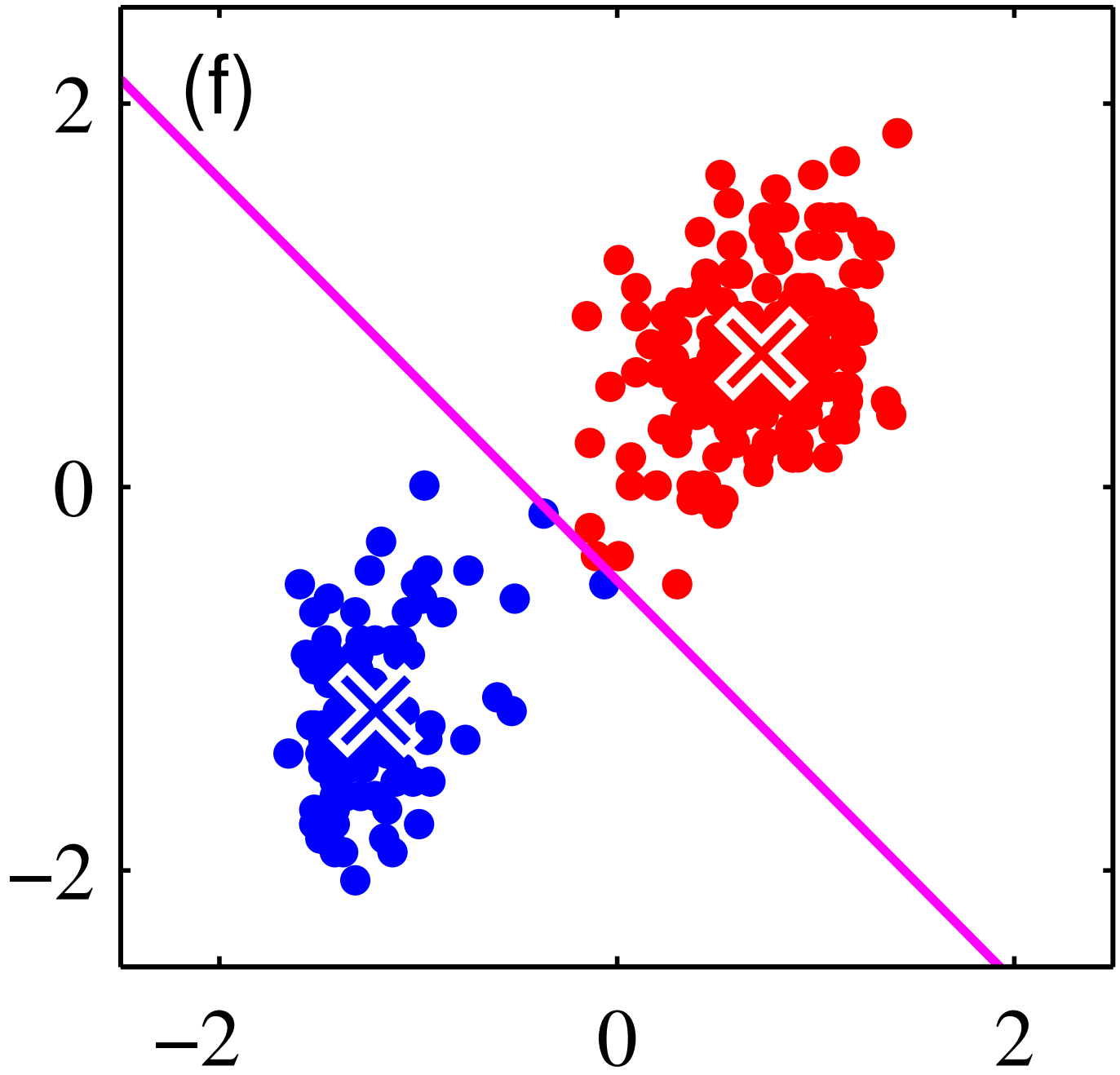

(a)
2
0
−2
−2
0
2

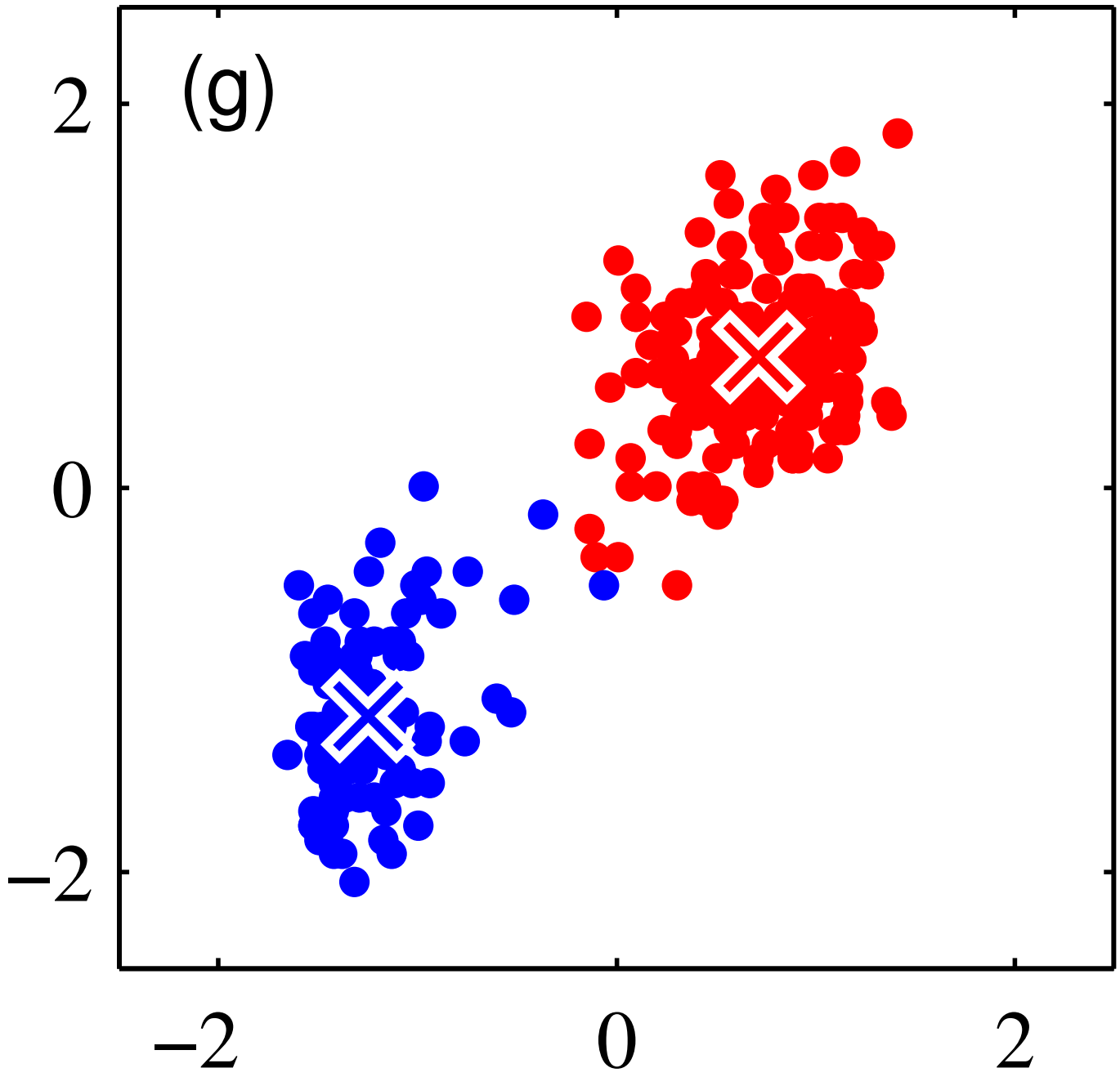

(b)
2
0
−2
−2
0
2

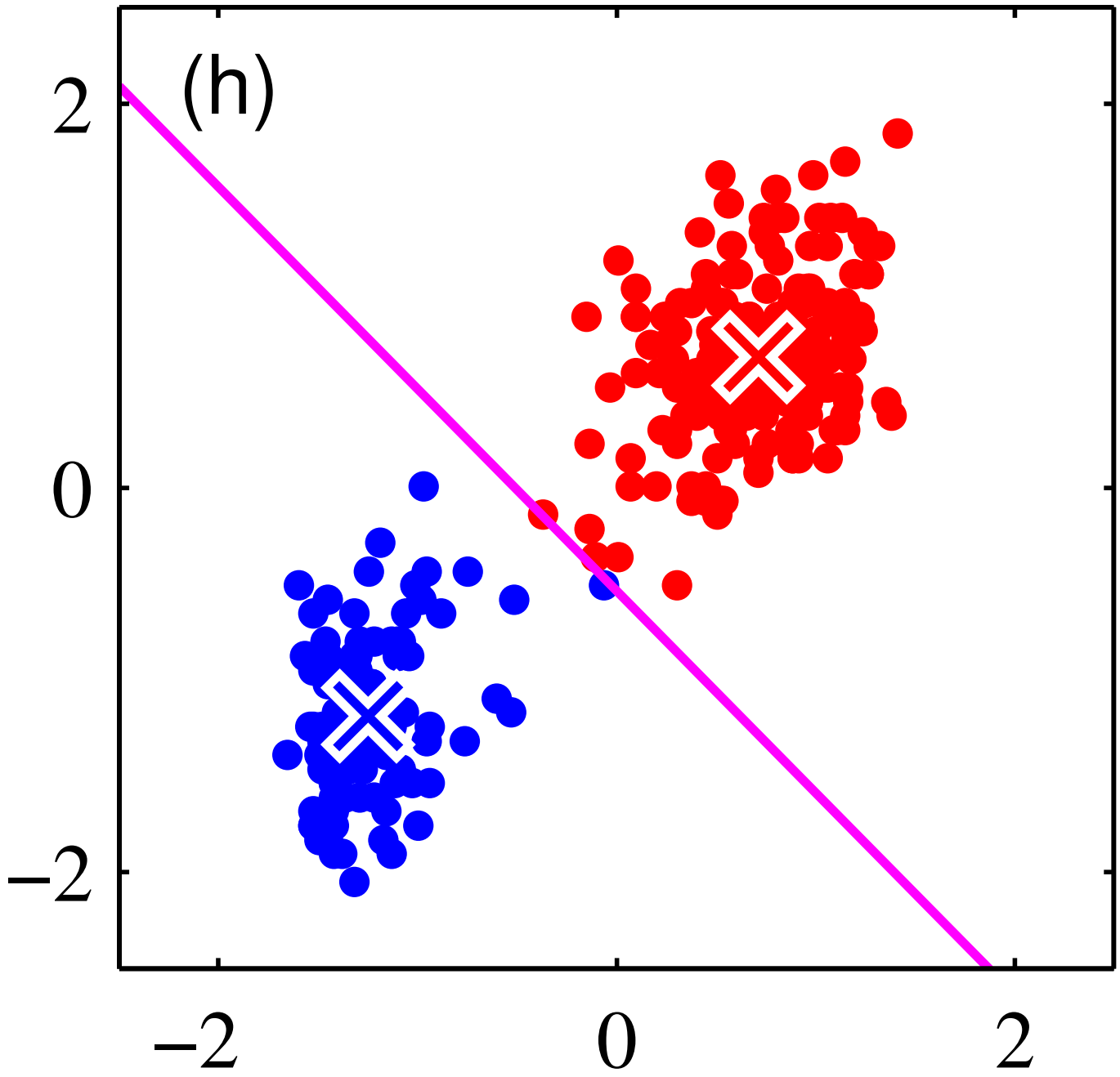

