

Announcements

Quiz next week!

Clarifications about the projects/final coursework

Github issues - wrong folder?

General questions?

Attendance

INM431: Machine Learning

Unsupervised learning, K-Means clustering and Mixture Models

Pranava Madhyastha (pranava.madhyastha@city.ac.uk)

Decomposition of the error

$$\underbrace{E_{\mathbf{x},y,D} \left[\left[h_D(\mathbf{x}) - y \right]^2 \right]}_{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[\left(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right)^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} \left[(\bar{y}(\mathbf{x}) - y)^2 \right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} \left[(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \right]}_{\text{Bias}^2}$$

Boosting

$$H_T(\vec{x}) = \sum_{t=1}^T \alpha_t h_t(\vec{x})$$

Iterative addition of weak learners: a learning algorithm that outputs a classifier that performs slightly better than chance.

At iteration t add a new weak learner $h_t(x)$ with a step-size α_t (a constant)

Test time, return the weighted sum!

Instead of learning a single classifier, learn many weak classifiers that are good at different parts of the input space

Original Boosting algorithm (Schapire 1990)

1. Train a first classifier f_1 on a training subset by sampling from the original train set.
2. Train a second classifier f_2 on a training subset by sampling from the original train set such that half of the samples are where f_1 makes mistakes and the other half the rest.
3. Train a third classifier f_3 on the a new training subset sampled from original train set such that it is made of samples where f_1 and f_2 disagree.

Original Boosting algorithm (Schapire 1990)

$$g = \text{MajorityVote}(f_1, f_2, f_3)$$

Weak learning assumption: trained weak classifiers are slightly better than random guessing.

The combination of these can provably attain arbitrarily small error $\rightarrow 0$.

There are several versions of boosting algorithms - AdaBoost, GradientBoostingMachines, XGboost and many others

Unsupervised learning

Bird's eye view:

Data has lots of rich hidden structures!

We want methods that can discover this structure automatically.

There are many forms of unsupervised learning that help in revealing different latent or hidden structures from the data

In this course, we will look into K-Means and Gaussian Mixture Models

Clustering

A form of unsupervised learning

Especially useful when we don't really know what exists in the data

- exploratory data analysis
- detecting anomalies/exceptions

Input: samples without any labels

Output: Patterns

Example: document clustering

Input: large number of medical documents (100,000)

Output:

cluster	size	top_words
1	23	risk, health, diabetes, intervention, treatment
2	4	hiv, inflammation, env, testing, study
3	38	cell, infection, determine, cells, function
4	7	research, program, cancer, disparities, students
5	8	cancer, brain, imaging, tumor, metastatic
6	3	microbiome, crc, gut, psoriasis, gut_microbiome
7	3	cdk, nmdar, nmdars, calpain, nrs
8	7	research, core, center, support, translational
9	3	lung, ipf, expression, cells, methylation
10	4	mitochondrial, metabolic, redox, ros, bde

Example: document clustering

Input: large number of medical documents (100,000)

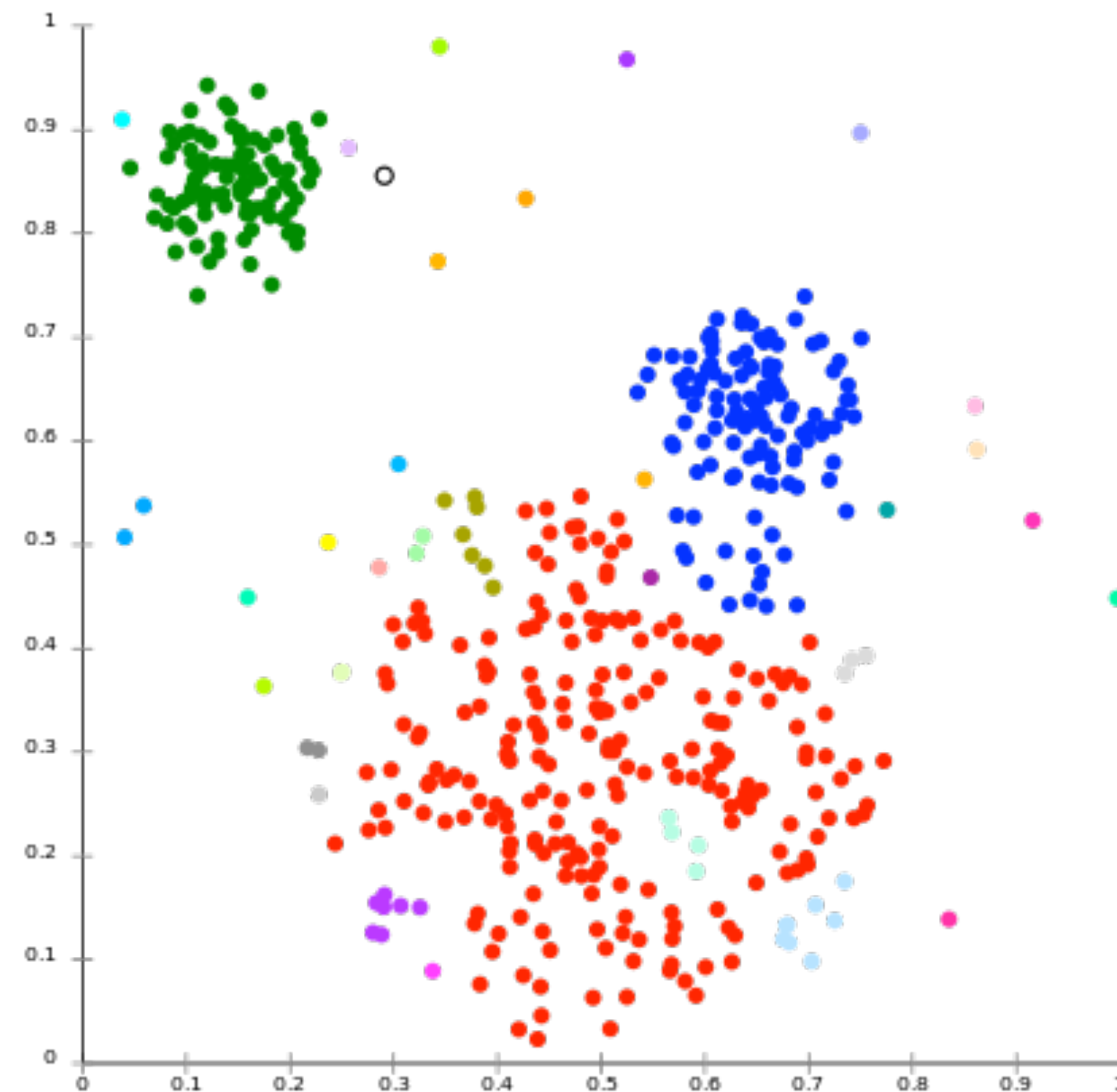
Output:

cluster	size	top_words
1	23	risk, health, diabetes, intervention, treatment
2	4	hiv, inflammation, env, testing, study
3	38	cell, infection, determine, cells, function
4	7	research, program, cancer, disparities, students
5	8	cancer, brain, imaging, tumor, metastatic
6	3	microbiome, crc, gut, psoriasis, gut_microbiome
7	3	cdk, nmdar, nmdars, calpain, nrs
8	7	research, core, center, support, translational
9	3	lung, ipf, expression, cells, methylation
10	4	mitochondrial, metabolic, redox, ros, bde

Example: clustering gene expression

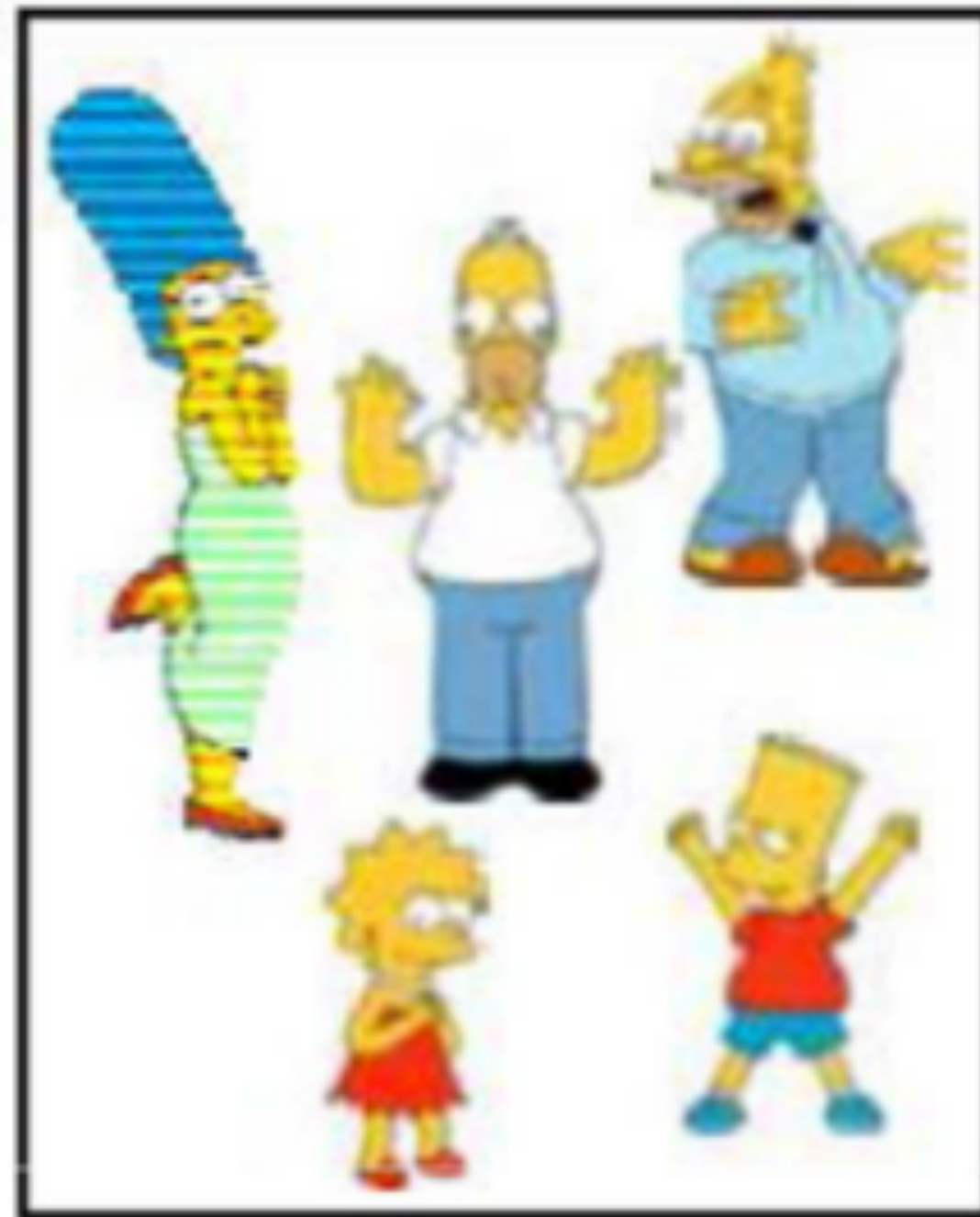
Input: Large number of transactions that includes fraudulent transactions

Output:



Idea

Cluster **similar** instances together!



Similarity

Popular: distance function!

$$\text{dist}(\text{instance}_1, \text{instance}_2) = || \text{instance}_1 - \text{instance}_2 ||_2^2$$

The final results are dependent on the type of similarity function being used!

Clustering with K-Means

Basic intuition: Take a set of data points as input and return a partitioning of the points into K clusters.

- We want similar points to be in the same cluster and dissimilar points to be in different clusters.

Clustering with K-Means

A flat clustering algorithm

Iterative clustering algorithm:

- Pick K random points as cluster centres
- Iterate:
 - 1) Assign data-points to the closest cluster centre
 - 2) Change the cluster centre to the average of its assigned points

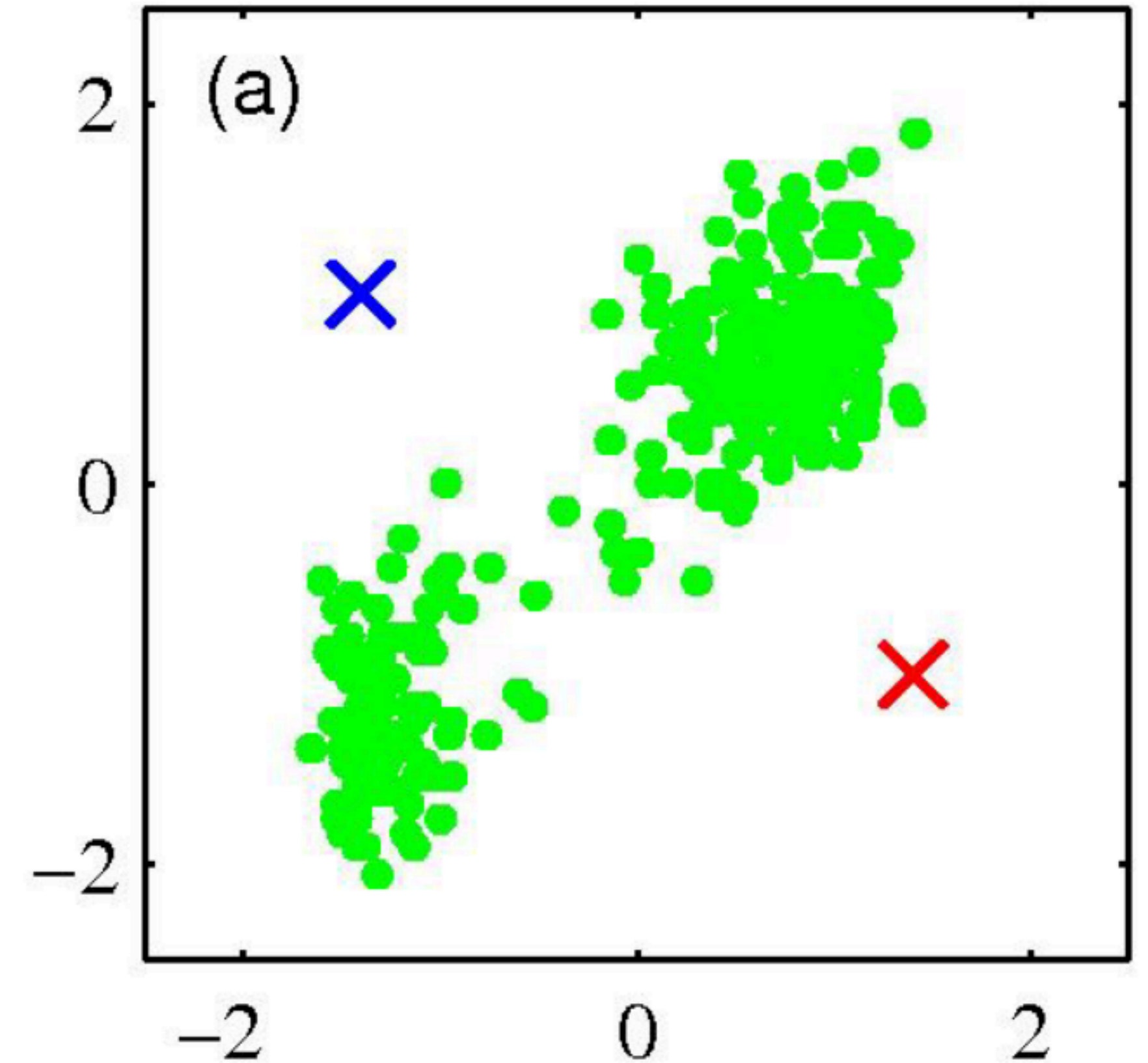
Stop when the assignments stagnate

Example: initialise

Given: a large number of points

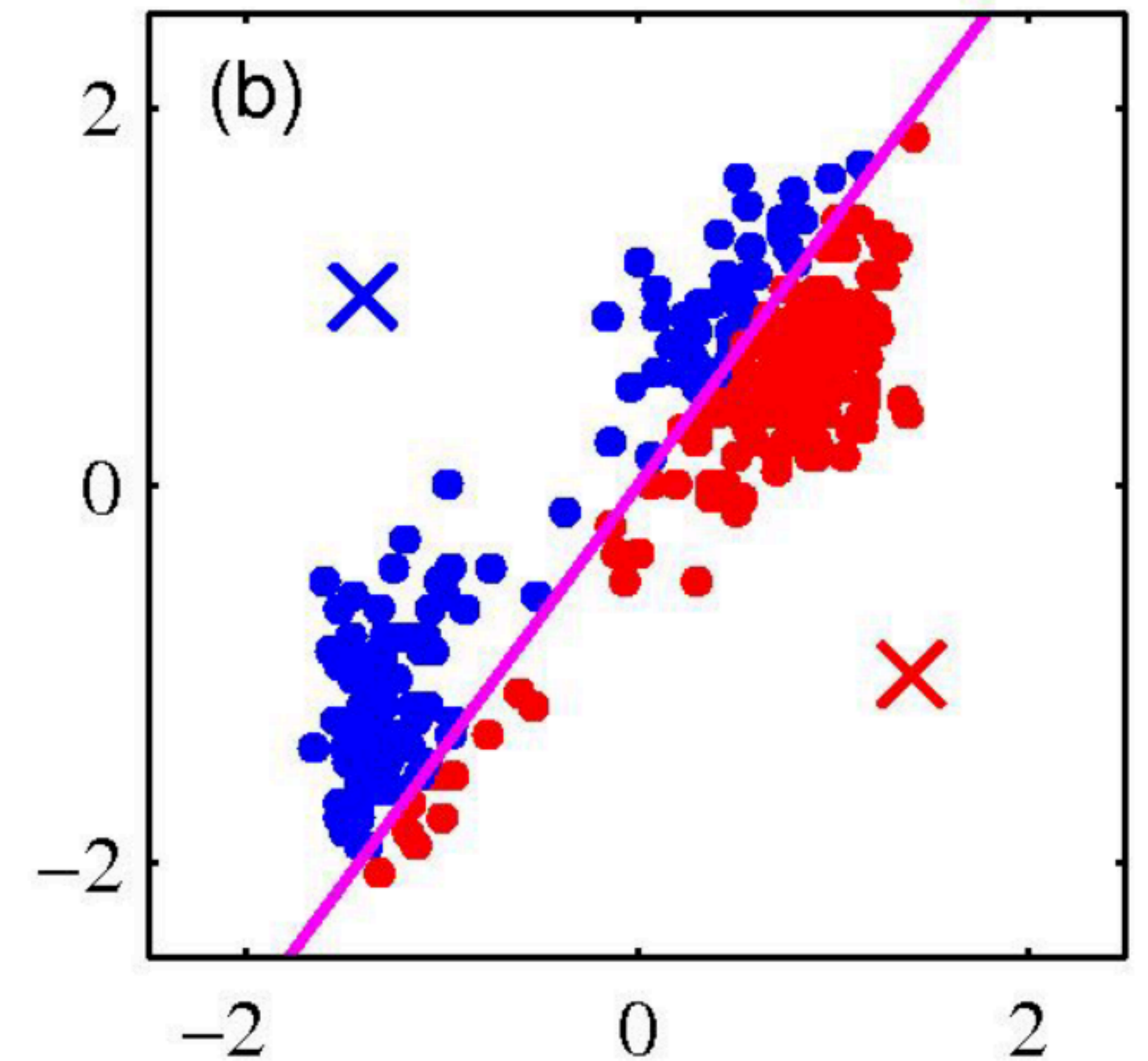
- Choose $K=2$ random points as cluster centres

Cluster centre \approx Mean!



