(How do we find the subtypes of a disease? –) Clustering and designing new k-means algorithm variants via probabilistic modeling

Optional !Saturday Track!

CS5691 PRML Jul-Nov 2025

(used previously in “Algorithmic Thinking in Bioinformatics” Course

Manikandan Narayanan

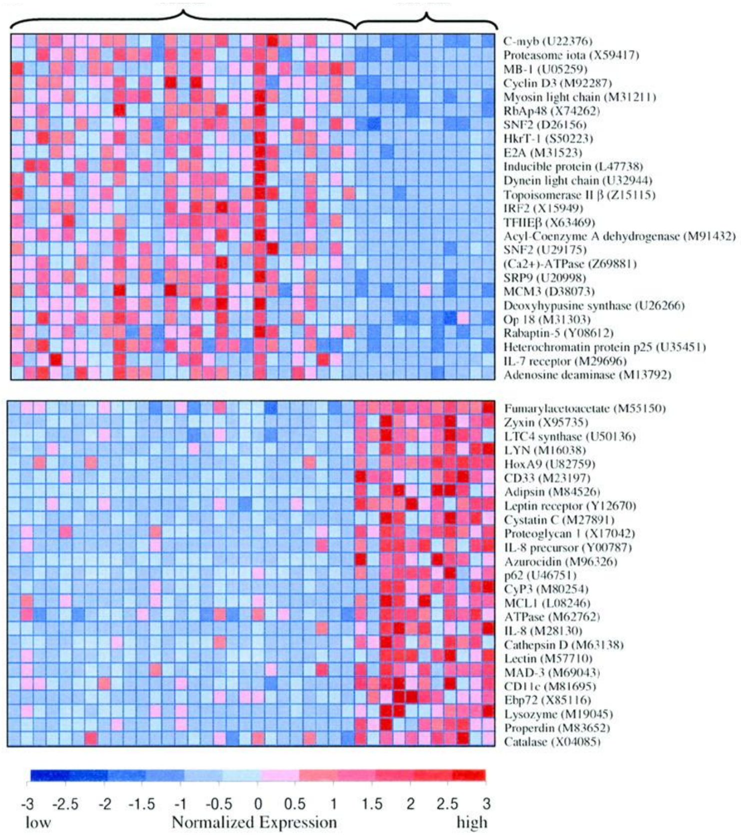
Associate Professor, Computer Science and Engineering, IIT Madras)

Context: Seen so far…

- Session on Motivation and Background for “Algorithmic Thinking in Bioinformatics” course
- Sessions on several biological questions & their algorithmic solutions

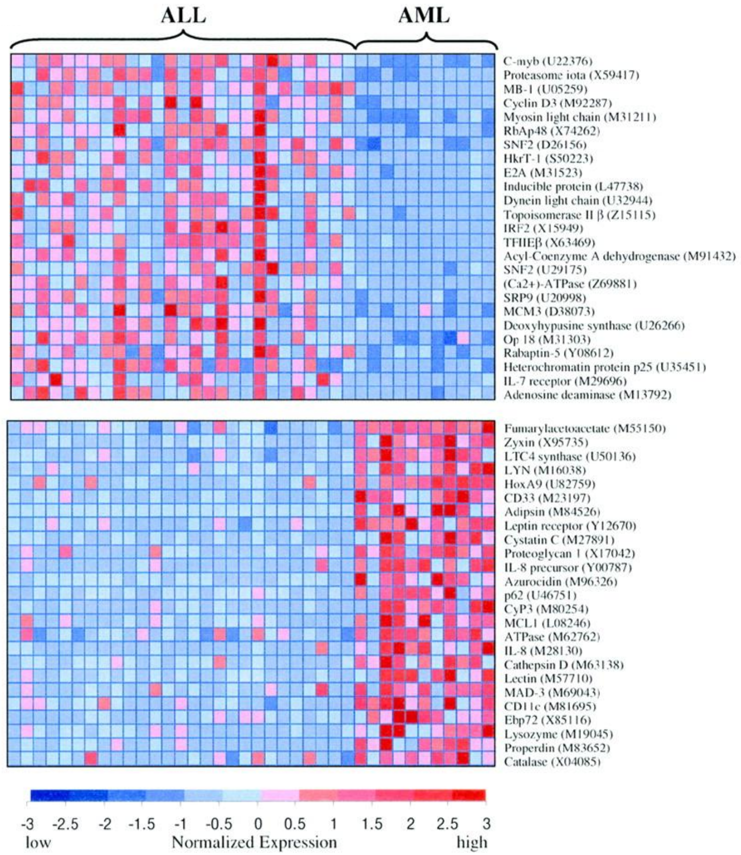
- This session: Think about a
  - clinical question (“how do we identify the different (sub)types of a disease?”),
  - associated computational problem (clustering), and
  - its algorithmic solution(s) (k-means algorithm and designing this algo.’s new variants using probabilistic modeling).