(c)
2
0
−2
−2
0
2

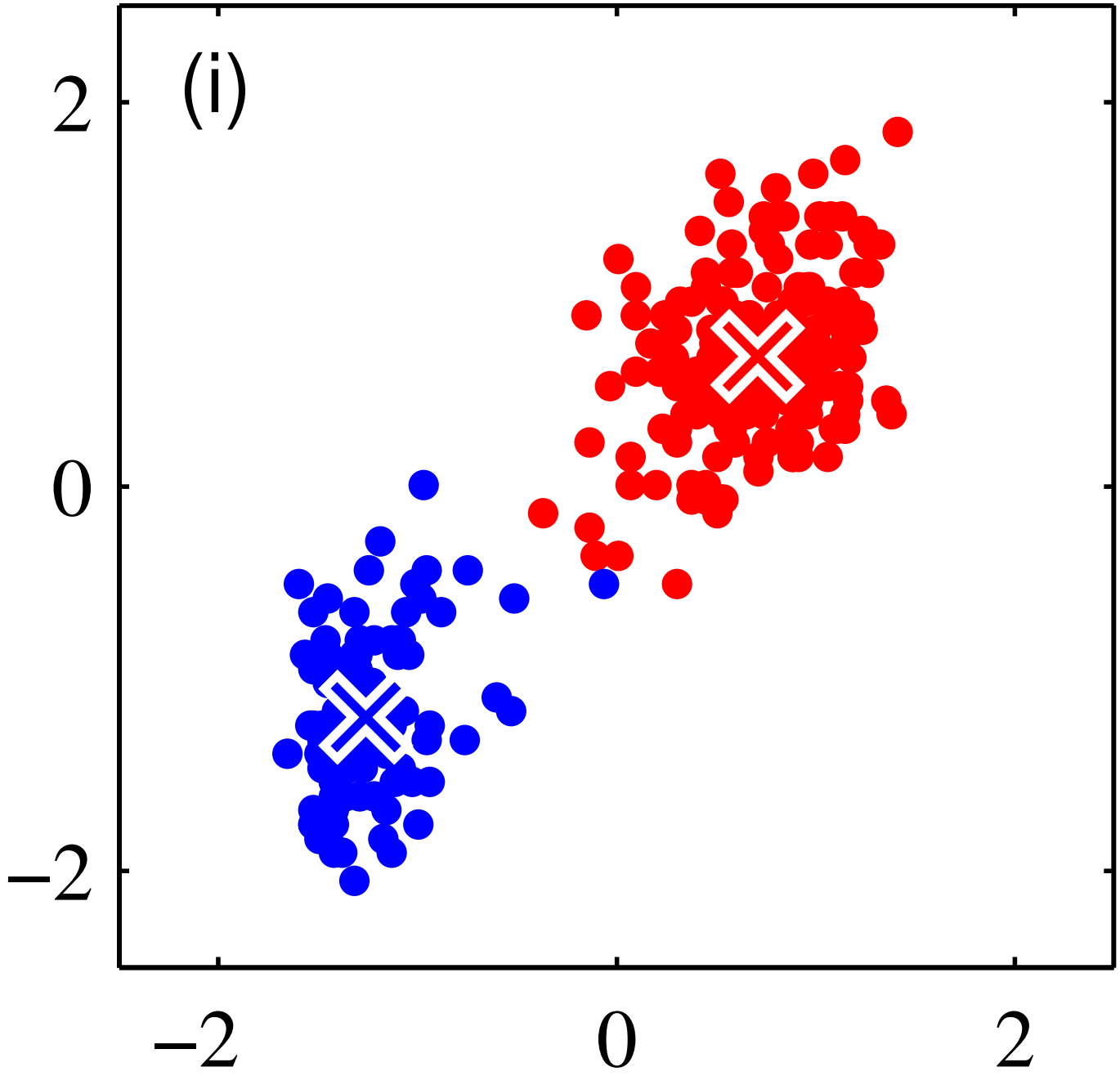

(d)
2
0
−2
−2
0
2

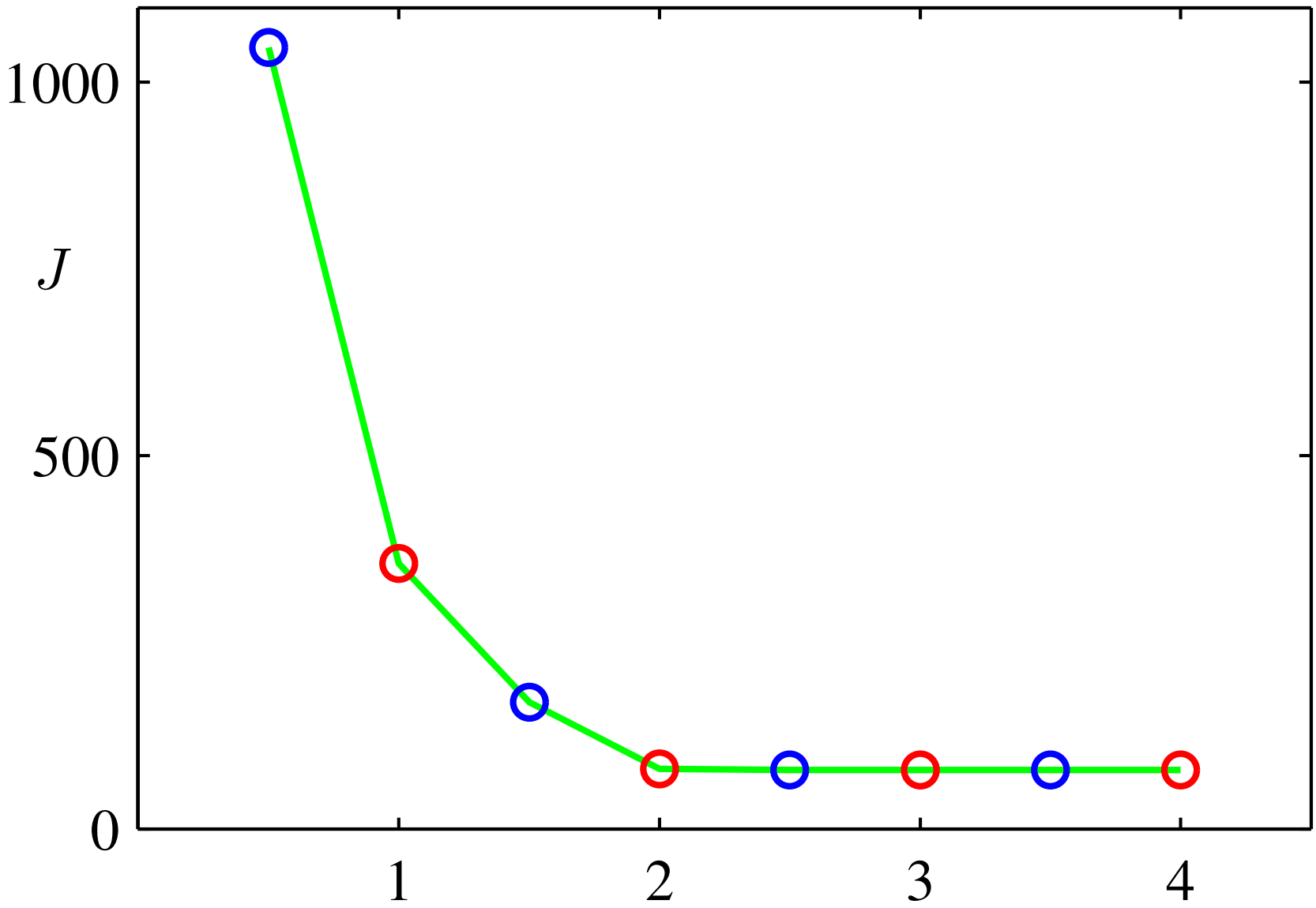

(e)
2
0
−2
−2