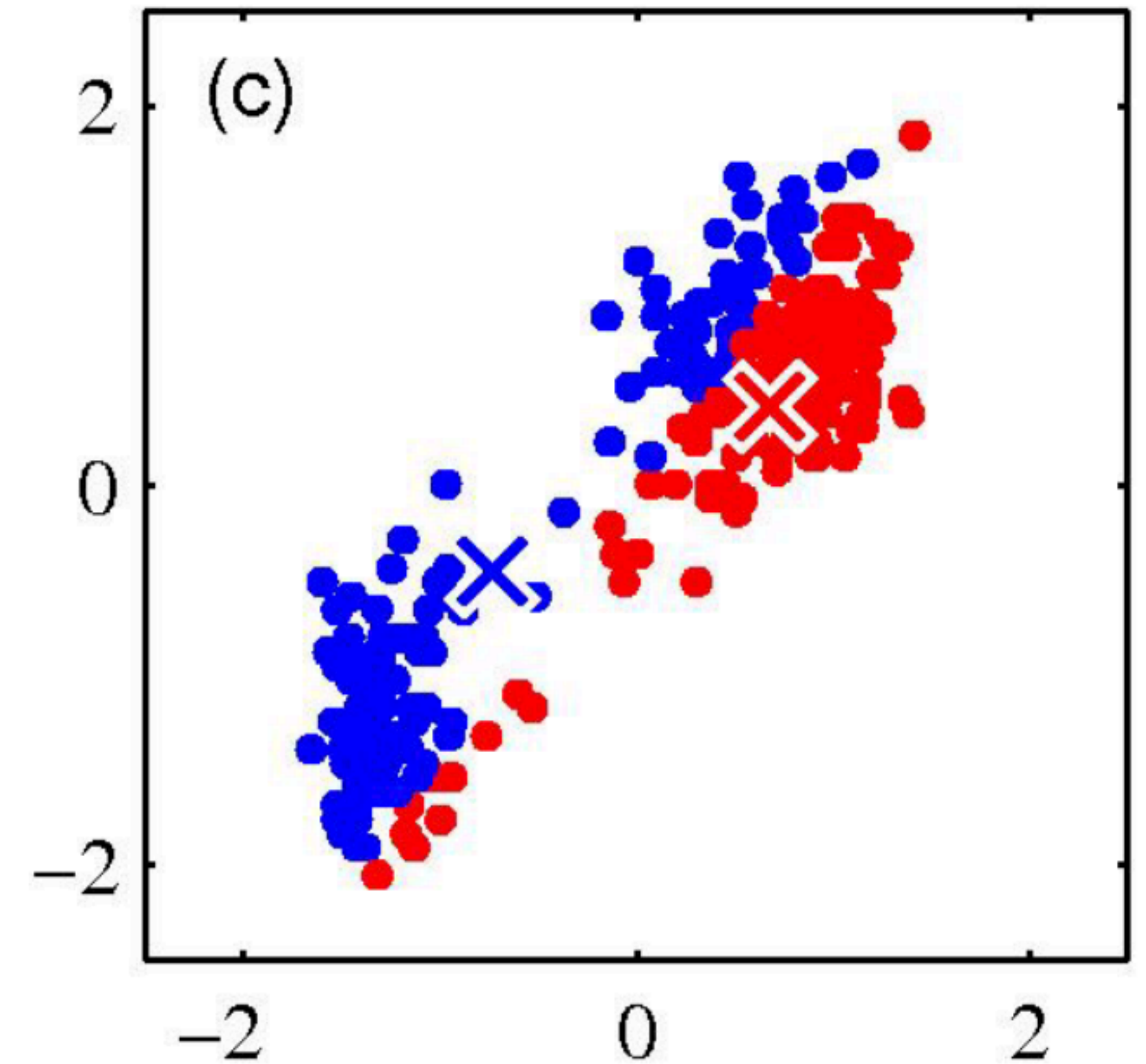
Example: iteration - step 1

Assign data points to the closes cluster centre



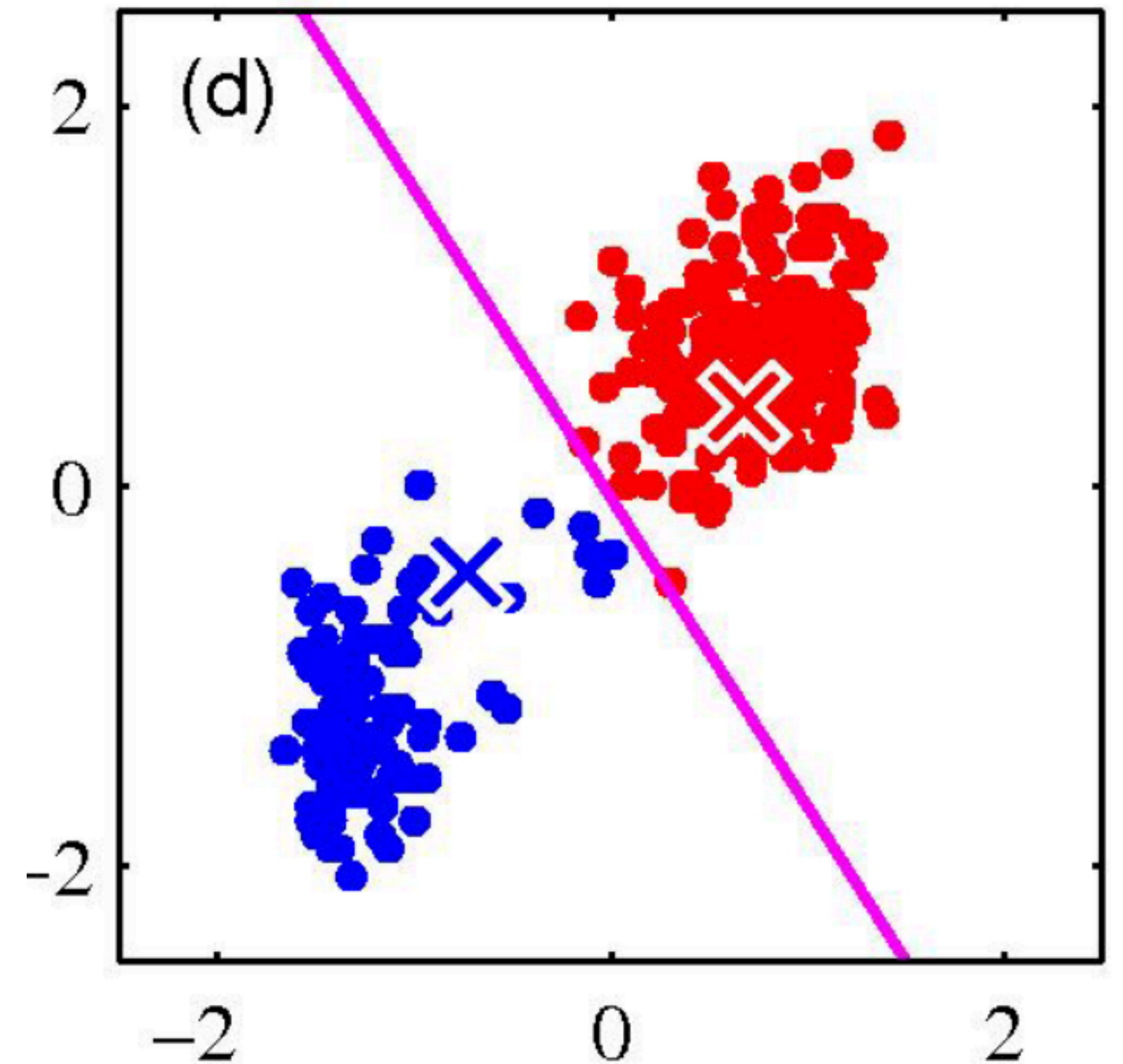
Example: iteration - step 2

Change cluster centre to the mean of the assigned points



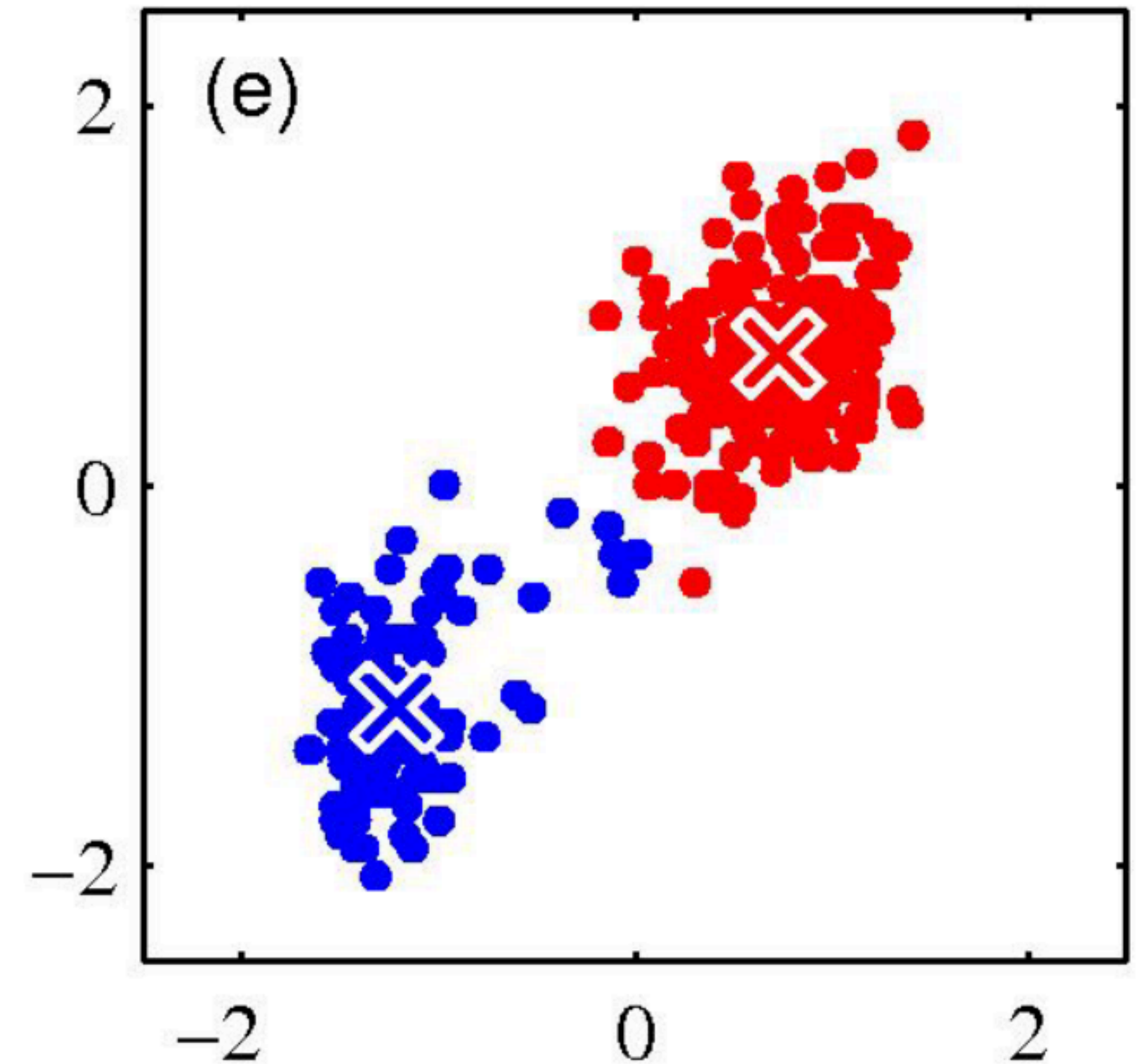
Example: iterate

Repeat until the assignments stagnate
(convergence criteria)



Example: finally

At convergence, we have $K=2$ distinct clusters.



Formally

Input: data points: $\mathcal{D} = \{x_1, \dots, x_n\} \in \mathbb{R}^d$

Output: assignment of each of the datapoint to a cluster $\{1, \dots, K\}$

Each cluster is represented by a centroid μ_k

Define $r_{nk} \in \{0, 1\}$ a binary indicator variable (indicates the cluster assignment of data point x_n)

We use an **alternative minimisation** objective:

$$\min_{\mu} \min_r \sum_{i=1}^n \sum_{k \in K} r_{nk} |x - \mu_i|^2$$

Formally: alternating minimisation

Step 1: Fix cluster centres μ_k and optimise cluster assignments r_{nk} :

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

Step 2: Fix r_{nk} and optimise for μ :

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

μ_k is the mean of the k^{th} cluster hence the term 'K- Means'

Alternating minimisation



A chicken or egg problem!

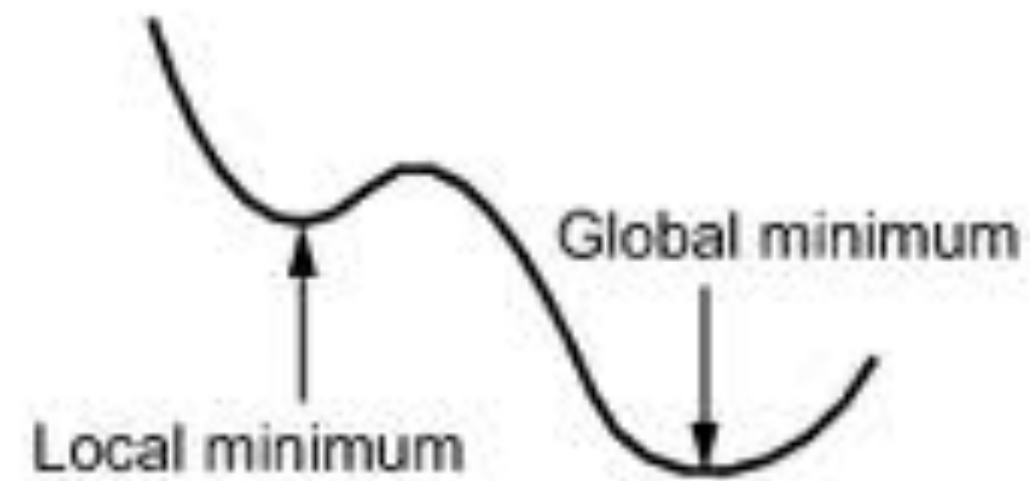
$$\min_{\mu} \min_r \sum_{i=1}^n \sum_{k \in K} r_{nk} |x - \mu_i|^2$$

Tackle a hard problem by solving two easy problems!

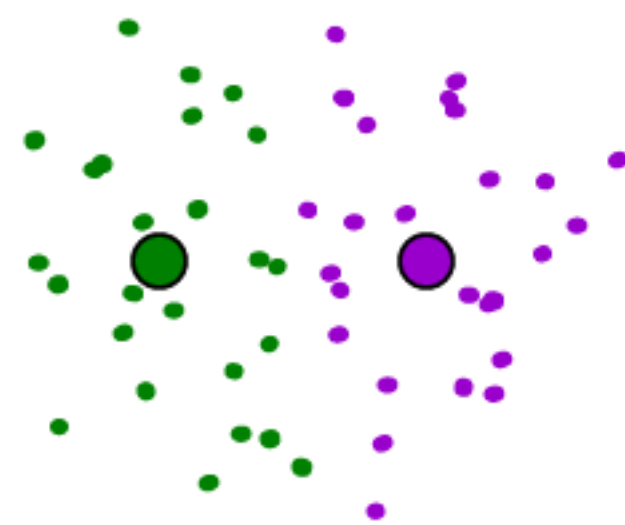
Each step is guaranteed to decrease the objective - thus it will converge!

Issues?

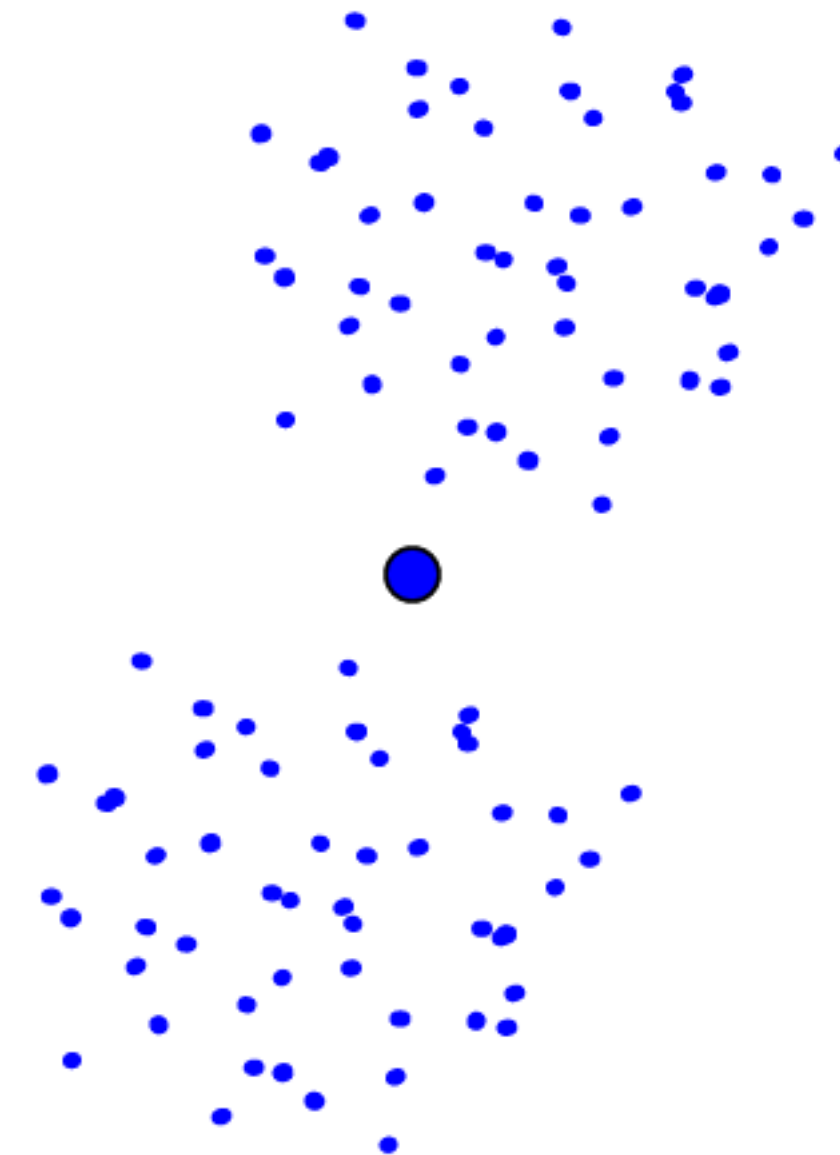
It will converge, but not always to the global minimum!



Consider



vs



Issues?

Further: initialisation can be extremely influential towards better solutions - heuristic driven!

Different similarity functions can give different results!

Exercise

You have the following set of 2-dimensional data points:

$\{1.9, 1.9\}$, $\{0.9, 1.1\}$, $\{1.8, 2.0\}$, $\{0.8, 1.0\}$, $\{1.1, 0.9\}$, $\{2.0, 1.9\}$, $\{1.0, 0.9\}$, $\{1.9, 1.8\}$

Apply K-means with $K=2$ clusters using the following initial values for cluster prototypes: $\mu_1 = \{1.0, 1.0\}$ and $\mu_2 = \{2.0, 2.0\}$.

Which are the values of the final prototypes for each cluster?

To which cluster does each data point belong to?

Solution

Step 1: we already have the initialisation!

Alternative minimisation:

Step 2: fix μ , estimate the assignments:

$$r_1 = \{0, 1, 0, 1, 1, 0, 1, 0\}; r_2 = \{1, 0, 1, 0, 0, 1, 0, 1\}$$

Step 3: fix assignments, estimate μ

$$\mu_1 = \frac{(x_2 + x_4 + x + 5 + x_7)}{4} = \{0.95, 0.975\}; \mu_2 = \frac{(x_1 + x_3 + x_6 + x_8)}{4} = \{1.9, 1.9\}$$

Iterate again:

Step 2: fix μ , estimate the assignments:

$$r_1 = \{0, 1, 0, 1, 1, 0, 1, 0\}; r_2 = \{1, 0, 1, 0, 0, 1, 0, 1\}$$

Step 3: fix assignments, estimate μ

$$\mu_1 = \frac{(x_2 + x_4 + x + 5 + x_7)}{4} = \{0.95, 0.975\}; \mu_2 = \frac{(x_1 + x_3 + x_6 + x_8)}{4} = \{1.9, 1.9\}$$

Convergence! We have found the corresponding cluster assignments and the mean values!

More

<https://shabal.in/visuals/kmeans/1.html>

Latent variables!

Assumption of latent constraints

Probabilistic models with hidden (i.e. non-observed) variables are also known as latent variable models (LVMs).

These are very powerful for extracting regularities and abstractions from data.

These latent variables can also serve as a bottleneck, computing a compressed representation of the data.

A motivating example



Input

Bangs

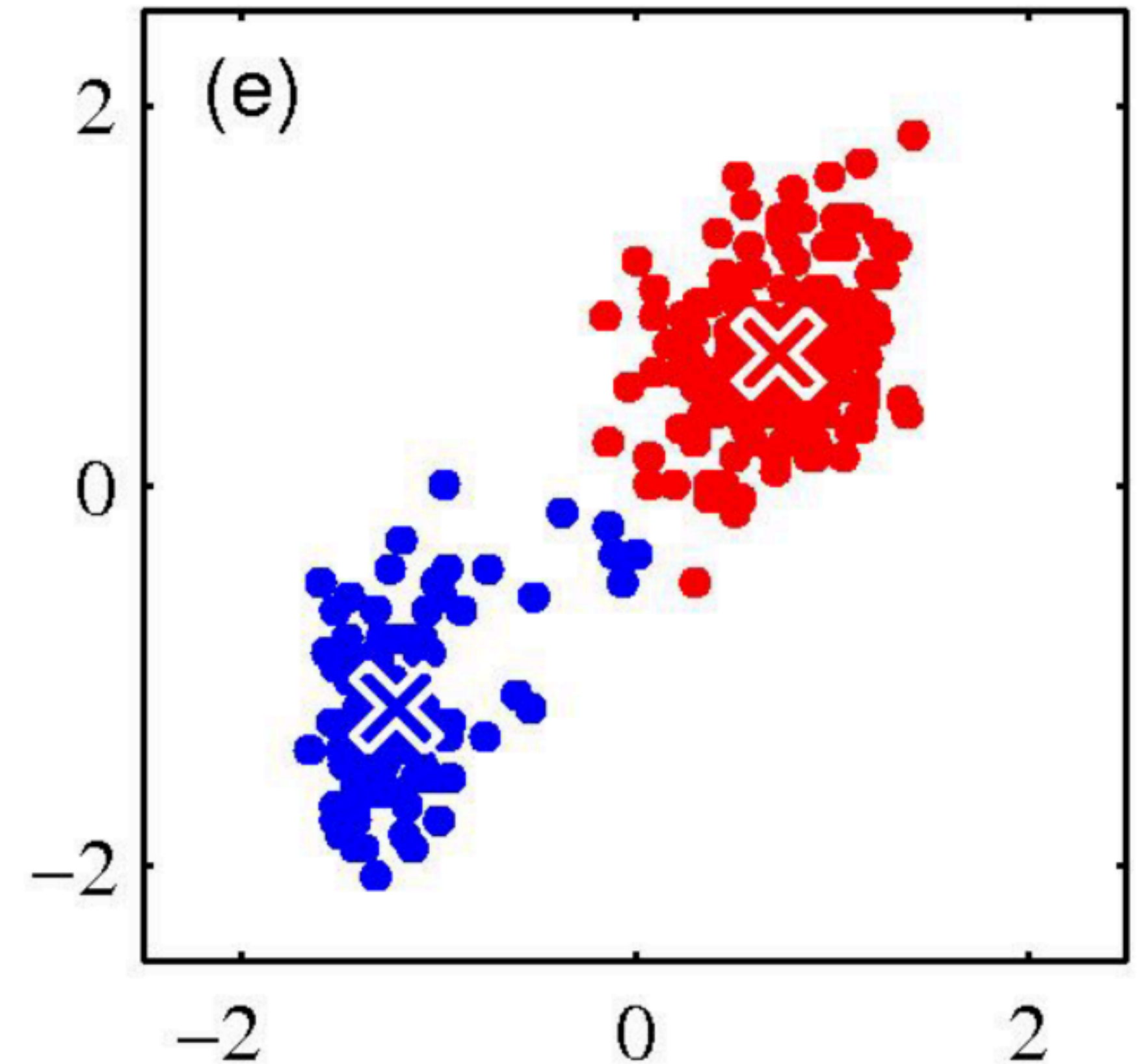
Blond Hair

Eyeglasses

Back to K-Means

Hard assignments!

Problems?

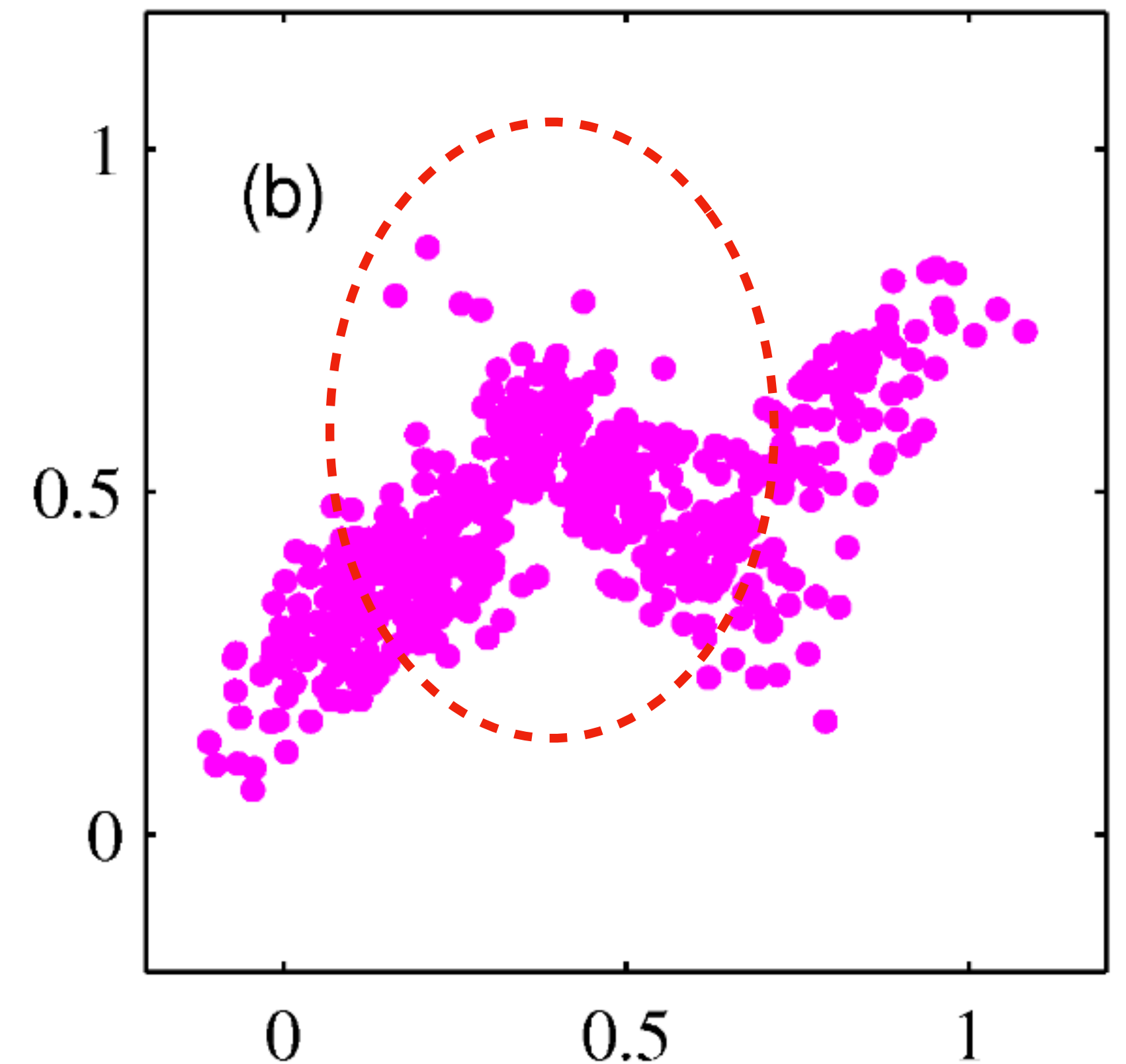


Problems with hard assignments

Clusters may overlap!

Some may be wider than others

Distances may not always be the right thing to use



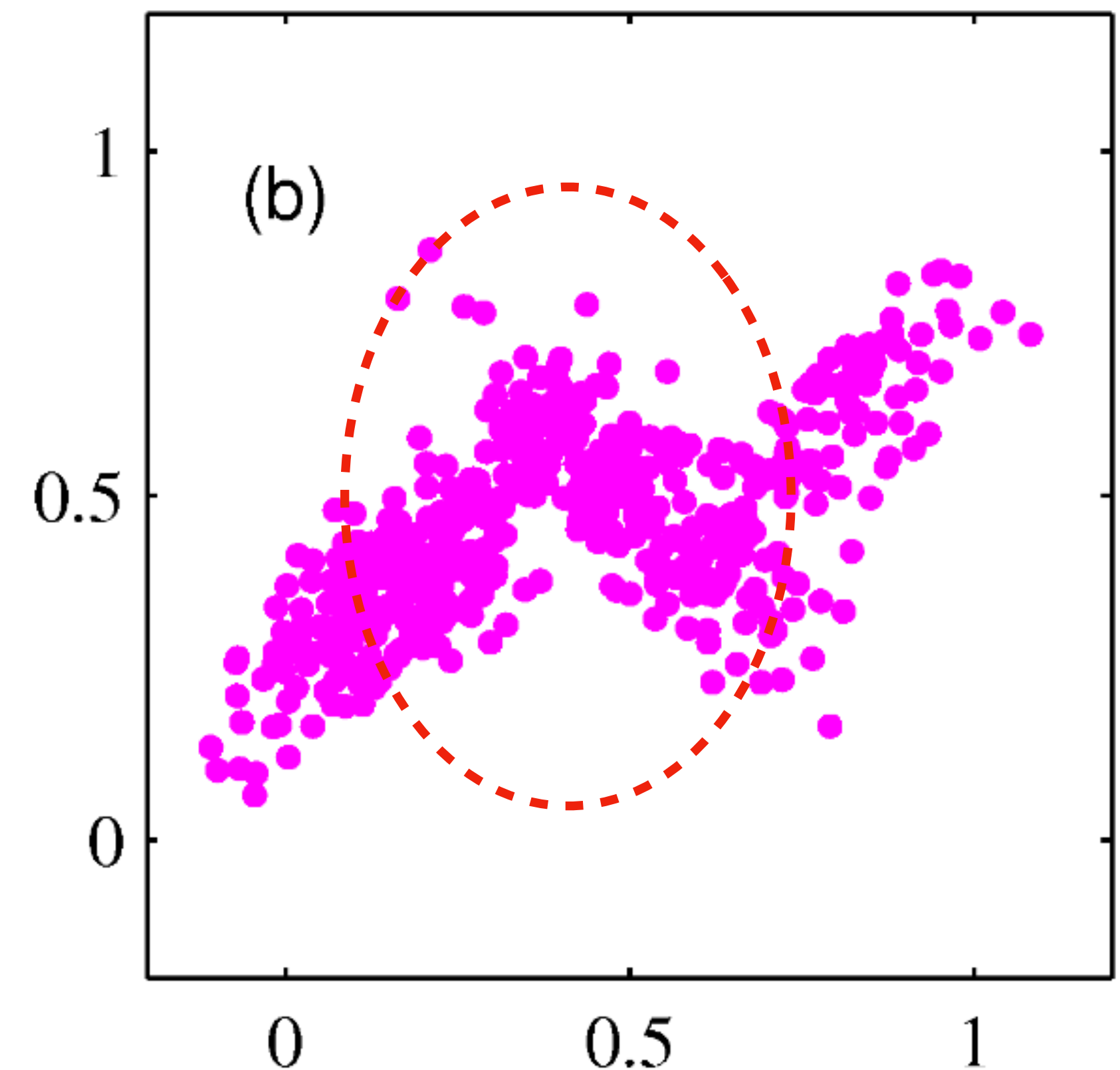
Probabilistic Clustering

What if we had a probabilistic model?