0
2


(f)
2
0
−2
−2
0
2


(g)
2
0
−2
−2
0
2

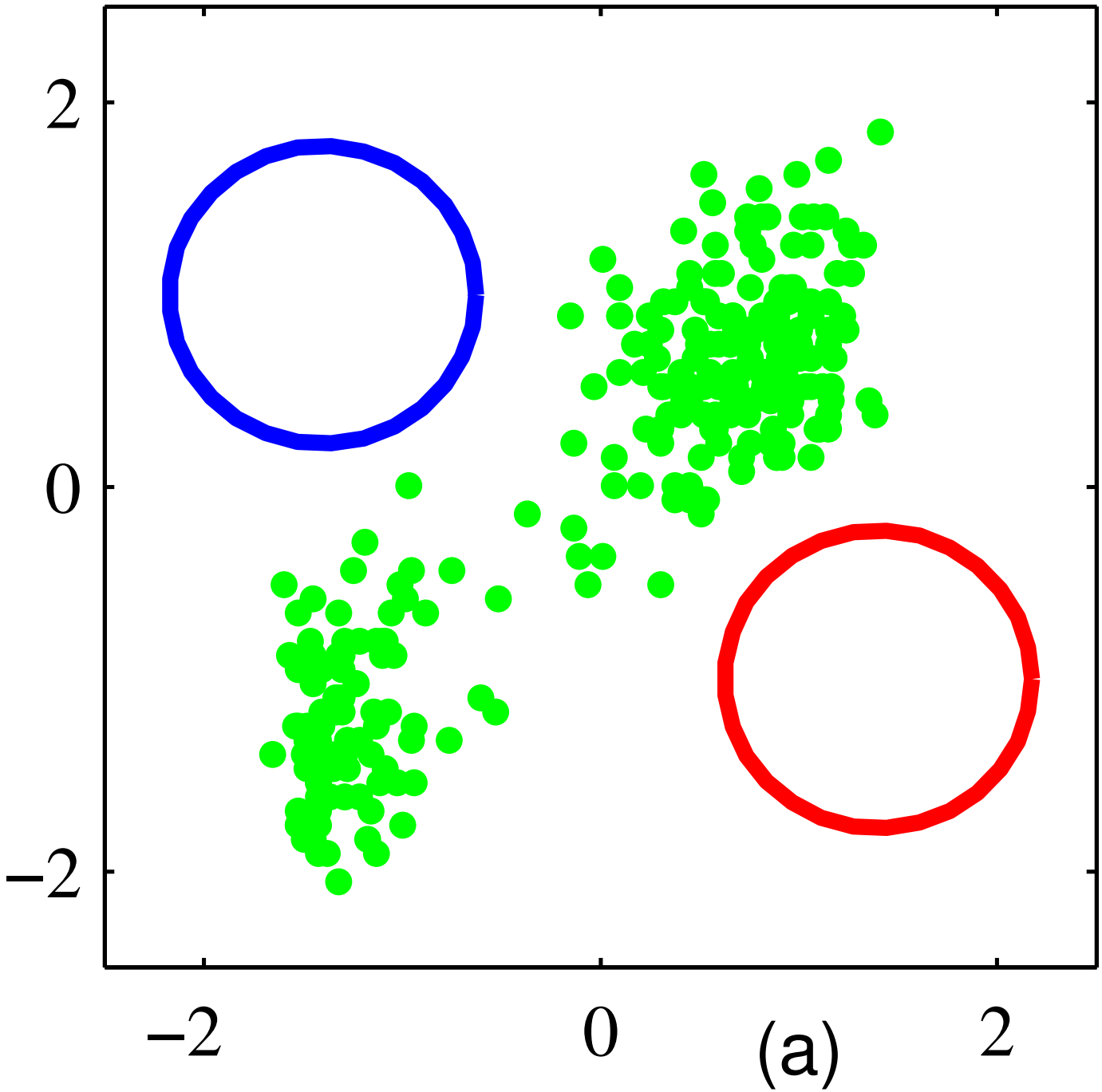

(h)
2
0
−2
−2
0
2

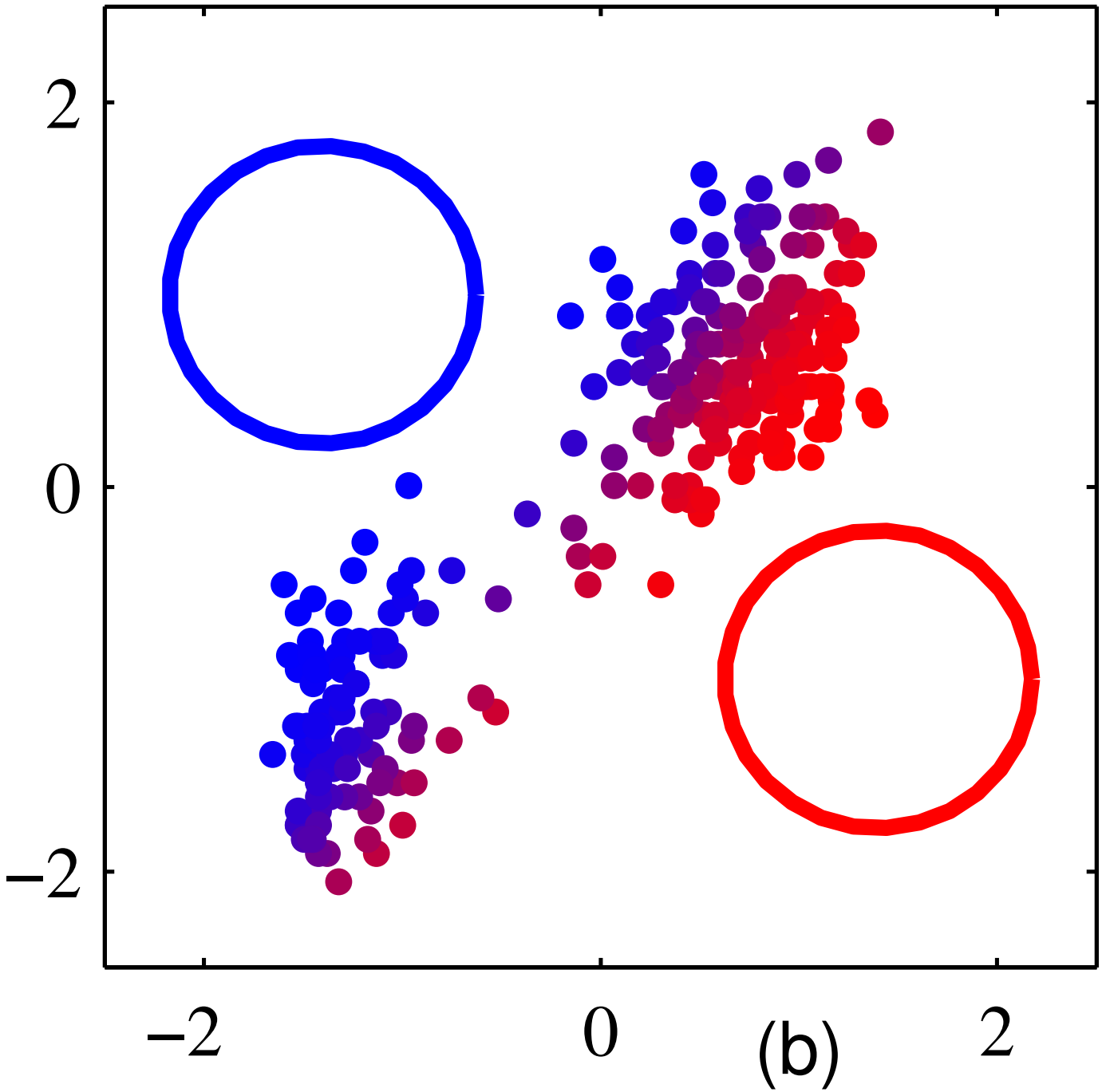