- can help us give a notion of probability of a data point belonging to the set of clusters
- allows for overlap
- cluster sizes won't matter

Can provide a generative model for the data

So far we have studied generative models with labels Y ! Here we have no labels



Mixture models: Gaussian framework

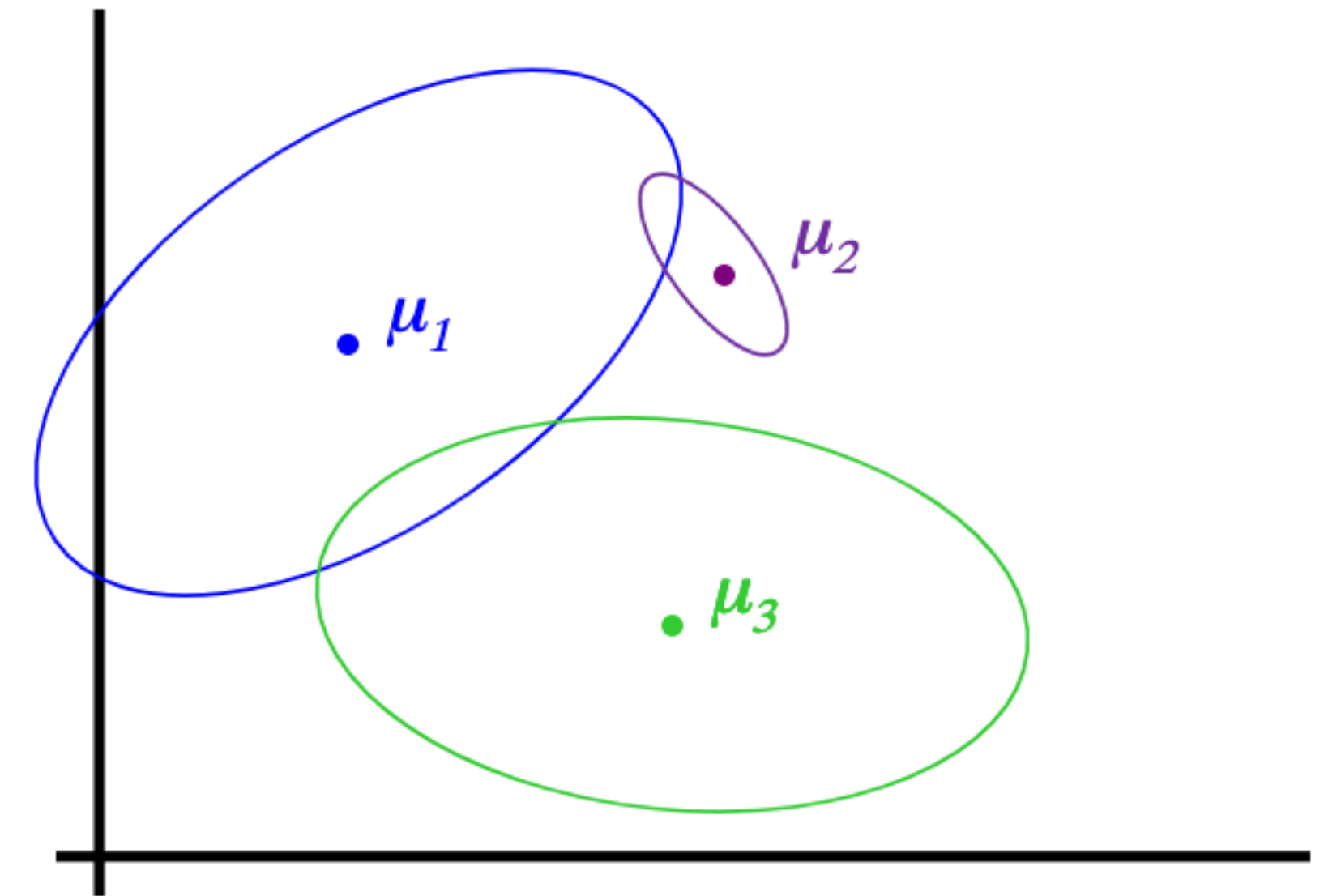
$P(Z)$: There are k components (latent labels Y)

$P(X | Z)$ = Each component generates data from a multivariate Gaussian with mean μ_i and covariance matrix Σ_i

Sample each point from a generative process with an assumption of Gaussian distribution:

- Choose component i with $P(Z = i)$
- Sample datapoint $\sim \mathcal{N}(\mu_i, \Sigma_i)$

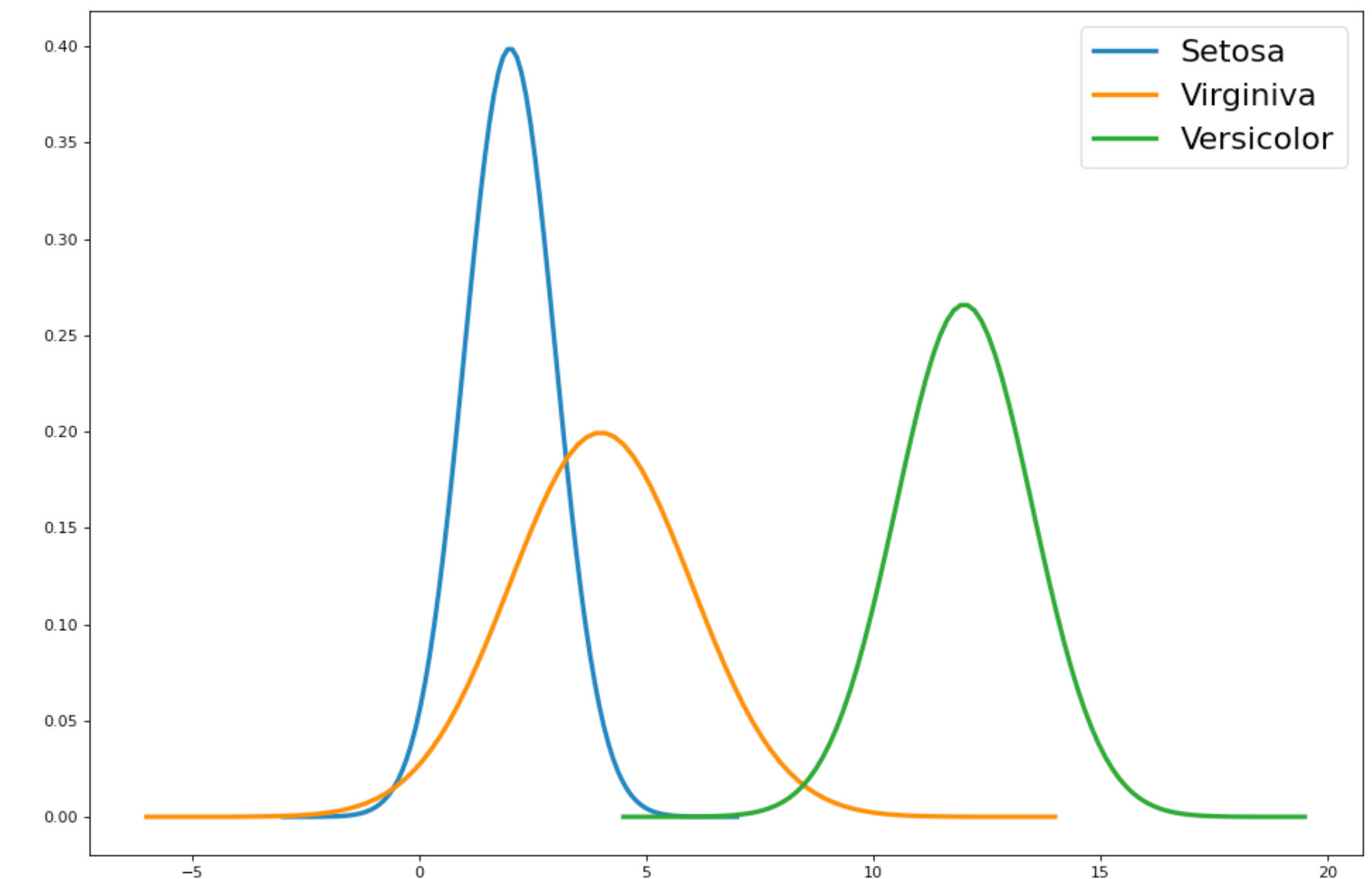
This is called a Gaussian Mixture Model (GMM)



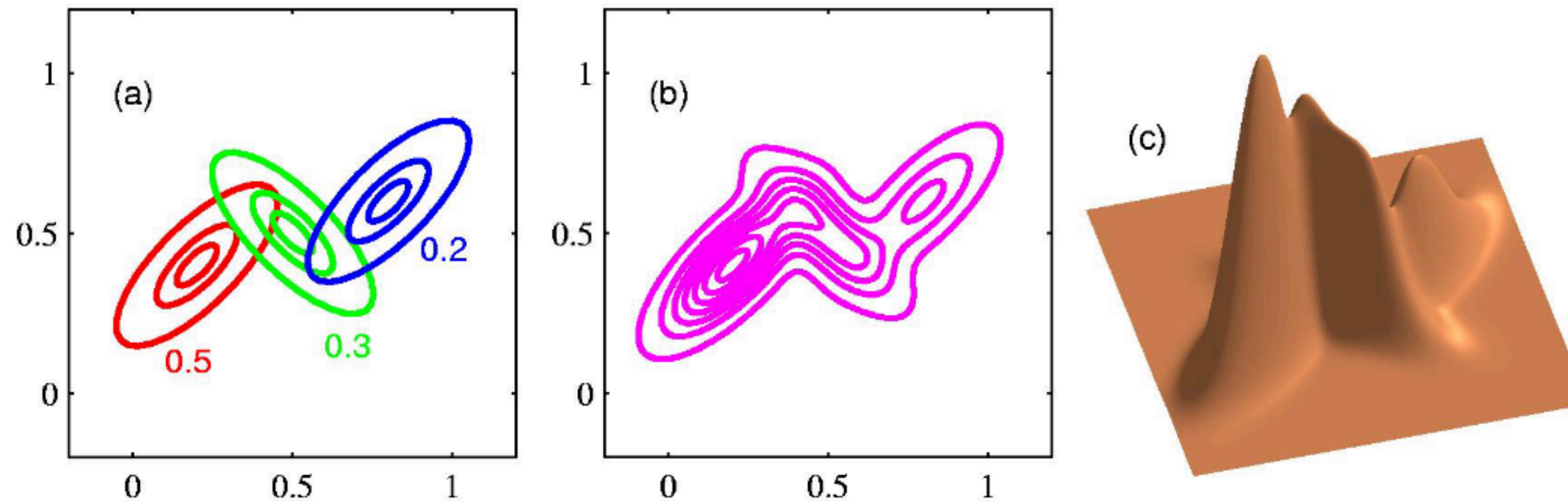
We have seen something similar before

Gaussian Naïve Bayes

- Multinomial over ' Y ' (the original labels)
- **INDEPENDENT** Gaussians for each x_{ij} (feature) given Y



Let's relax the assumption!



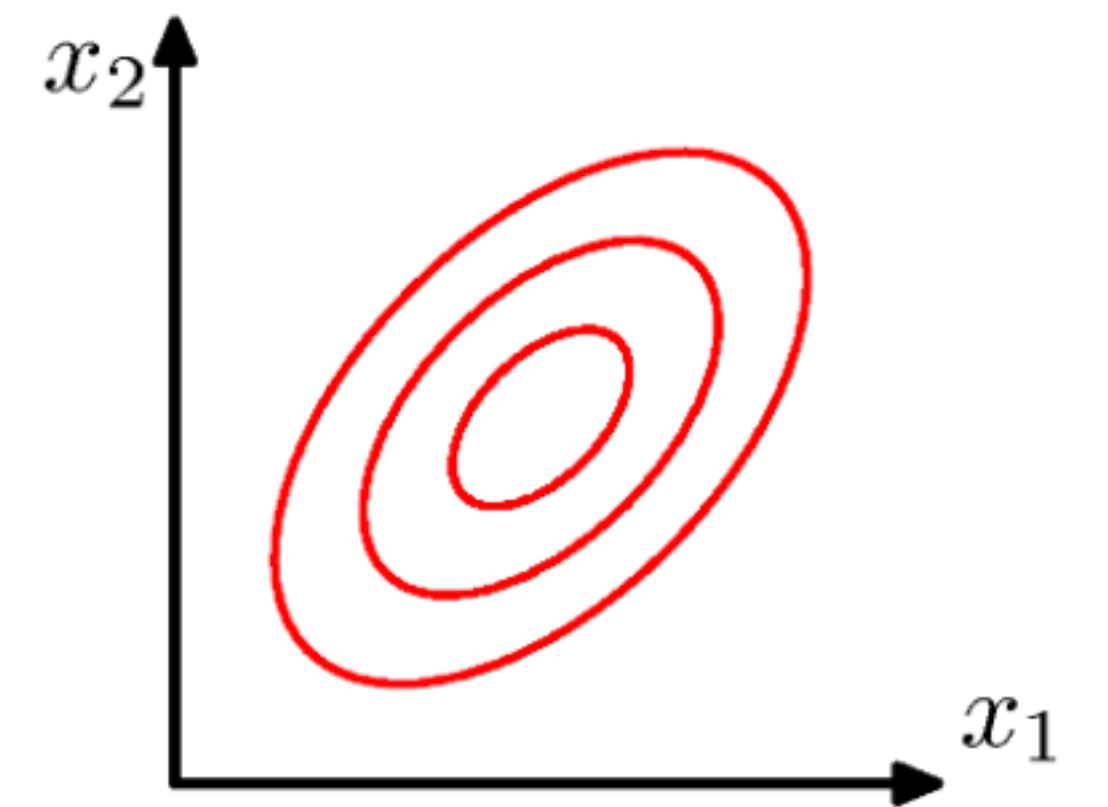
That is, no feature independence assumption!

Let's focus directly on the data point x_i and let x_{ij} co-vary

We will now use a multivariate Gaussian distribution, instead of a univariate Gaussian distribution!

Multivariate Gaussian distribution

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$



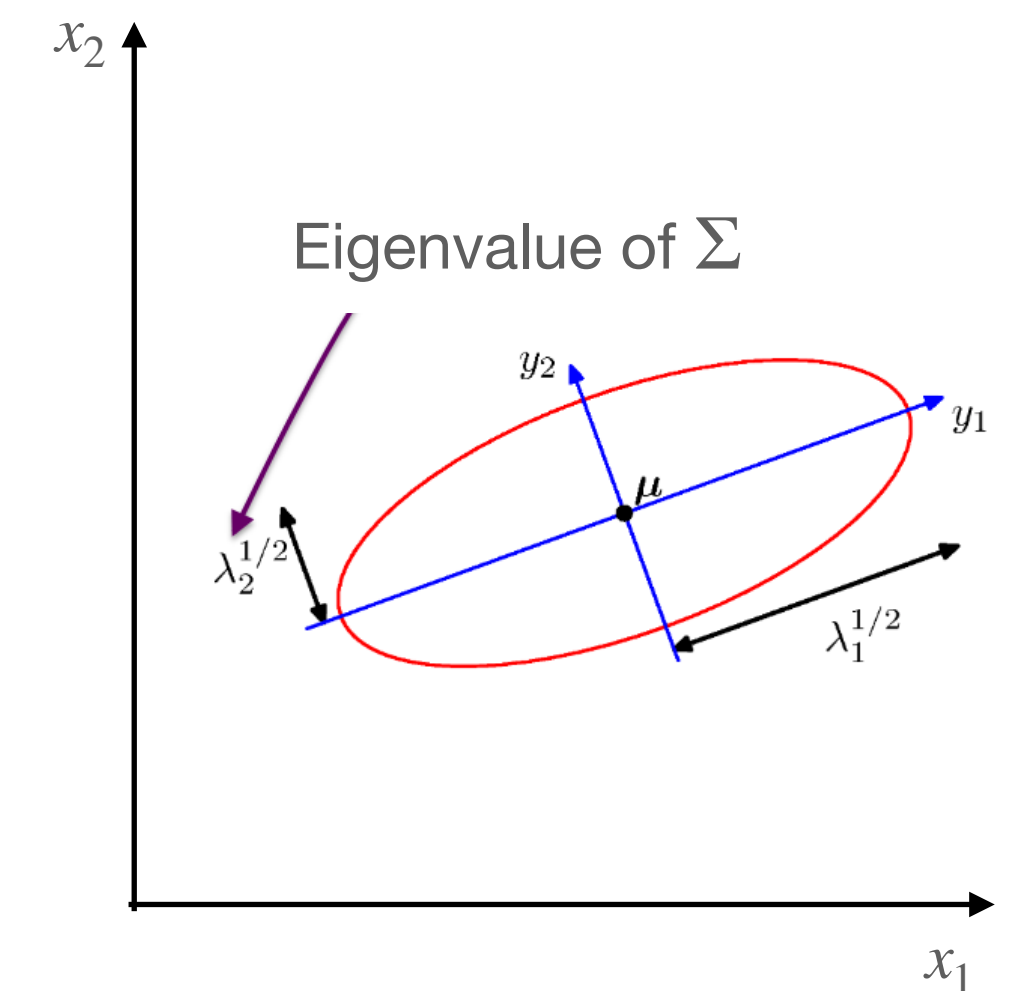
where, D are the set of dimensions or the features.