(i)
2
0
−2
−2
0
2

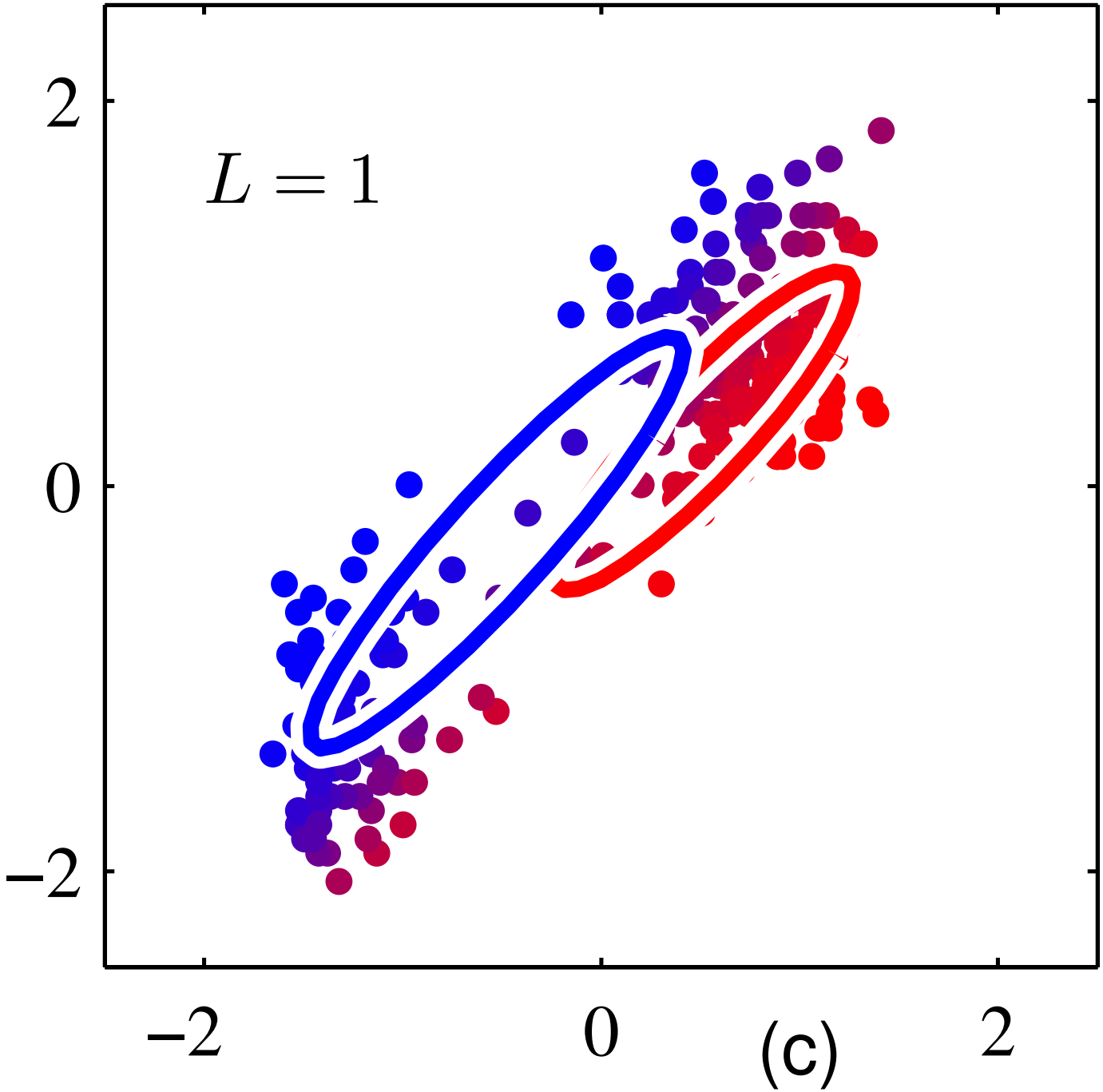

1000
J
500
0
1
2
3
4

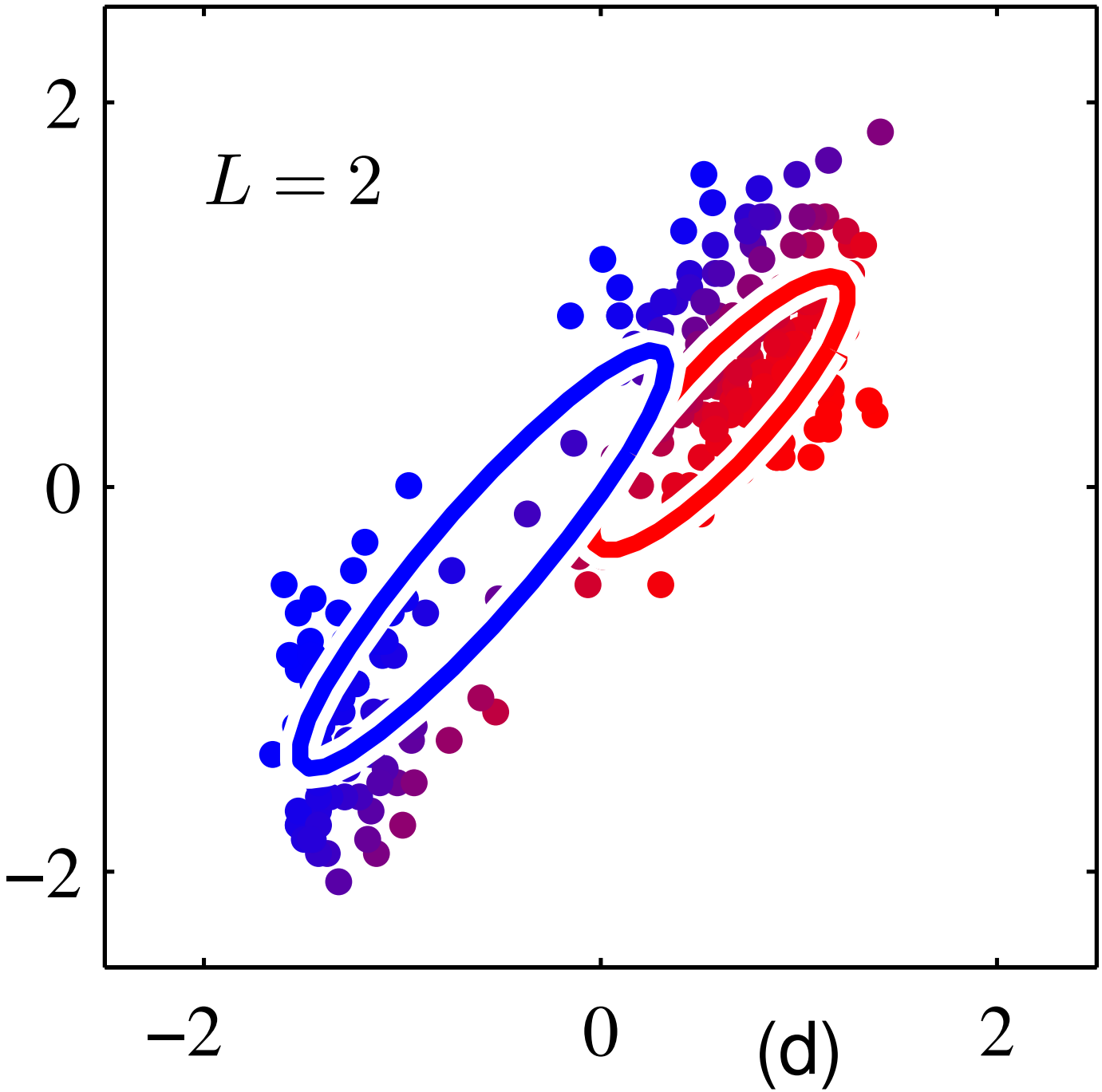
Mixture of Gaussians (MoG)

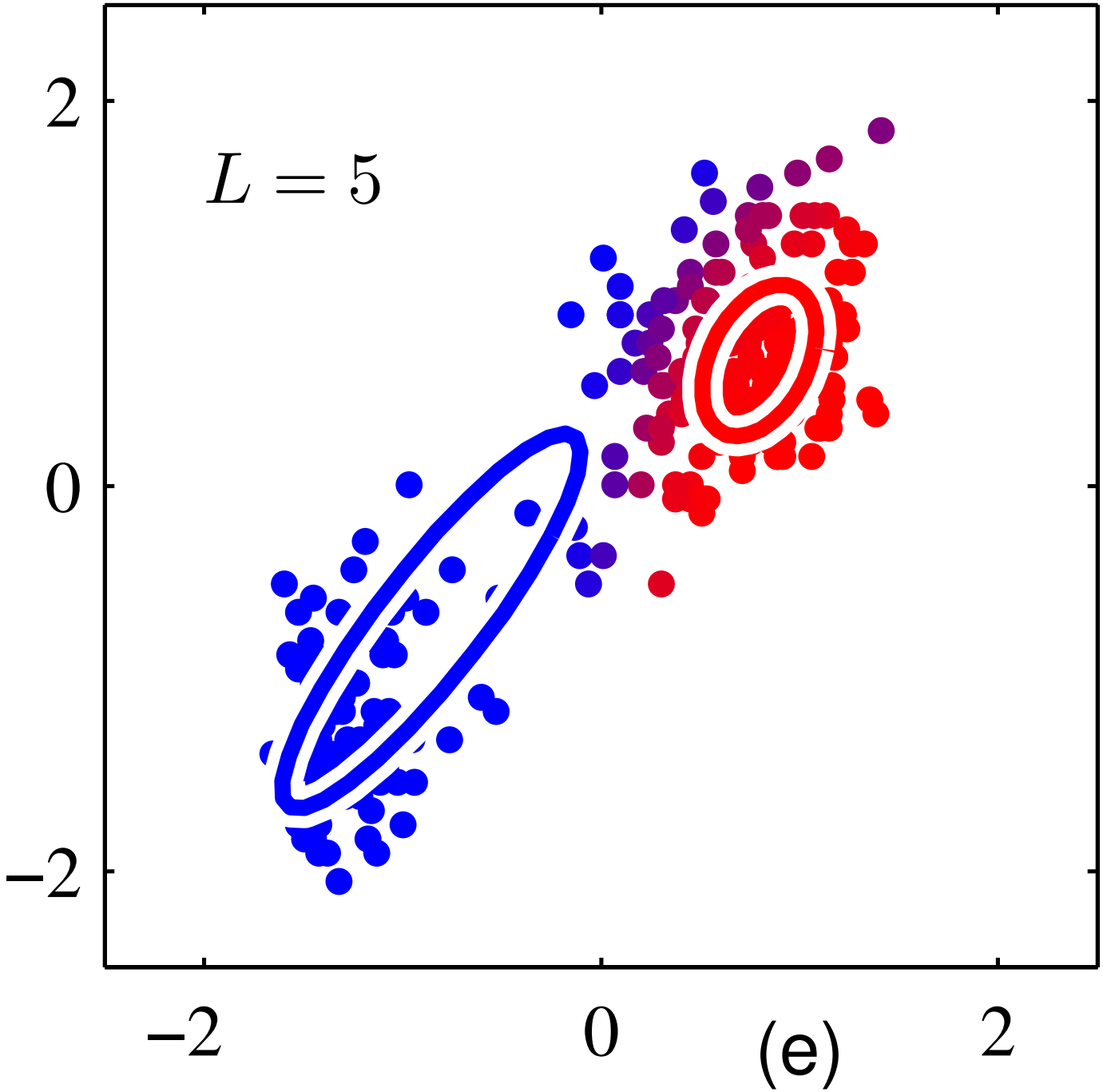
- A soft K-means that incorporates uncertainty in the cluster assignment to each data point (e.g., points on the boundary in slide 1):
  - $r_{nk} \in [0, 1]$ rather than $r_{nk} \in \{0, 1\}$
- A model-based method that assumes data points are *independently* sampled from K Gaussians $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Two important questions to address in MoG:
  - What is the objective function of MoG?
  - How to fit MoG to optimize such objective function?
- To further motivate MoG, let's see the following example from Bishop (2006) that model the same data from the previous K-means example.

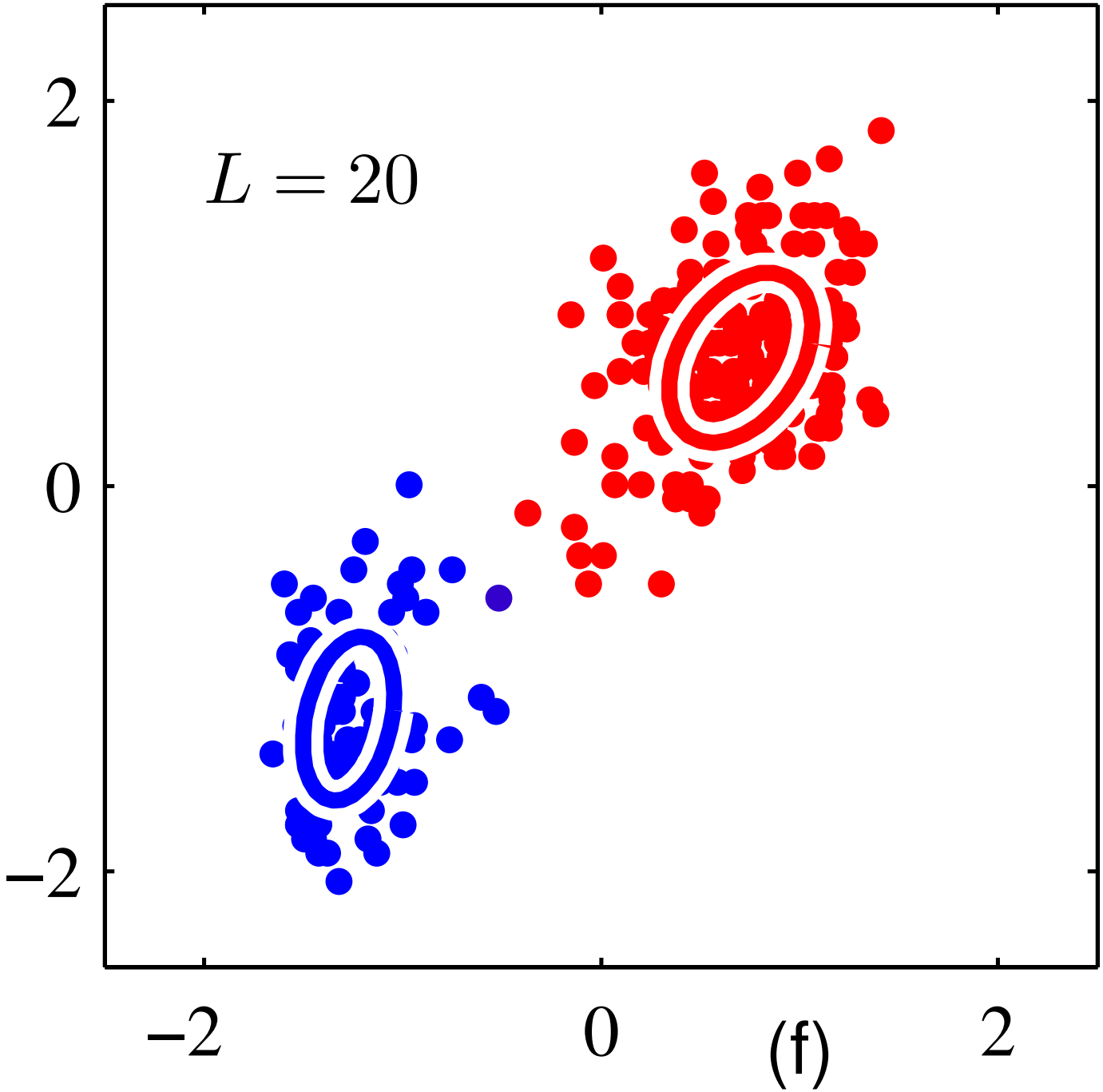

2
0
−2
−2
0
2
(a)


2
0
−2
−2
0
2
(b)