Multivariate Gaussian distribution

Covariance matrix, Σ , =
degree to which x_j vary
together

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

where, D are the set of dimensions or the features.



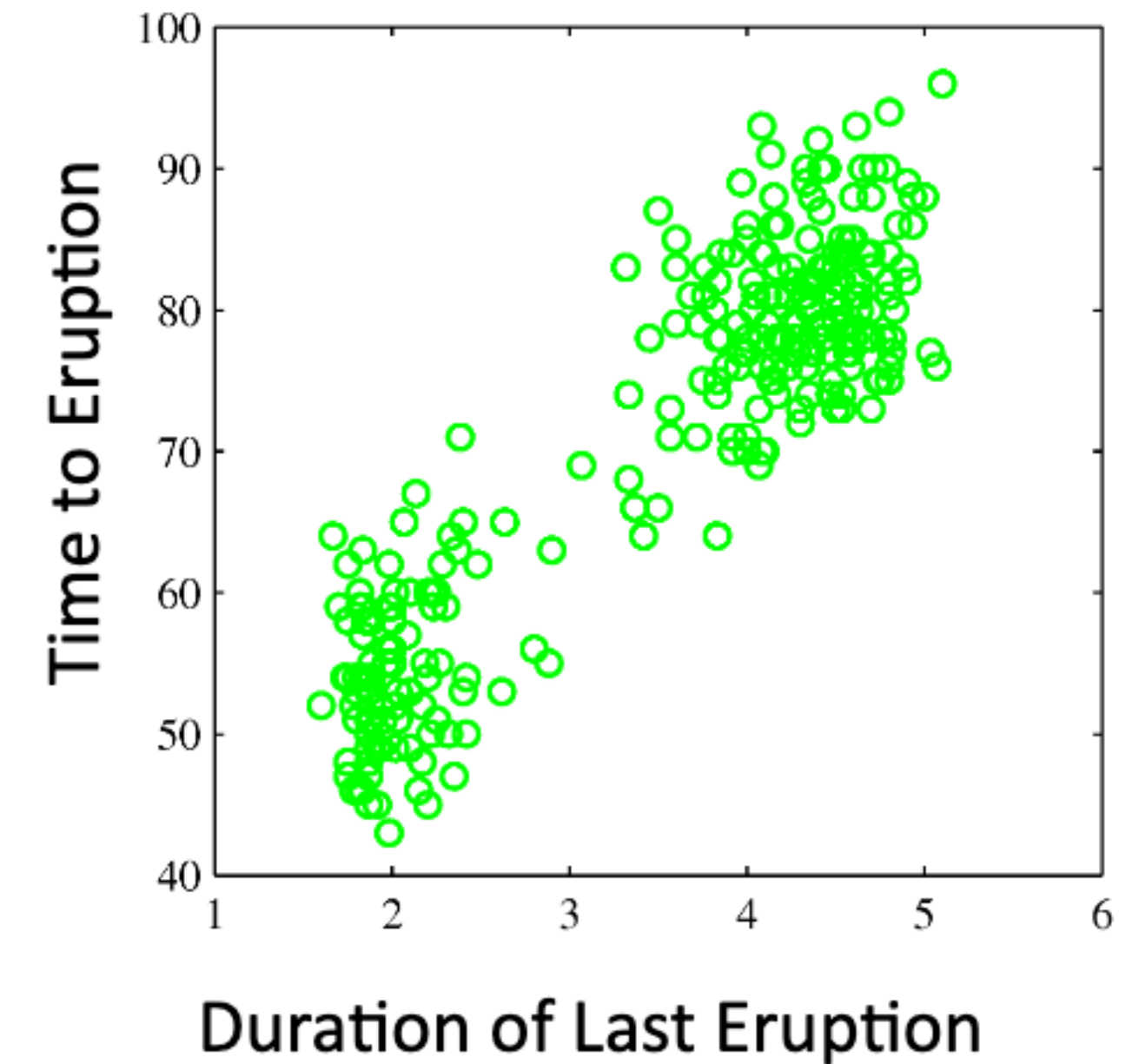
Mixture of Gaussians

Always Multivariate Gaussians!

Allows for multiple peaks

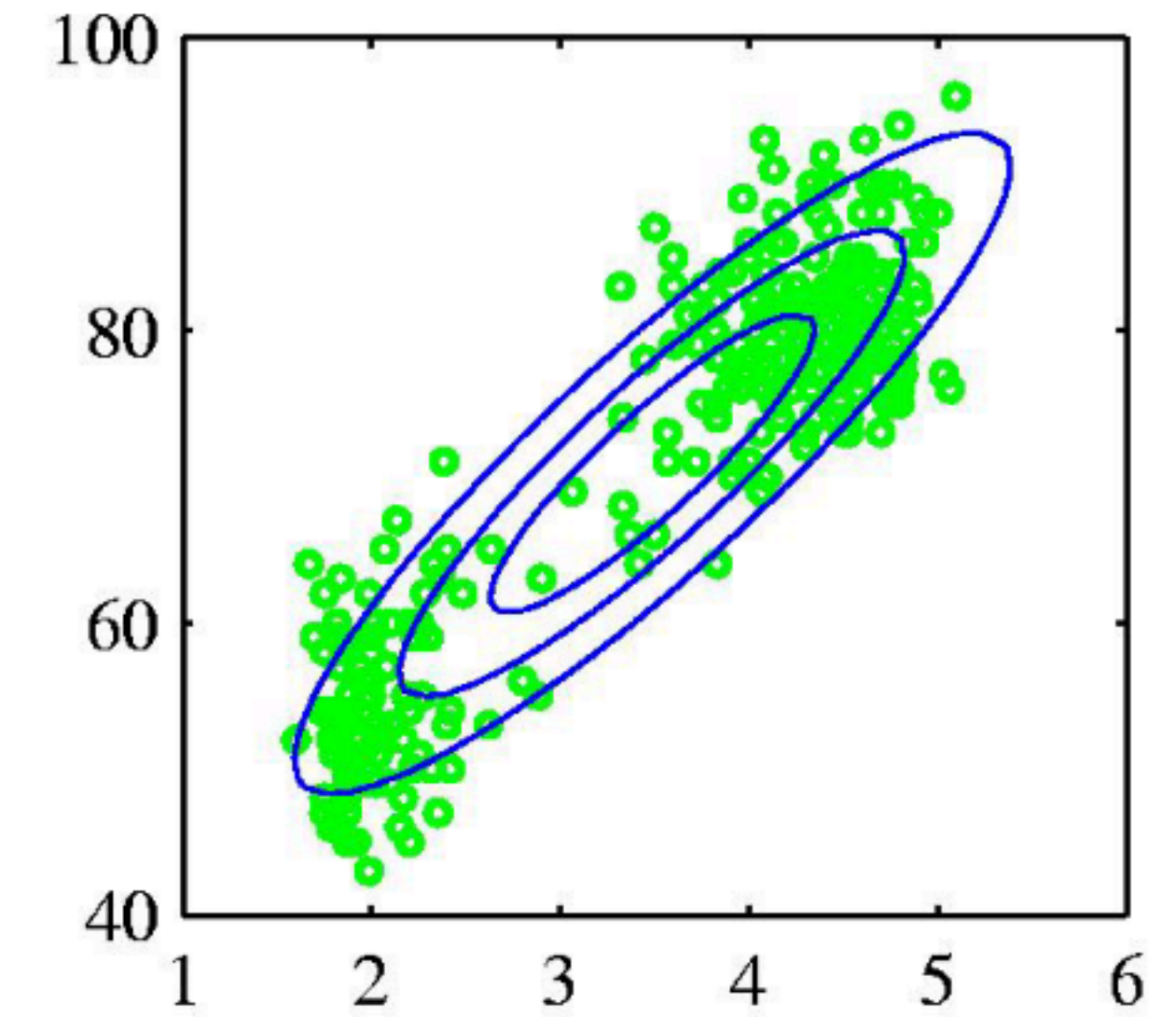
Lab assignment:

The 'old faithful' geyser dataset

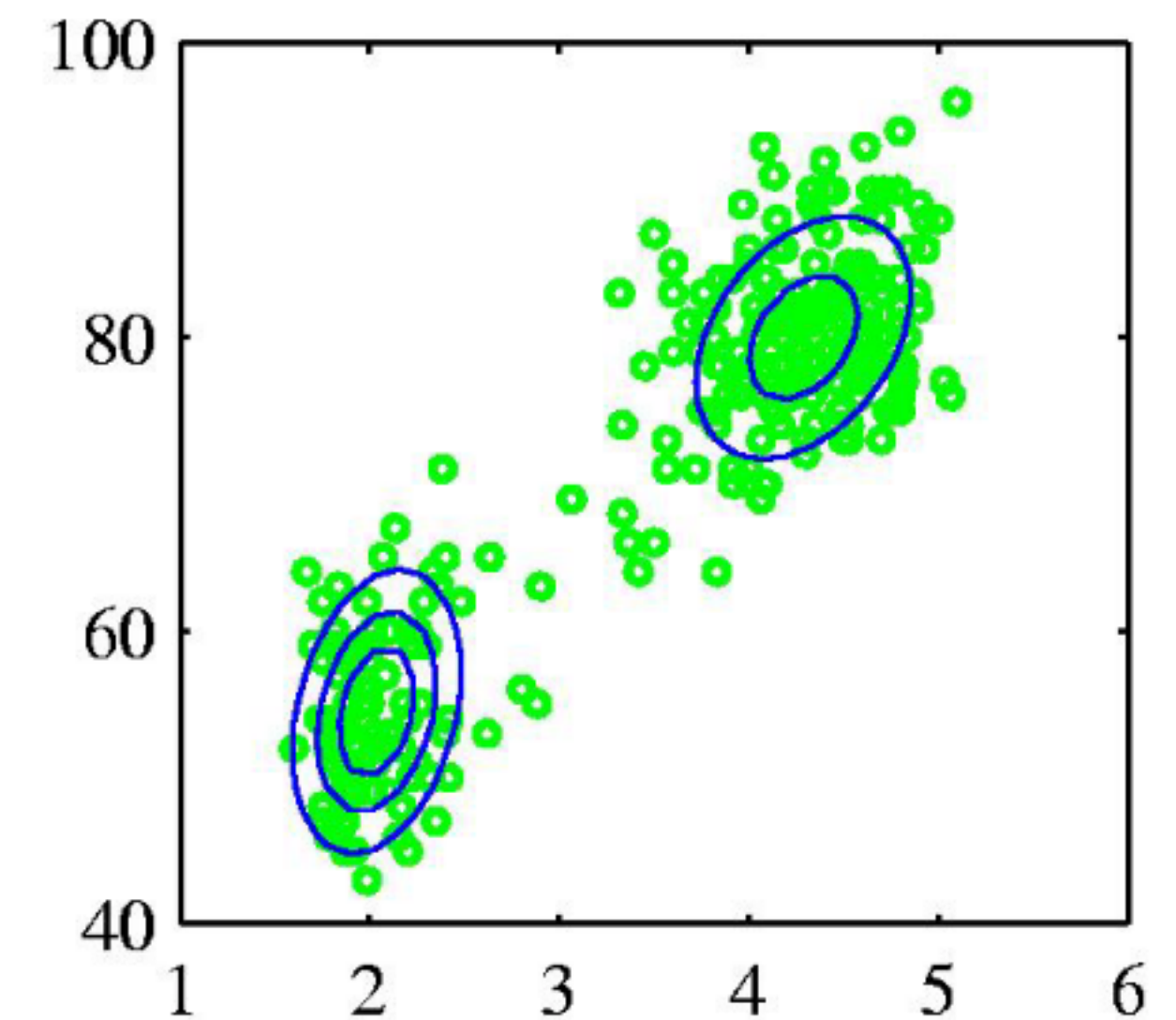


Mixture of Gaussians

Fitting a single Gaussian



A mixture of two Gaussians



Mixture of Gaussians

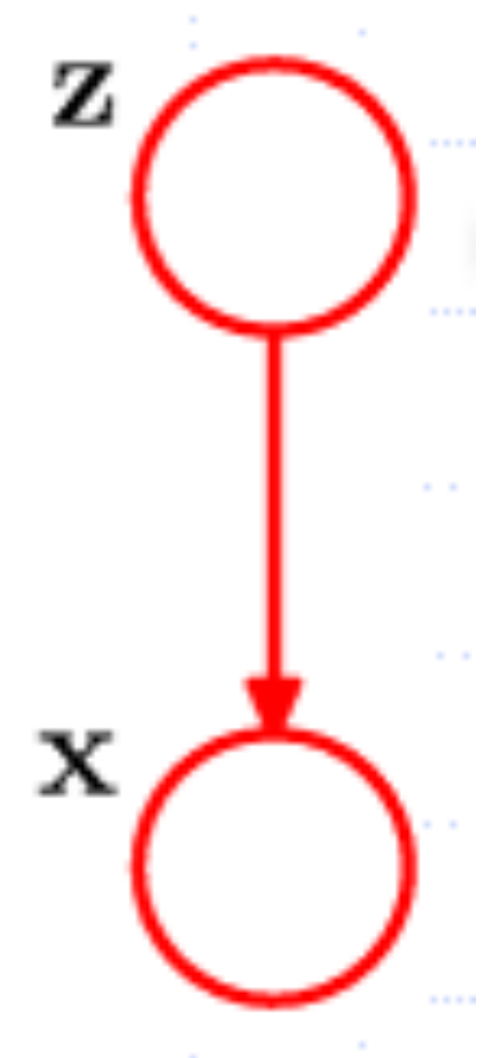
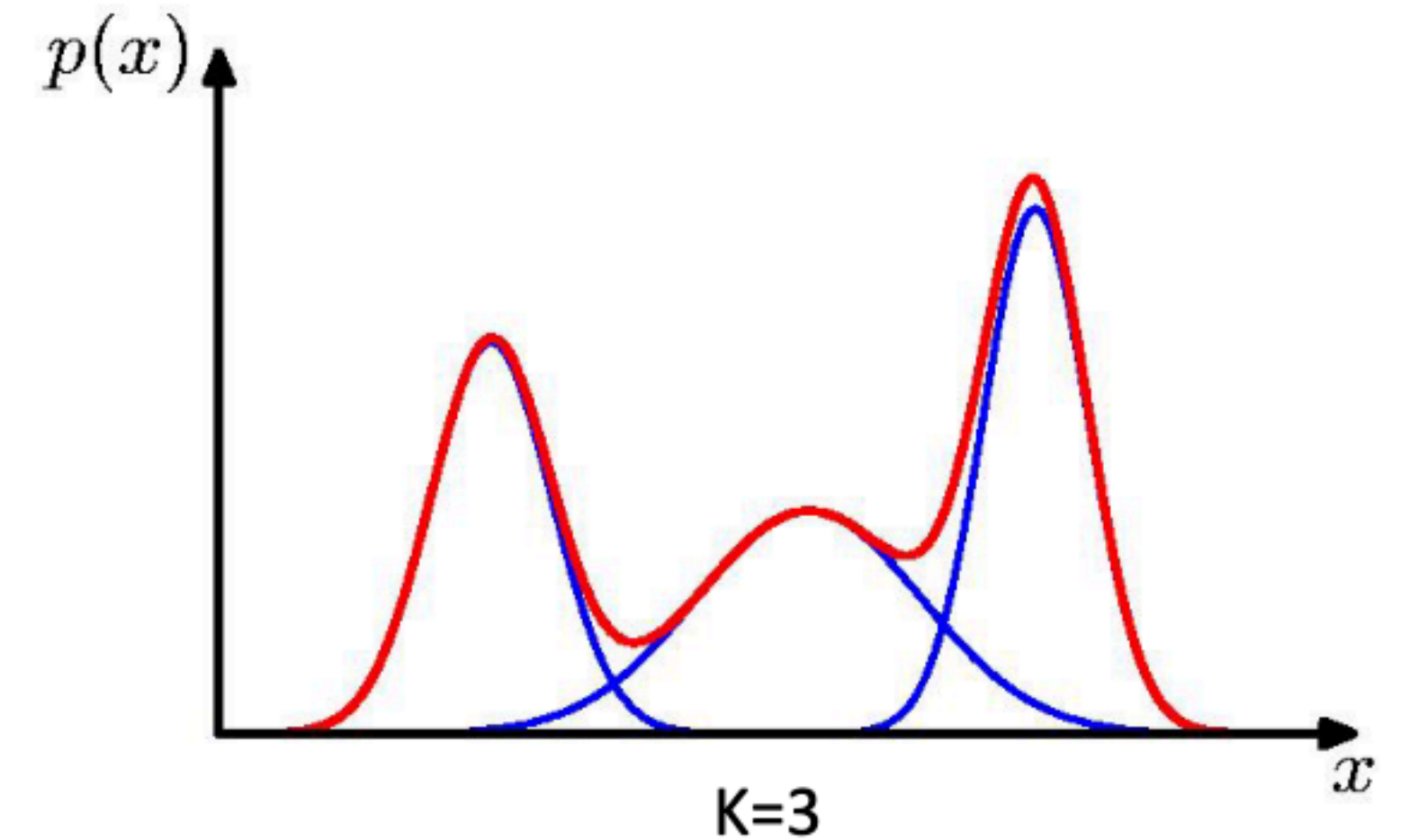
$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Define an indicator variable z_k , characterised by :