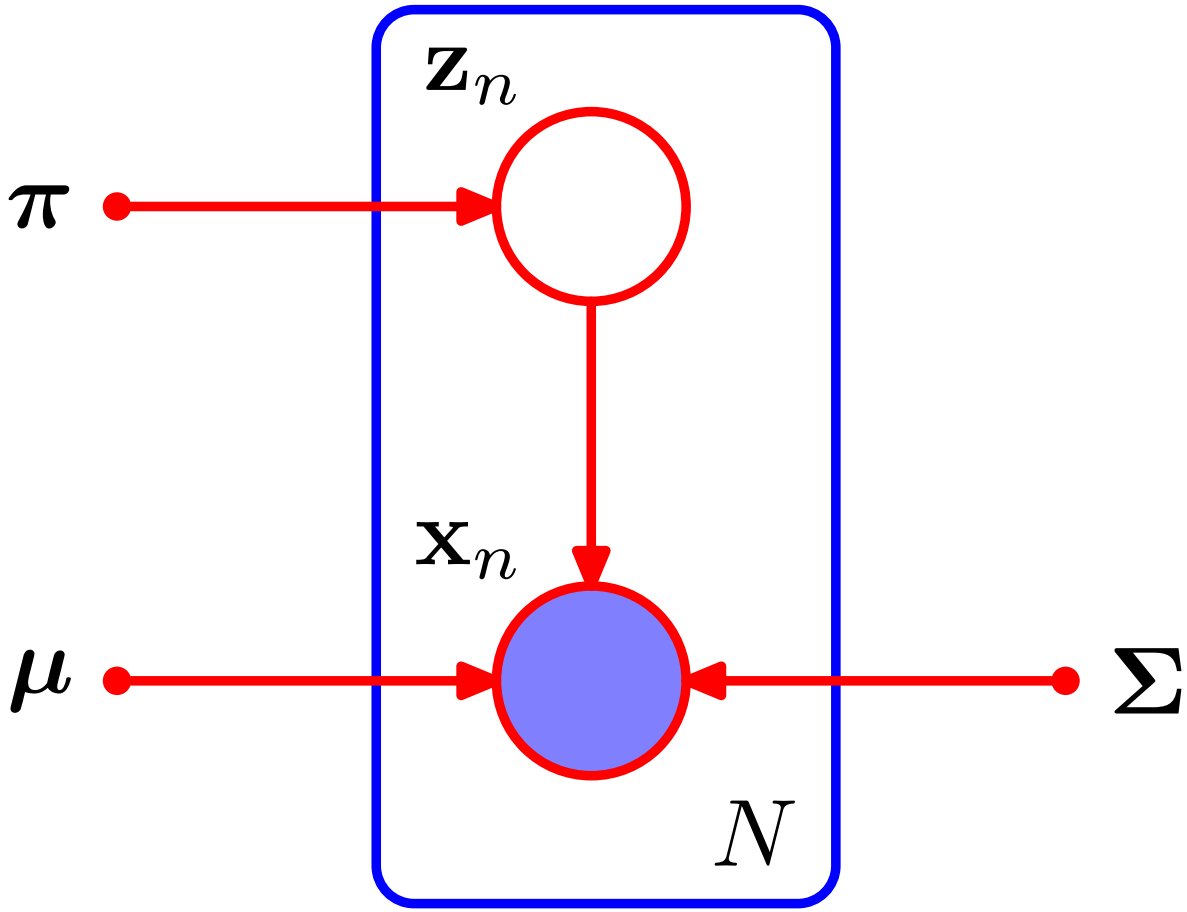

$L = 1$
2
0
−2
−2
0
2
(c)


$L = 2$
2
0
−2
−2
0
2
(d)


$L = 5$
2
0
−2
−2
0
2
(e)


$L = 20$
2
0
−2
−2
0
2
(f)

A graphical model view of MoG



Pattern recognition and machine learning, Chapter 9 p433, (Bishop, 2006)

- Density function of one data point $\mathbf{x}_n$:

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} p(z_k) p(\mathbf{x}_n | \boldsymbol{\theta}_k)$$
$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where z_k is the latent variable, π_k is the probability of choosing Gaussian k to represent $\mathbf{x}_n$ (i.e., probability of z_k equal to 1). π_k is called the **mixing proportion**.

- Density function of one data point $\mathbf{x}_n$:

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} p(z_k) p(\mathbf{x}_n|\boldsymbol{\theta}_k)$$
$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  where z_k is the latent variable, π_k is the probability of choosing Gaussian k to represent $\mathbf{x}_n$ (i.e., probability of z_k equal to 1). π_k is called the **mixing proportion**.
- Objective function - log likelihood of all data points:

$$\ln p(\mathbf{X}) = \ln \prod_{n=1}^{N} p(\mathbf{x}_n)$$
$$= \ln \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$= \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Objective function:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Maximum likelihood (ML) solutions for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:

$$\frac{\partial \ln p(\mathbf{X})}{\partial \boldsymbol{\mu}_k} = 0 \Longrightarrow \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k) \mathbf{x}_n$$

$$\frac{\partial \ln p(\mathbf{X})}{\partial \boldsymbol{\Sigma}_k} = 0 \Longrightarrow \boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

$$\frac{\partial \ln p(\mathbf{X})}{\partial \pi_k} = 0 \Longrightarrow \pi_k = \frac{N_k}{N}$$

where $N_k = \sum_{n=1}^{N} \gamma(z_k)$ and $\gamma(z_k) = p(z_k = 1 | \mathbf{x}_n)$

In more details, $\gamma(z_k)$ is the **posterior probability** (aka **responsibility** of z_k to $\mathbf{x}_n$):

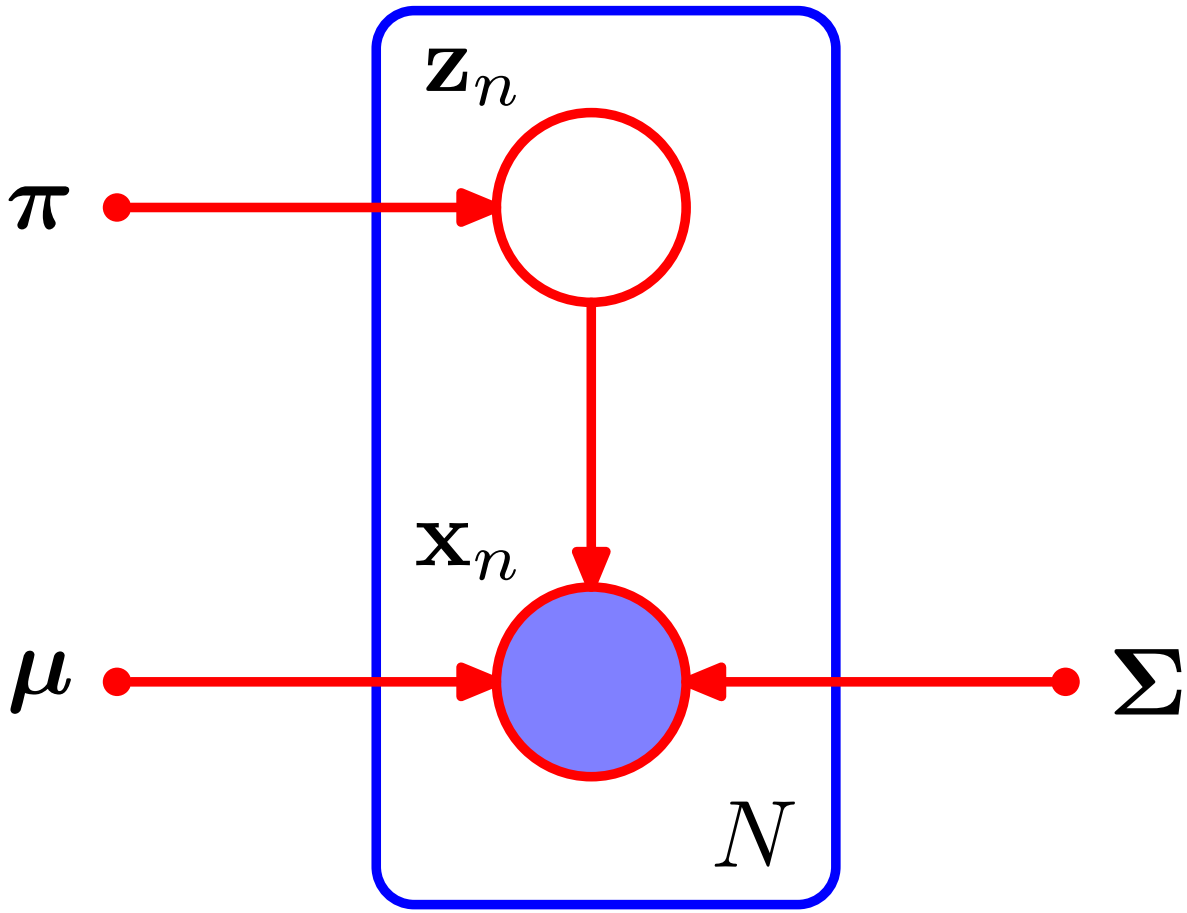
$$\gamma(z_k) = p(z_k = 1|\mathbf{x}_n)$$

$$= \frac{p(z_k = 1)p(\mathbf{x}_n|z_k = 1)}{p(\mathbf{x}_n)} \qquad \text{(Bayes' Rule)}$$

$$= \frac{p(z_k = 1)p(\mathbf{x}_n|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}_n|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$$p(z_k = 1|\mathbf{x}_n)$$



Pattern recognition and machine learning, Chapter 9 p433, (Bishop, 2006)

- Back to the ML solutions for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:

$$\frac{\partial \ln p(\mathbf{X})}{\partial \boldsymbol{\mu}_k} = 0 \Longrightarrow \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)\mathbf{x}_n$$

$$\frac{\partial \ln p(\mathbf{X})}{\partial \boldsymbol{\Sigma}_k} = 0 \Longrightarrow \boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

$$\frac{\partial \ln p(\mathbf{X})}{\partial \pi_k} = 0 \Longrightarrow \pi_k = \frac{N_k}{N}$$

$$N_k = \sum_{n=1}^{N} \gamma(z_k);\ \gamma(z_k) = p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- Because $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k, $\gamma(z_k)$ all depend on each other, there is no analytical solution.
- We resort to a powerful optimization algorithm - **Expectation Maximization (EM)**.

EM algorithm for MoG :

1. Initialize $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k.

EM algorithm for MoG :

1. Initialize $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k.
2. **E-step**. Evaluate the responsibilities $\gamma(z_k)$ using $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:

$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2}$$

EM algorithm for MoG :

1. Initialize $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k.
2. **E-step**. Evaluate the responsibilities $\gamma(z_k)$ using $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:

$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2}$$

3. **M-step**. Re-estimate $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k based on the ML solutions:

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)\mathbf{x_n} \tag{3}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)(\mathbf{x} - \boldsymbol{\mu}_k^{new})(\mathbf{x} - \boldsymbol{\mu}_k^{new})^T \tag{4}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{5}$$

where $N_k = \sum_{n=1}^{N} \gamma(z_k)$.

EM algorithm for MoG :

1. Initialize $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k.
2. **E-step**. Evaluate the responsibilities $\gamma(z_k)$ using $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:

$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2}$$

3. **M-step**. Re-estimate $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k based on the ML solutions:

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)\mathbf{x_n} \tag{3}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)(\mathbf{x} - \boldsymbol{\mu}_k^{new})(\mathbf{x} - \boldsymbol{\mu}_k^{new})^T \tag{4}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{5}$$

   where $N_k = \sum_{n=1}^{N} \gamma(z_k)$.

4. Evaluate the log likelihood:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{6}$$

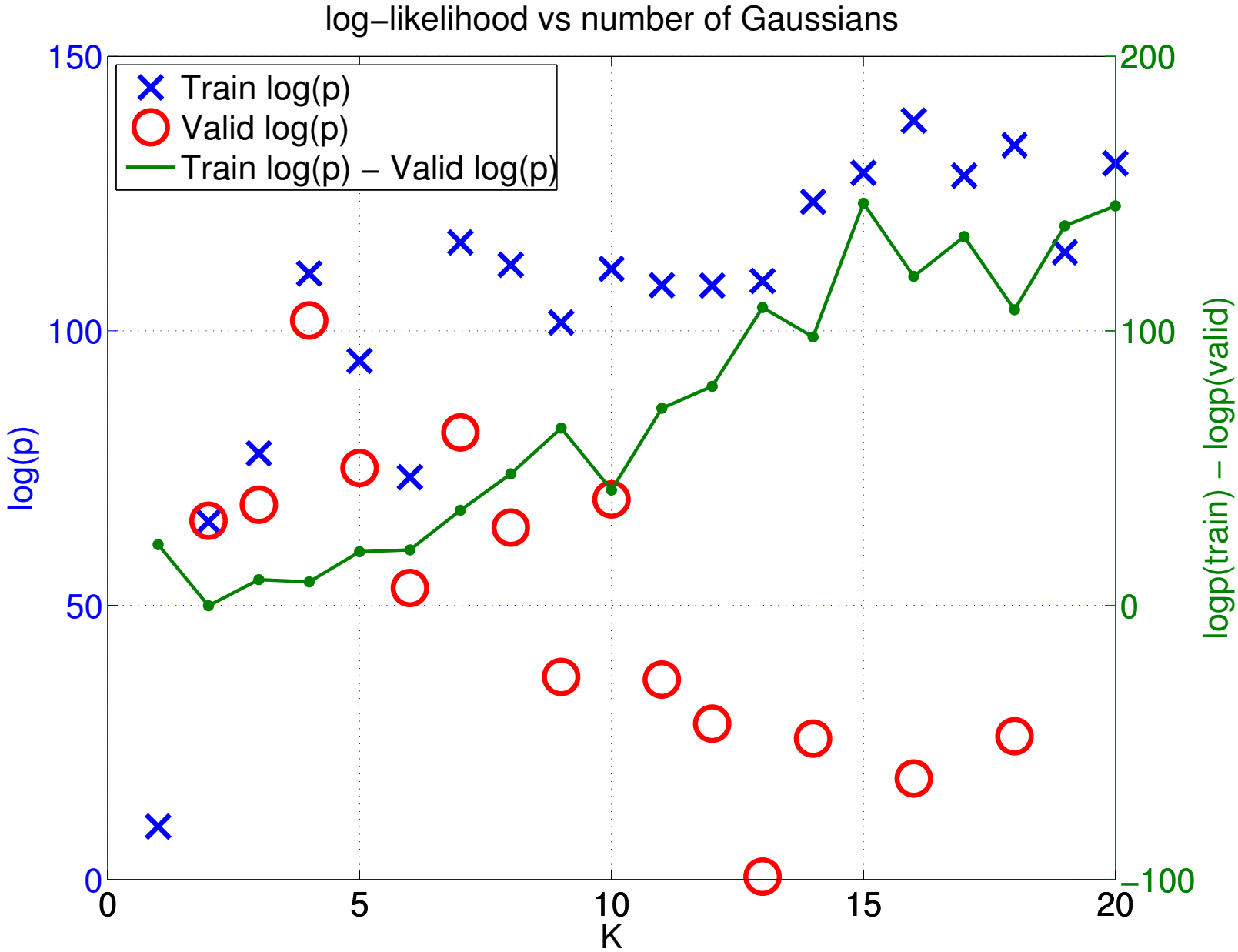
Assignment 3

- This assignment is about fitting mixtures of axis-aligned Gaussians to two-dimensional data. First copy and unzip the archive below into your directory.
- `http://www.cs.toronto.edu/~bonner/courses/2014s/csc321/assignments/hw3_matlab.zip`

Assignment 3 PART 1 (5 points)

- Run `moginit` to create training and validation datasets from 4 random Gaussians.
- Then use the function `mogem` to fit various numbers of Gaussians to the training data.
- Using performance on the *validation data*, determine the optimal number of Gaussians to fit to the training data.
- Present your results as a graph that plots both the validation density and the training density as a function of the number of Gaussians.
- Include a brief statement of what you think the graph shows.
- Also include a brief statement about the effects of changing the initial standard deviation used in `mogem`.
- Please do not change the random seeds in `mogem` (this will produce different data).

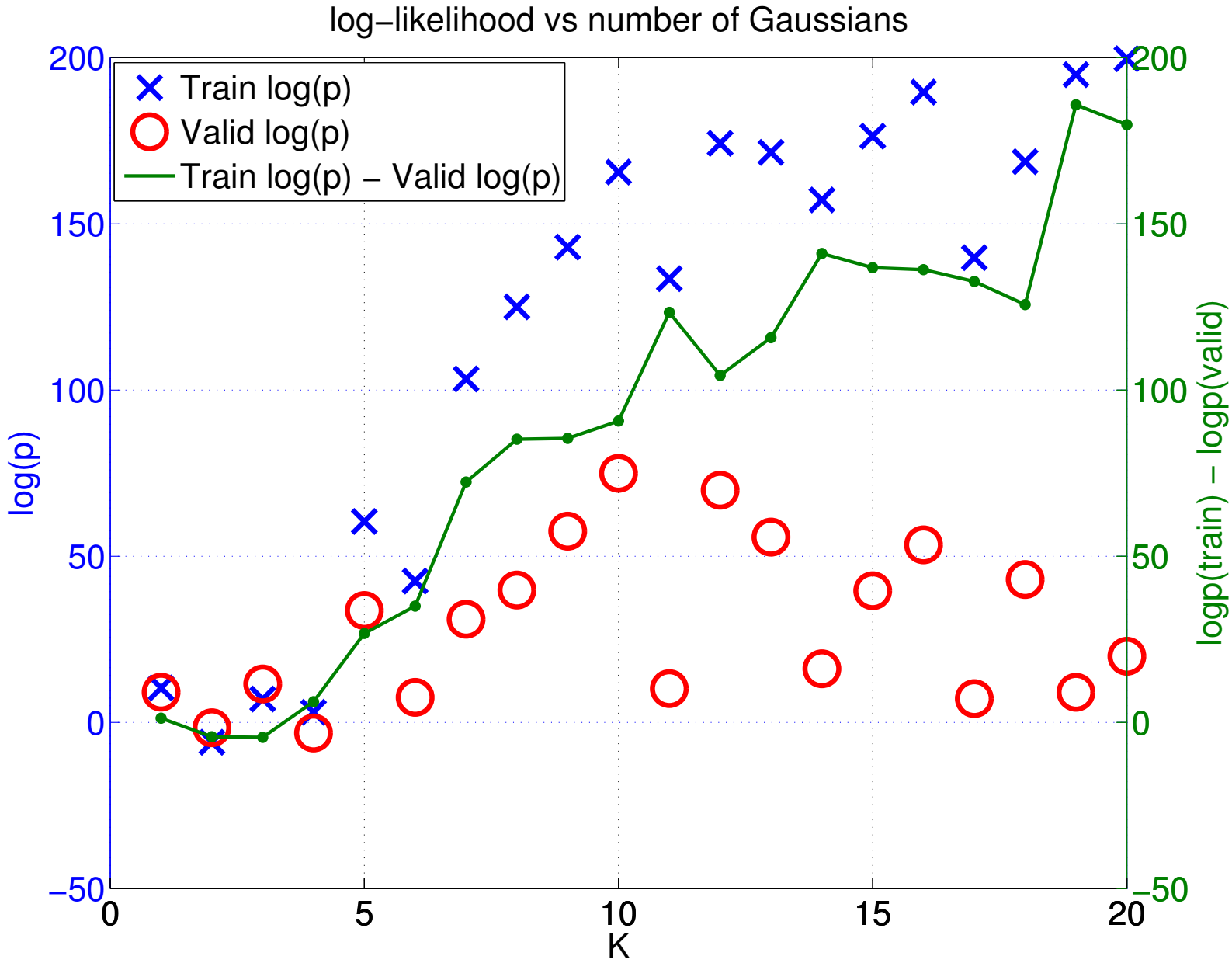
DEMO: Fitting 4 Gaussians



Training/validation generated from 4 Gaussians each with 30 cases (i.e., 120 cases in total)

Assignment 3 PART 2 (2 points)

- Change `moginit.m` to use only **12** cases per Gaussian, and **12** axis-aligned gaussians to generate the data and repeat the experiment above (without changing the random seeds).
- Present your results as a graph and include a brief statement of what you think the graph shows and why it differs from the graph in PART 1.