$$z_k \in \{0,1\}$$

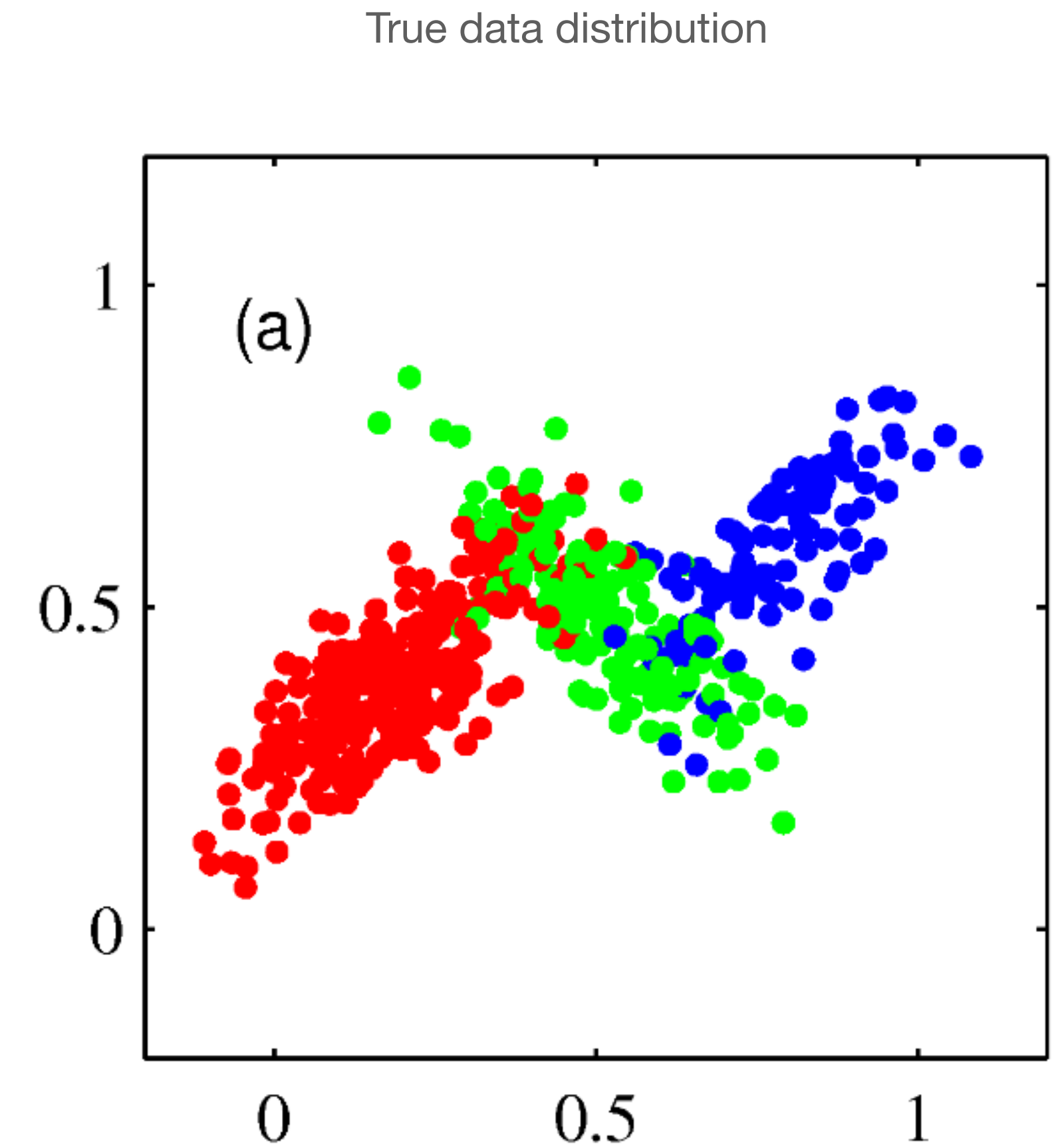
$p(z_k = 1) = \pi_k$ (the strength/mass: mixing coefficient)

$$\sum_{k=1}^K z_k = 1$$



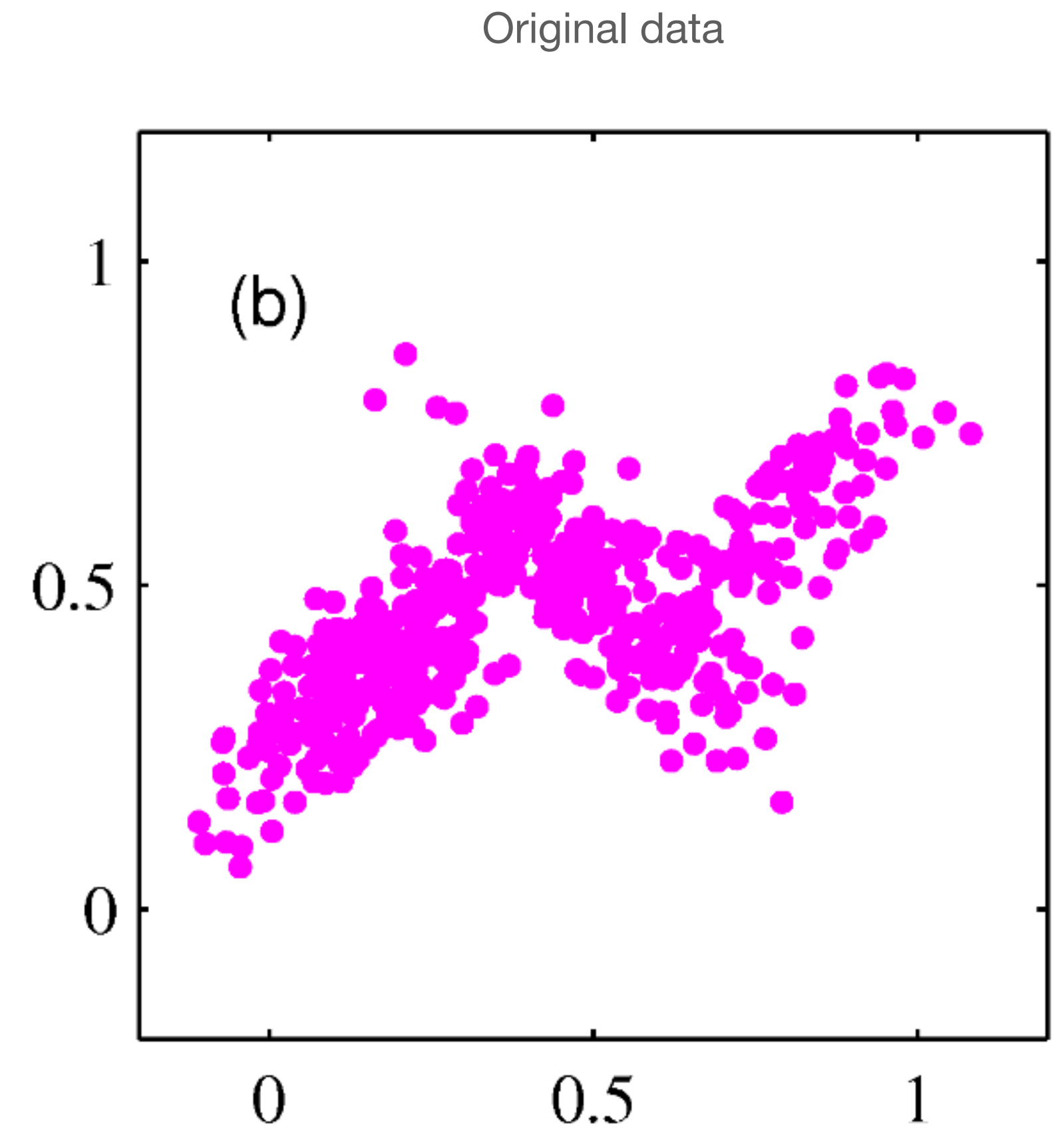
Soft assignments

Hard constraints will be problematic especially when we have overlaps



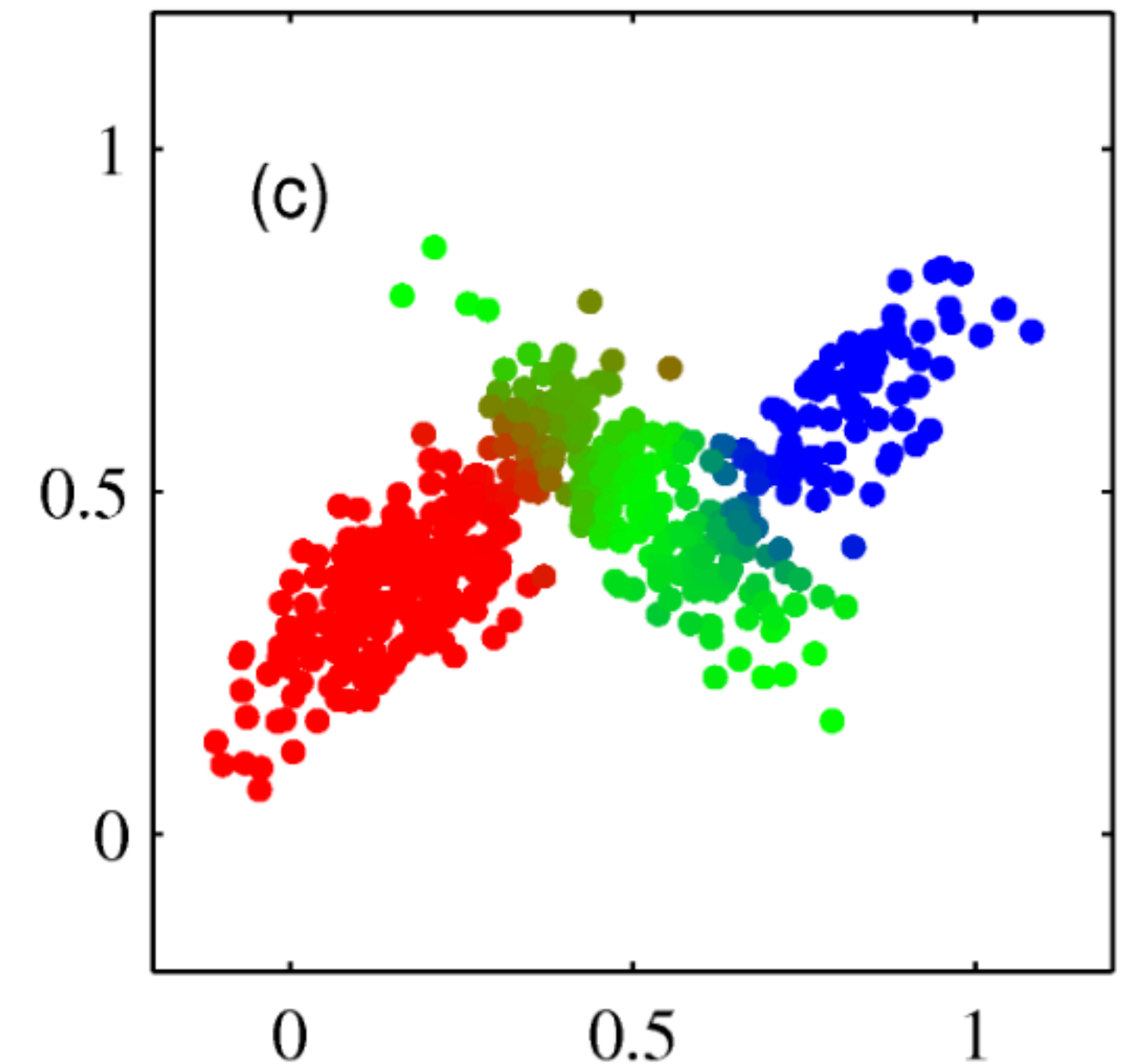
Soft assignments

We want to model this data as a mixture of multivariate Gaussians



Soft assignments

GMM - $P(Z = k | X)$ - posterior probability that a point was generated from k^{th} Gaussian



Revisiting MLE in supervised setting

For the univariate Gaussian setting:

$$\mu_{MLE} = \frac{1}{N} \sum_{i=1}^N x_i; \quad \sigma_{MLE}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{\mu})^2$$

For the multivariate case, this is simply:

$$\mu_{ML}^k = \frac{1}{n} \sum_{j=1}^n x_n; \quad \Sigma_{ML}^k = \frac{1}{n} \sum_{j=1}^n \left(\mathbf{x}_j - \mu_{ML}^k \right) \left(\mathbf{x}_j - \mu_{ML}^k \right)^T$$

we are just summing over the data for k^{th} Gaussian

Mixture of Gaussians (again)

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

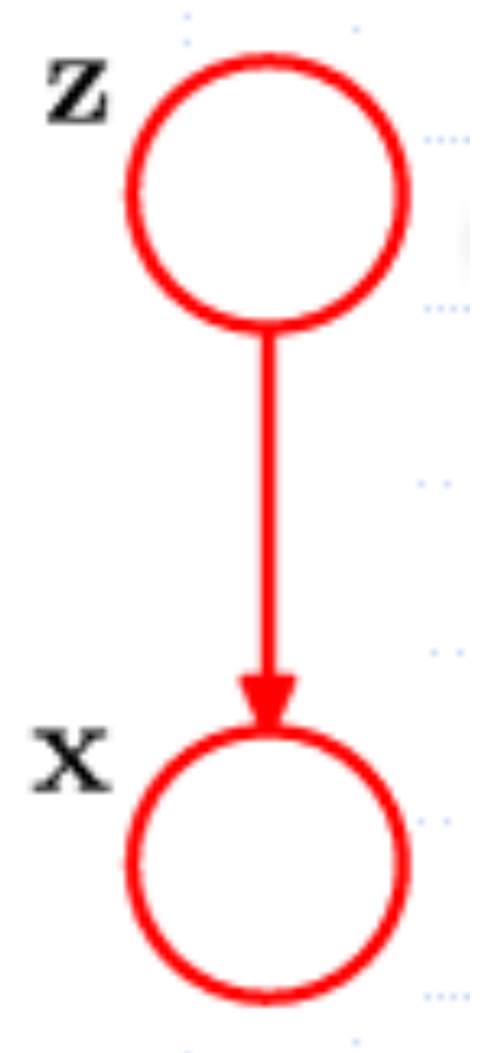
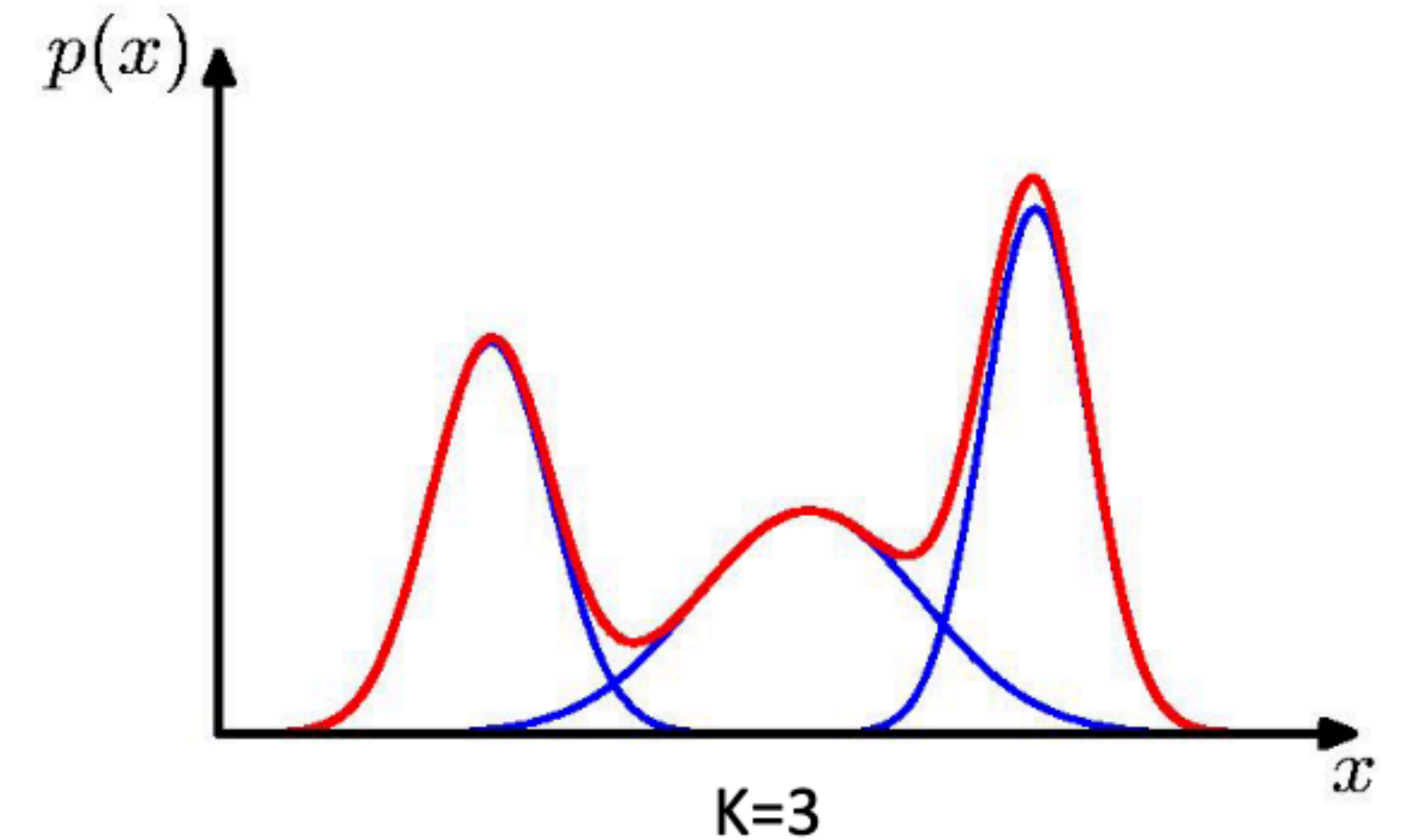
Define an indicator variable z_k , characterised by :