Training/validation generated from 12 Gaussians each with 12 cases (i.e., 144 cases in total)

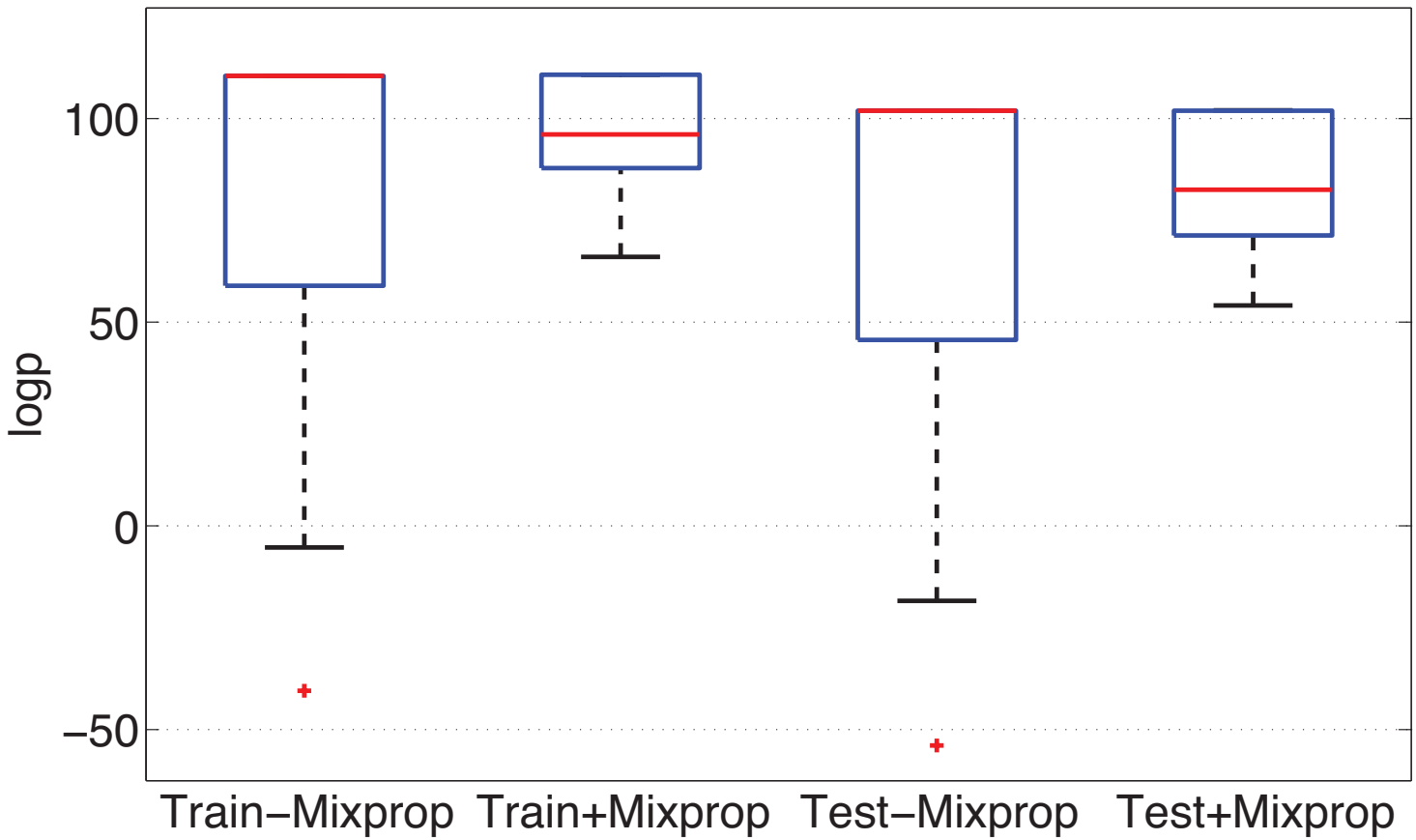
Assignment 3 PART 3 (3 points) - some programming

- Change `mogem.m` so that in addition to fitting the means and axis-aligned variances, it also fits the **mixing proportions** (Hint: Eq 5).
- Currently, `mogem` does not mention mixing proportions so it is currently assuming that they are all equal (which makes them all cancel out when computing the posterior probability of each Gaussian for each datapoint.).
- So the first thing to do is to include mixing proportions when computing the posterior, but keep them fixed (and not all equal).

Assignment 3 PART 3 (3 points) - some programming

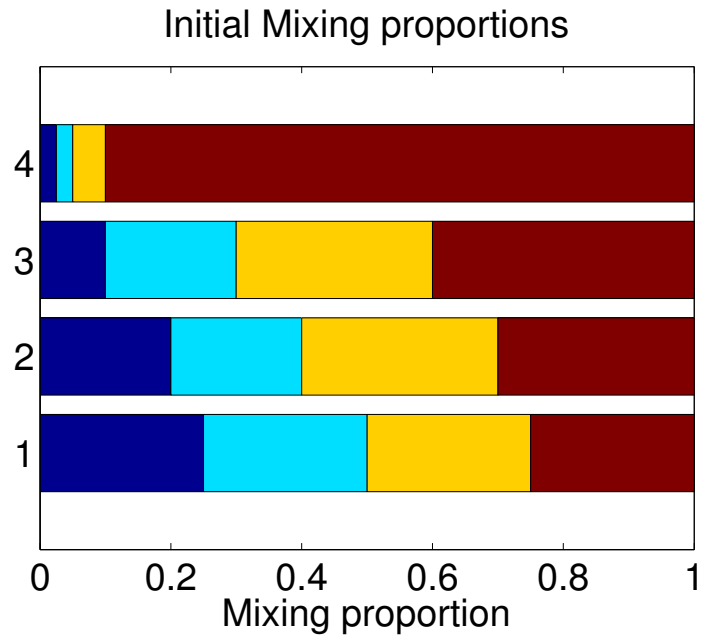
- Associate the code `mogem.m` with the EM algorithm in slide 27 or Lecture 14 (p9 - 12).
- Add/modify the code at the three commented places in `mogem.m`.

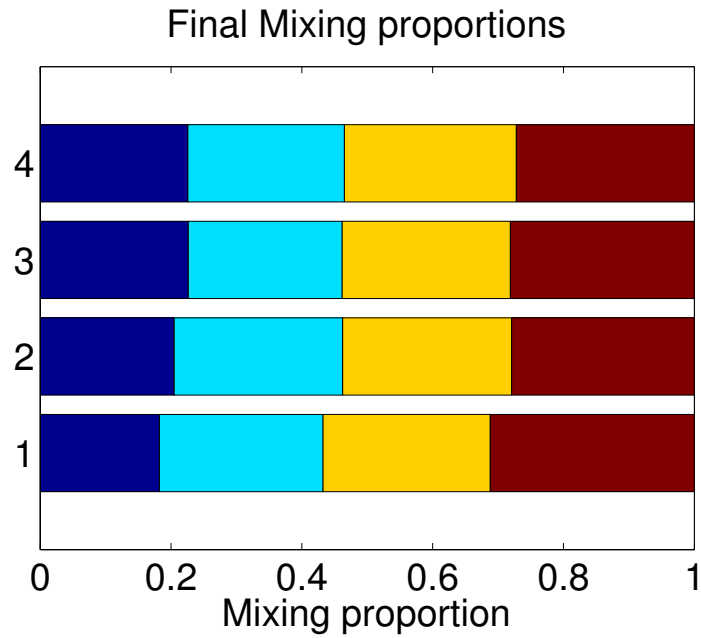
Improvement after adding mixing proportion



Initial and final mixing proportions

Once you have debugged this, try learning the mixing proportions. Make `mogem` print out the final mixing proportions and hand in several different examples of the final mixing proportions that you get when fitting 4 Gaussians to the data, when you start with mixing proportions [0.25, 0.25, 0.25,0.25], [0.3, 0.2, 0.2, 0.3], [0.1, 0.2, 0.3, 0.4] and [0.9, 0.025, 0.025, 0.05]. Do not change the data. **NB**: You may got different results if your `nIter, sdinit` are different from the those used below. Please report `nIter, sdinit` when reporting the final `mixprop` in your report.





Other initial parameters.: k=4; nIter = 20; sdinit = 0.15;

Use `barh` in Matlab to generate this plot.

ASSIGNMENT 3: due on March 18 at 3pm