$$z_k \in \{0,1\}$$

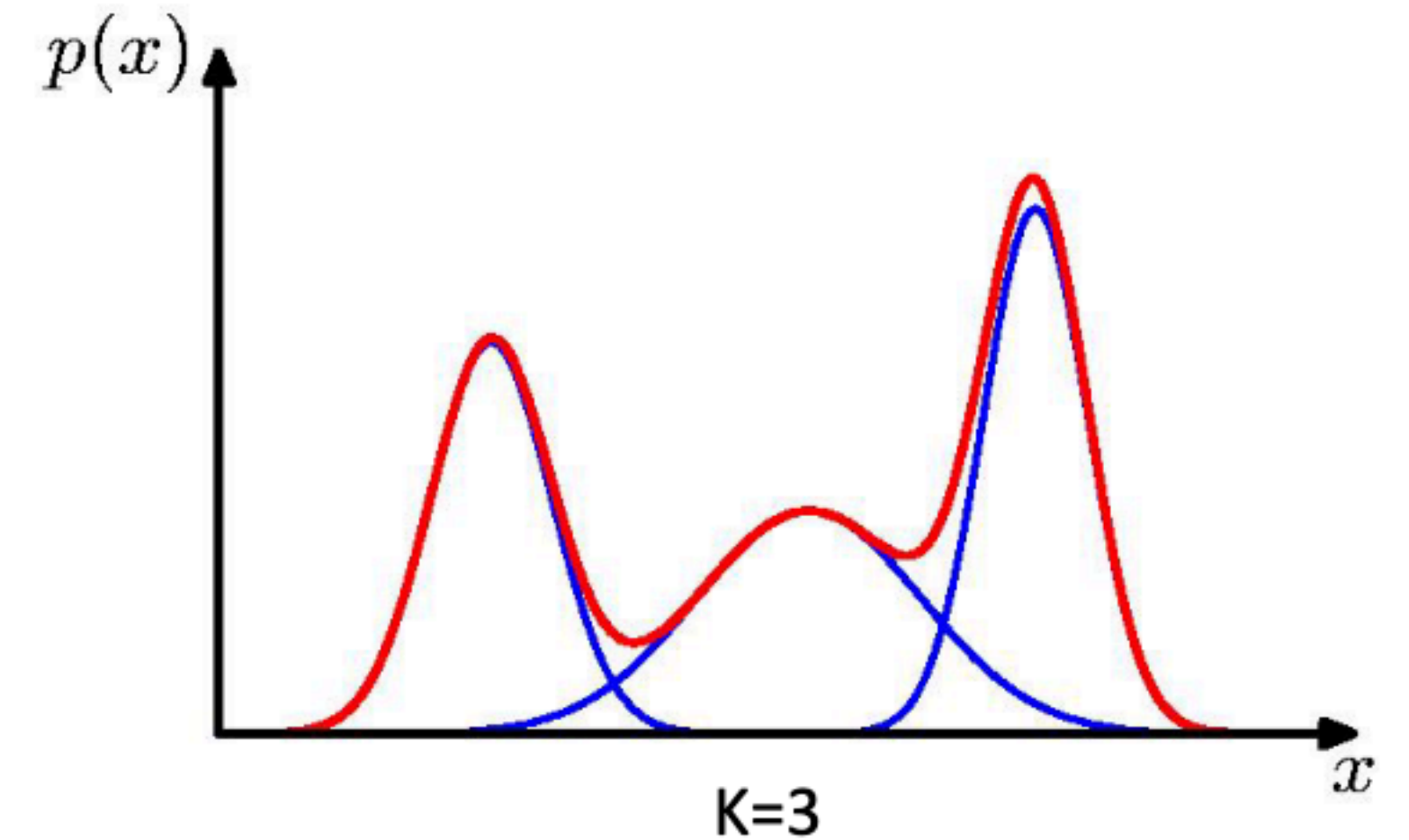
$p(z_k = 1) = \pi_k$ (the strength/mass: mixing coefficient)

$$\sum_{k=1}^K z_k = 1$$

Component



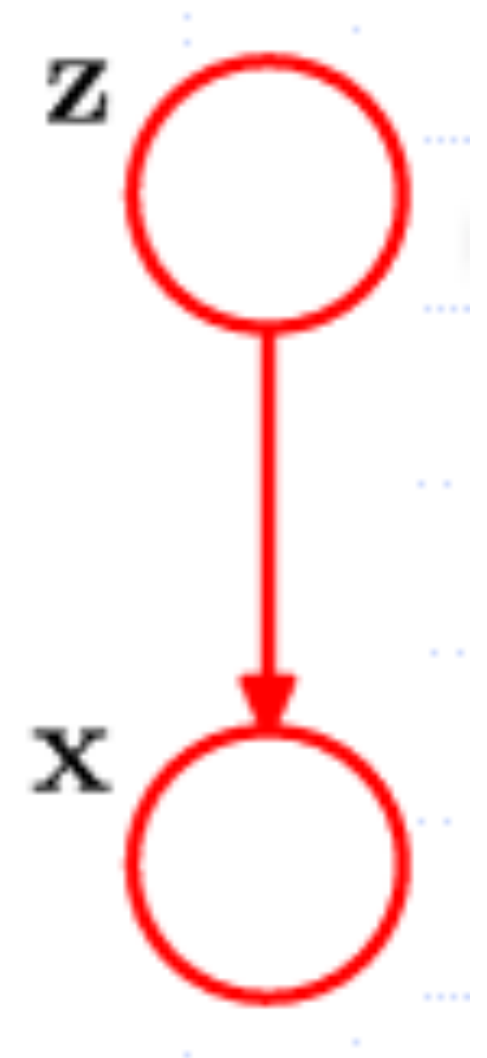
However, we are in an unsupervised setting



Let us define the joint distribution

$$p(\mathbf{x}) = \sum_z p(z)P(x|z) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

for convenience



Mixture of Gaussians

Now we are able to work with $p(\mathbf{x}, \mathbf{z})$ instead of $p(\mathbf{x})$, which will lead to significant simplifications

We also need to define another important quantity: the conditional probability of \mathbf{z} given \mathbf{x}

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1 \mid \mathbf{x}) &= \frac{\cancel{p(z_k = 1)} p(\mathbf{x} \mid z_k = 1)}{\sum_{j=1}^K p(z_j = 1) p(\mathbf{x} \mid z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\end{aligned}$$

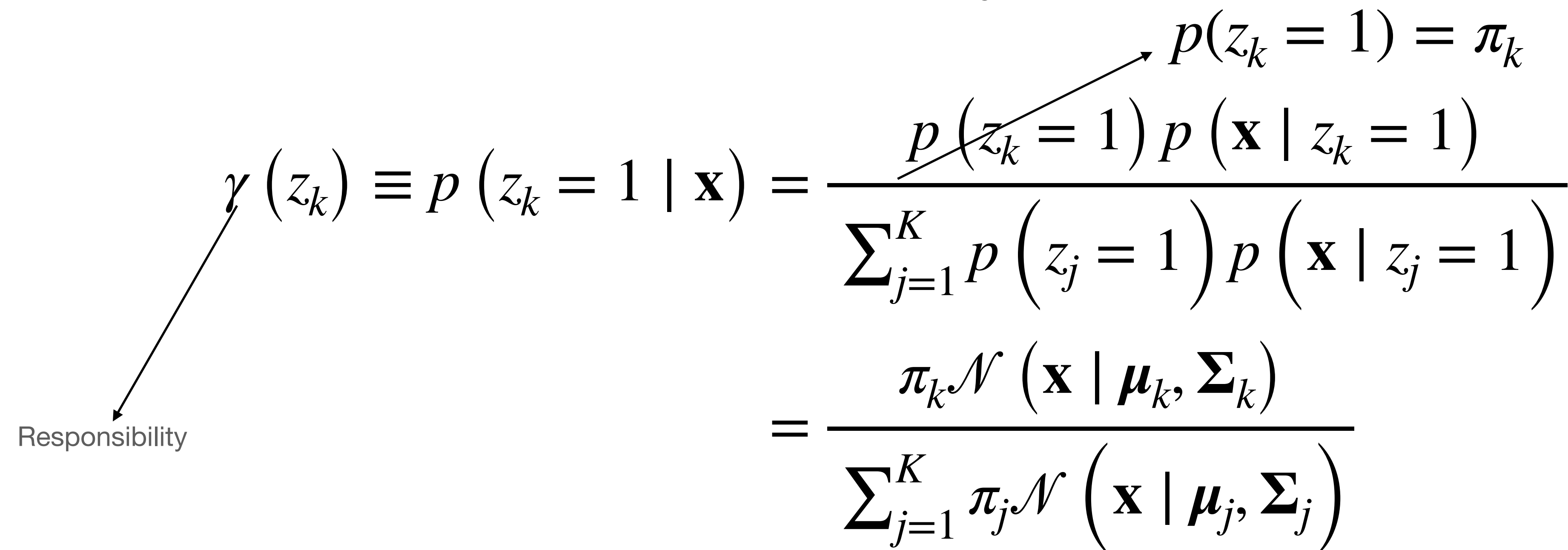
Mixture of Gaussians

Now we are able to work with $p(\mathbf{x}, \mathbf{z})$ instead of $p(\mathbf{x})$, which will lead to significant simplifications

We also need to define another important quantity: the conditional probability of \mathbf{z} given \mathbf{x}

$$\begin{aligned} \gamma(z_k) \equiv p(z_k = 1 \mid \mathbf{x}) &= \frac{p(z_k = 1) p(\mathbf{x} \mid z_k = 1)}{\sum_{j=1}^K p(z_j = 1) p(\mathbf{x} \mid z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \end{aligned}$$

Responsibility



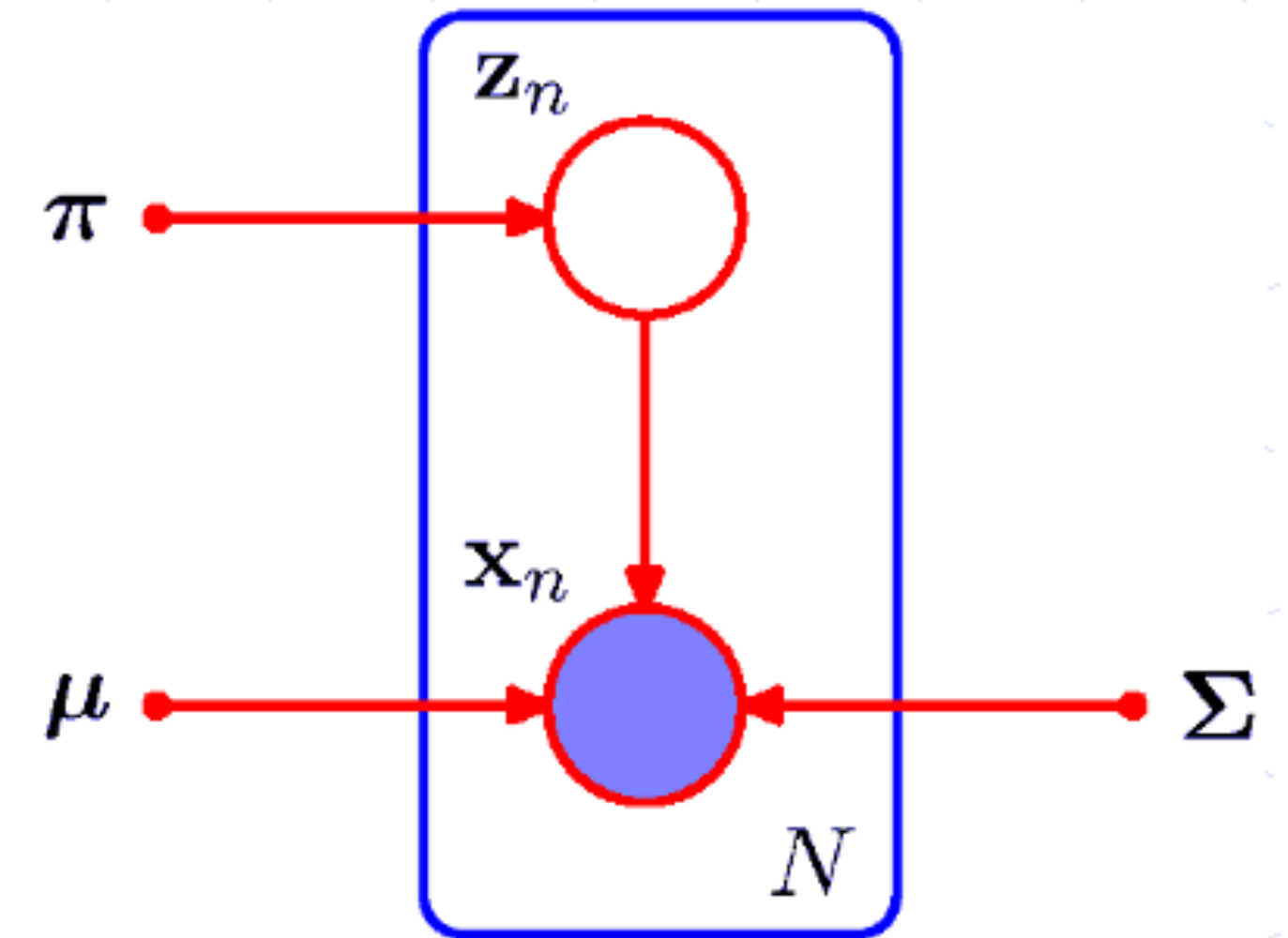
Mixture of Gaussians: plate notation

We have N datapoint consisting of $\{x_1, \dots, x_n\}$

The z_n is a latent variable

z_n in turn is controlled by the parameter π

Each of the x_n belong to a gaussian distribution parametrised by μ and Σ



MLE for GMMs

Suppose we have data $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ represented as matrix \mathbf{X}

Let us express the log-likelihood of the data using a GMM as:

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

- maximising this can be painful - singularities and with MLE, a K-component mixture model will have $K!$ solutions

Enter Expectation Maximisation

An iterative scheme to find a solution!

Alternate between two steps:

1. Compute an \mathbb{E} Expectation
2. Compute a maximisation

Similar to the previous alternate minimisation for K-Means!

Expectation Maximisation for GMM

$$\text{MLE for GMM: } \ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Setting the derivatives to 0 wrt $\boldsymbol{\mu}_k$:

$$0 = - \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

Expectation Maximisation for GMM

$$\text{MLE for GMM: } \ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

$$\text{Setting the derivatives to 0 wrt } \boldsymbol{\mu}_k: 0 = - \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

Rearranging:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n; \quad \text{where } N_k = \sum_{n=1}^N \gamma(z_{nk}) \text{ - no. of pts assigned to cluster } k$$

Expectation Maximisation for GMM

$$\text{MLE for GMM: } \ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Setting the derivatives to 0 wrt $\boldsymbol{\Sigma}_k$:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k) (\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Setting the derivatives to 0 wrt π_k :

$$\pi_k = \frac{N_k}{N}$$

Now we are armed with sub-problems!

EM for GMM the algorithm

Goal: given a potential initialisation of GMM, maximise the likelihood function wrt the all the means, covariances and the mixing coefficients

1. Initialise μ_k, Σ_k, π_k

2. Expectation step (E-step): $\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}$ (compute the expected class)

3. Maximisation step (M-step):

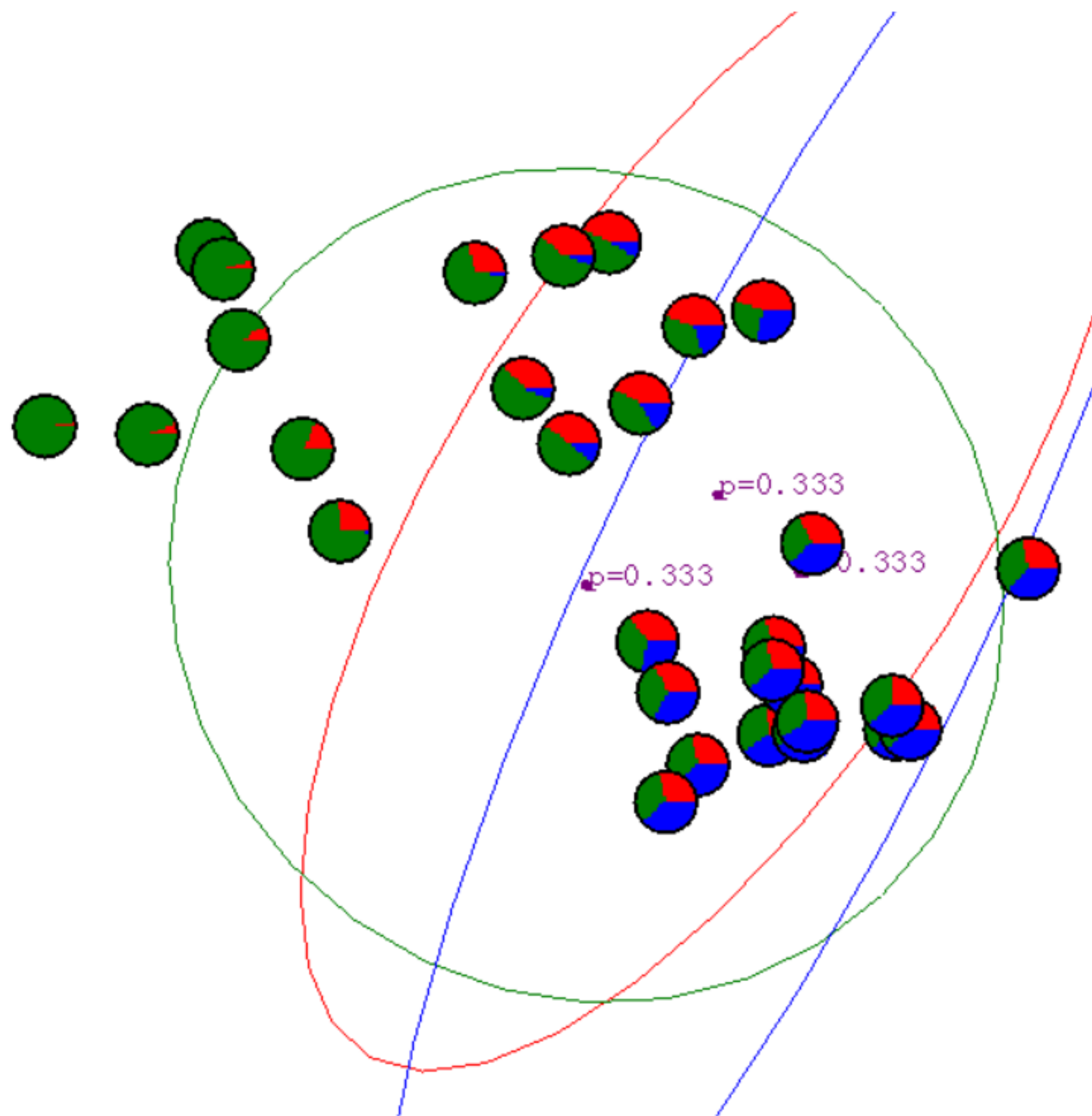
$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T$$

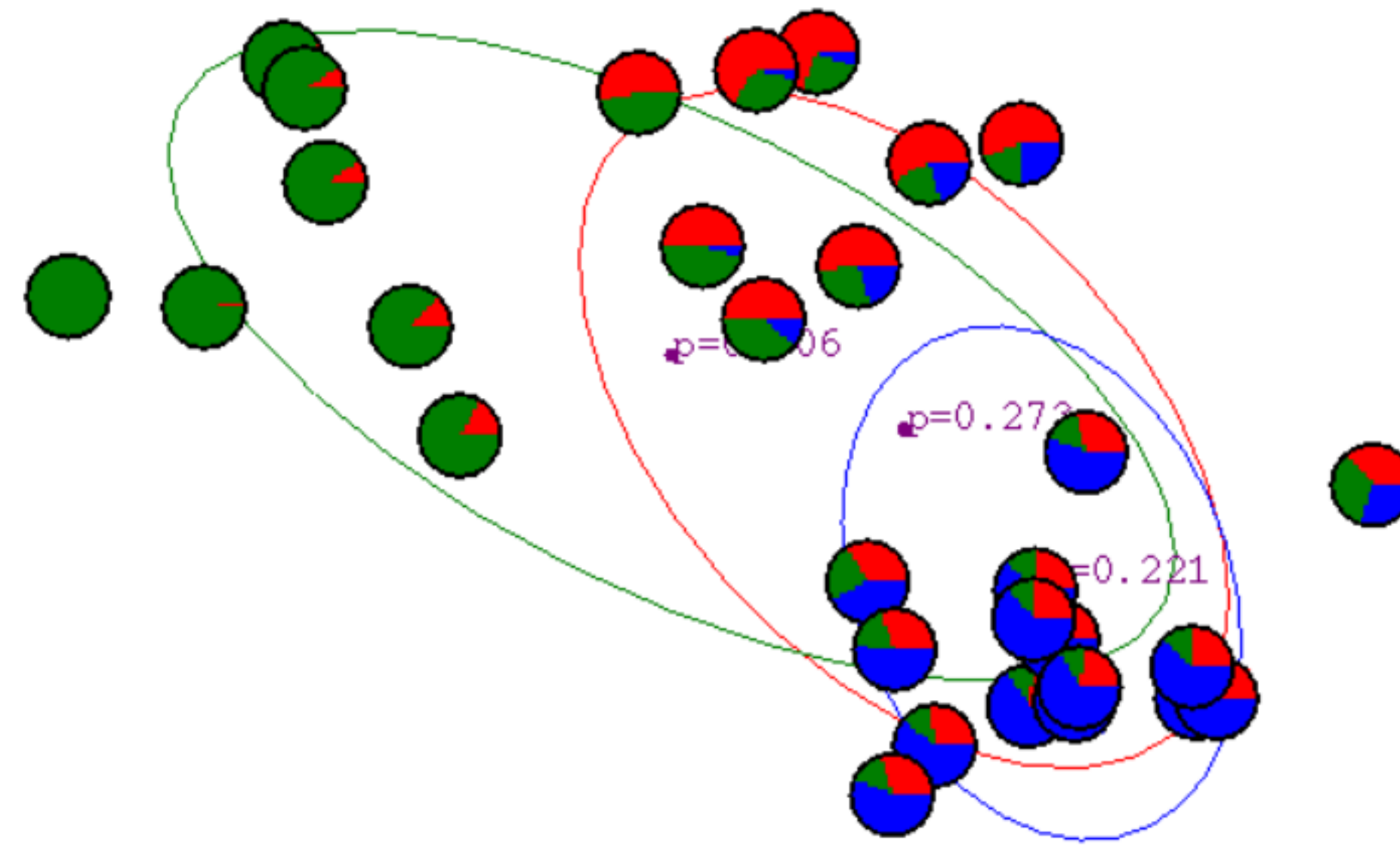
$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

Evaluate the log-likelihood and check for convergence. Else go to step 2!

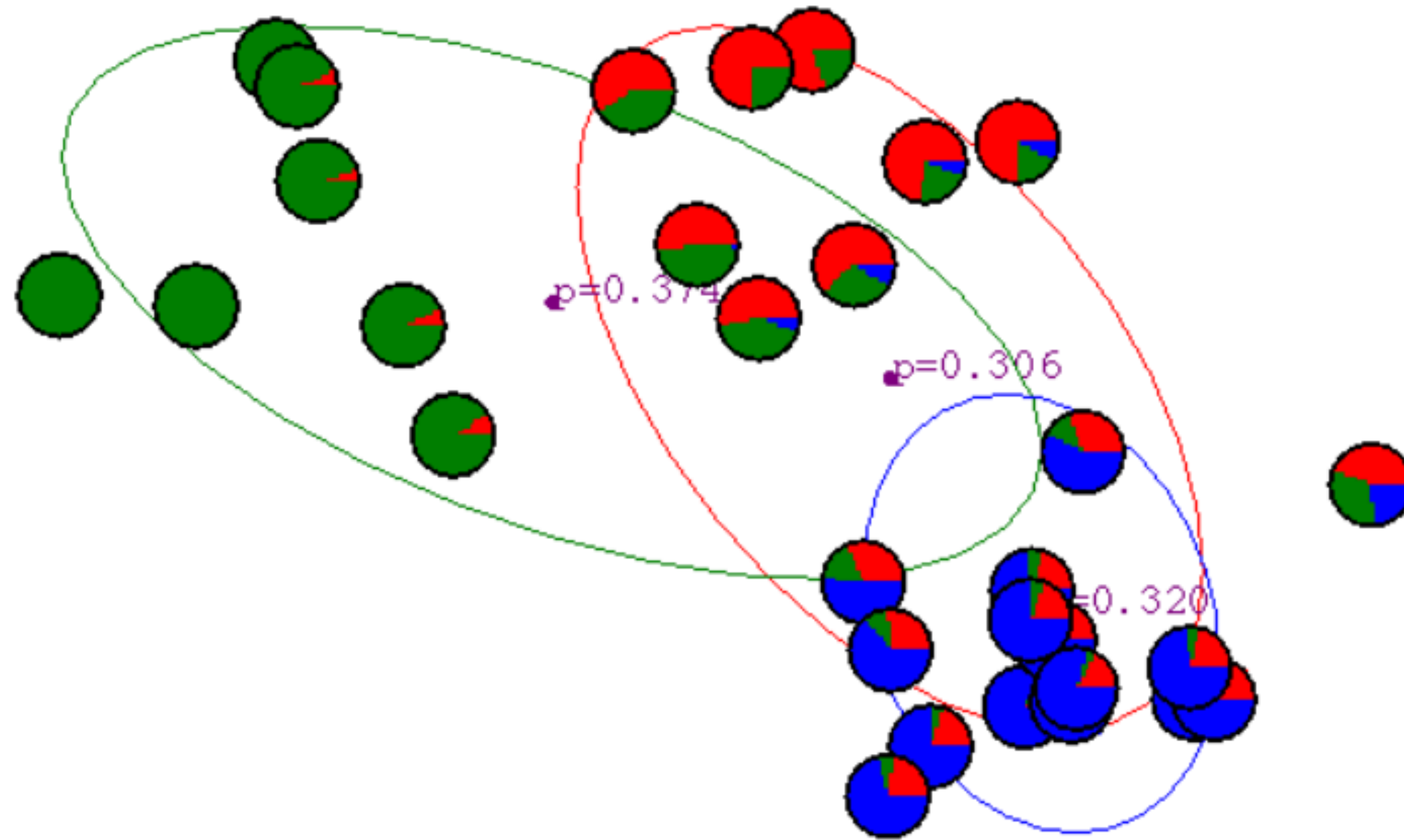
GMM



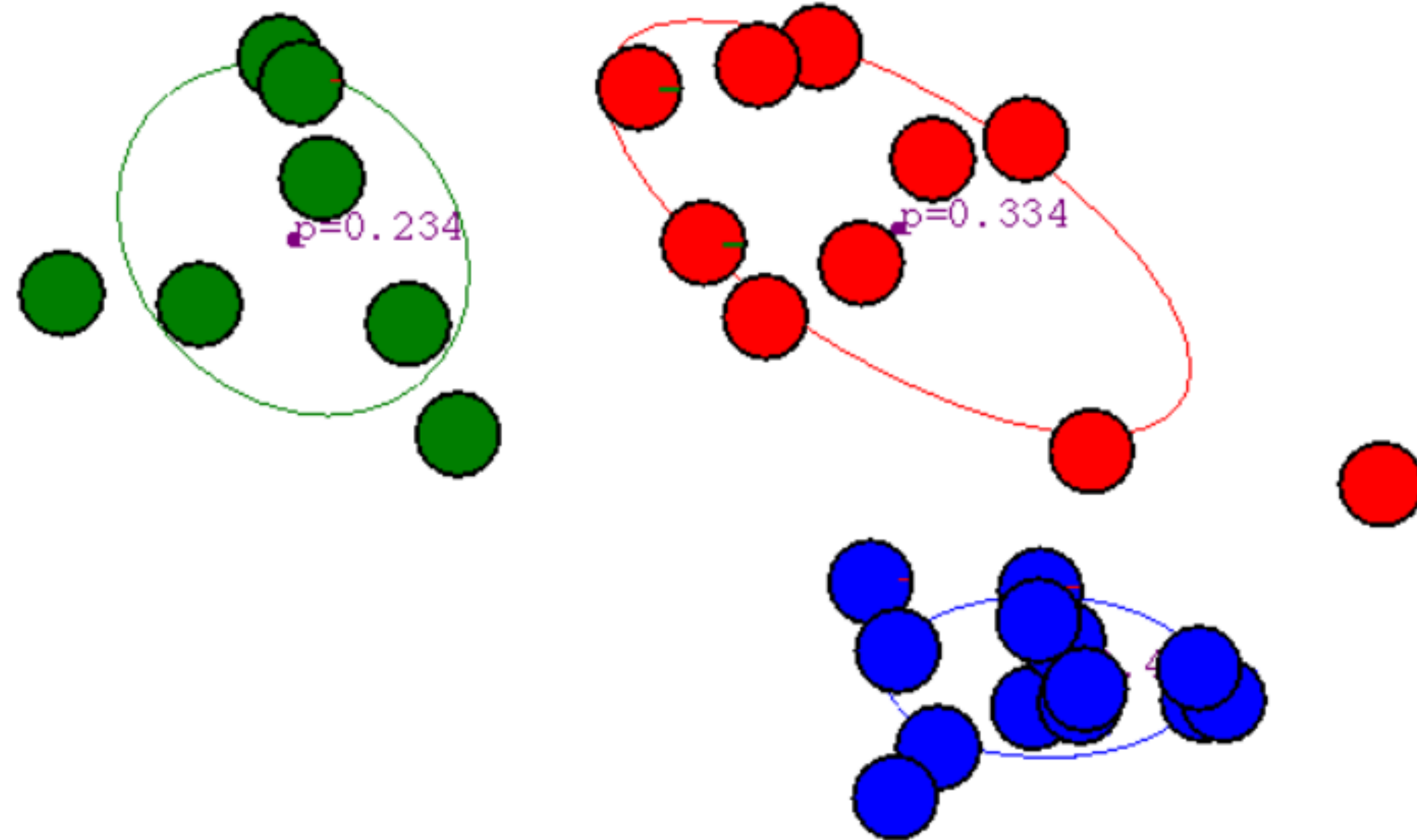
GMM iteration = 1



GMM iteration = 2



GMM iteration = 20



GMMs with hard assignments!

Clustering with K-Means -

$$\text{Assignment } d(\mathbf{x}_i, \mu_k) = \sqrt{\sum_{j=1}^d (x_{i,j} - \mu_{k,j})^2}; \text{ update means: } \mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

Clustering with GMM -

$$\text{Assignment } \gamma(z_{nk}) = \frac{\cancel{\pi_k} \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \cancel{\pi_j} \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}; \text{ update means: } \mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n$$

We can derive K-Means as a special case of GMM under hard-assignment without the need to estimate a covariance matrix. (Refer https://www.ece.iastate.edu/~namrata/EE527_Spring08/Dempster77.pdf for more details)

Expectation maximisation

EM takes more iterations to reach convergence compared to K-Means

One common approach is to initialise a GMM using K-Means

EM - converges but not necessarily to global maximum!

GMM classifier

GMM can be used as a classifier (we saw Naïve Bayes before)

General strategy:

1. Train a GMM for each class - predefined here (use EM)
2. At inference time: compute the likelihood of the test sample for each component. Select the class that obtains highest likelihood!

Gaussian discriminant analysis!

Next lecture

Hidden Markov models