Cancer example: Can we group cancer samples into different (sub)types?



Golub et al. Science 1999

Cancer example: from grouping to prediction!



Golub et al. Science 1999

Clustering: a popular data science task

- Human brains are good at grouping objects based on their similarity. How do we automate it?
  - Group N objects into K groups
  - Paradigms:
    - K-means clustering – centroid-based;
    - Hierarchical clustering – nested/stratified with top-down or bottom-up approaches;
    - Spectral clustering – graph connectivity-based;
    - etc.

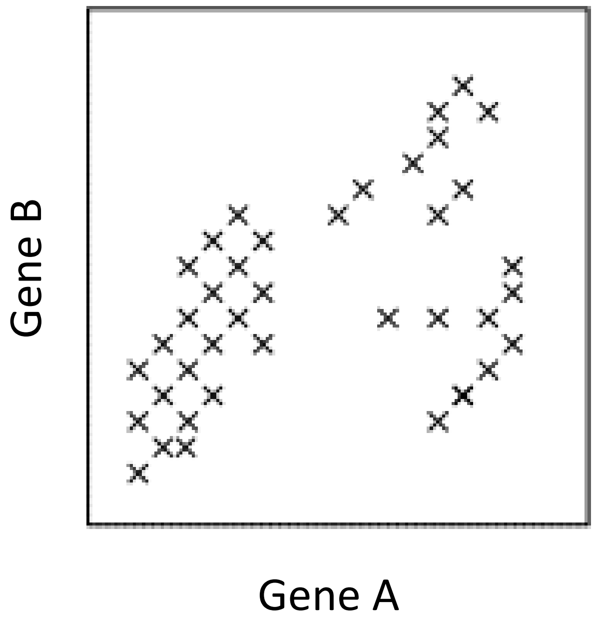
- Why is a good clustering popular? It
  - allows exploratory data analysis (unsupervised machine learning)
  - has predictive power
  - allows lossy compression
  - can reveal interesting outliers

Clustering: a popular data science task viewed thru' a probabilistic modeling lens

- Our approach in this lecture
  - Present k-means approach to clustering, starting with a non-probabilistic approach (hard k-means) and then transitioning to a probabilistic approach (soft k-means).
  - *Focus is on viewing (k-means) clustering as an inference+parameter-learning task based on a probabilistic mixture model* – ***this mixture dens. estn. approach will help us design new k-means algo. variants!***

- Sources for this lecture:
  - Main source:
    - David J. C. MacKay. Information Theory, Inference and Learning Algorithms (Chapters 20-22). 2003. – **cited as [DJM]** for content/figures taken as is from this book.
  - Other sources:
    - Christopher M. Bishop. Pattern Recognition and Machine Learning. 2006. – **cited as [CMB]** for figures taken as is from this book.
    - Stanford CS228 Probabilistic Graphical Models Course Lecture Notes. https://ermongroup.github.io/cs228-notes/ - cited as **[ECL]**

N=40 data points over two dimensions (genes)



[DJM]

K-means Clustering Outline

1) hard version: algorithm and analysis

2) soft version vanilla (aided by intuitive formula)

3) soft version vanilla (aided by a vanilla probabilistic model)

4) soft version enhancements (aided by an enhanced probabilistic model)

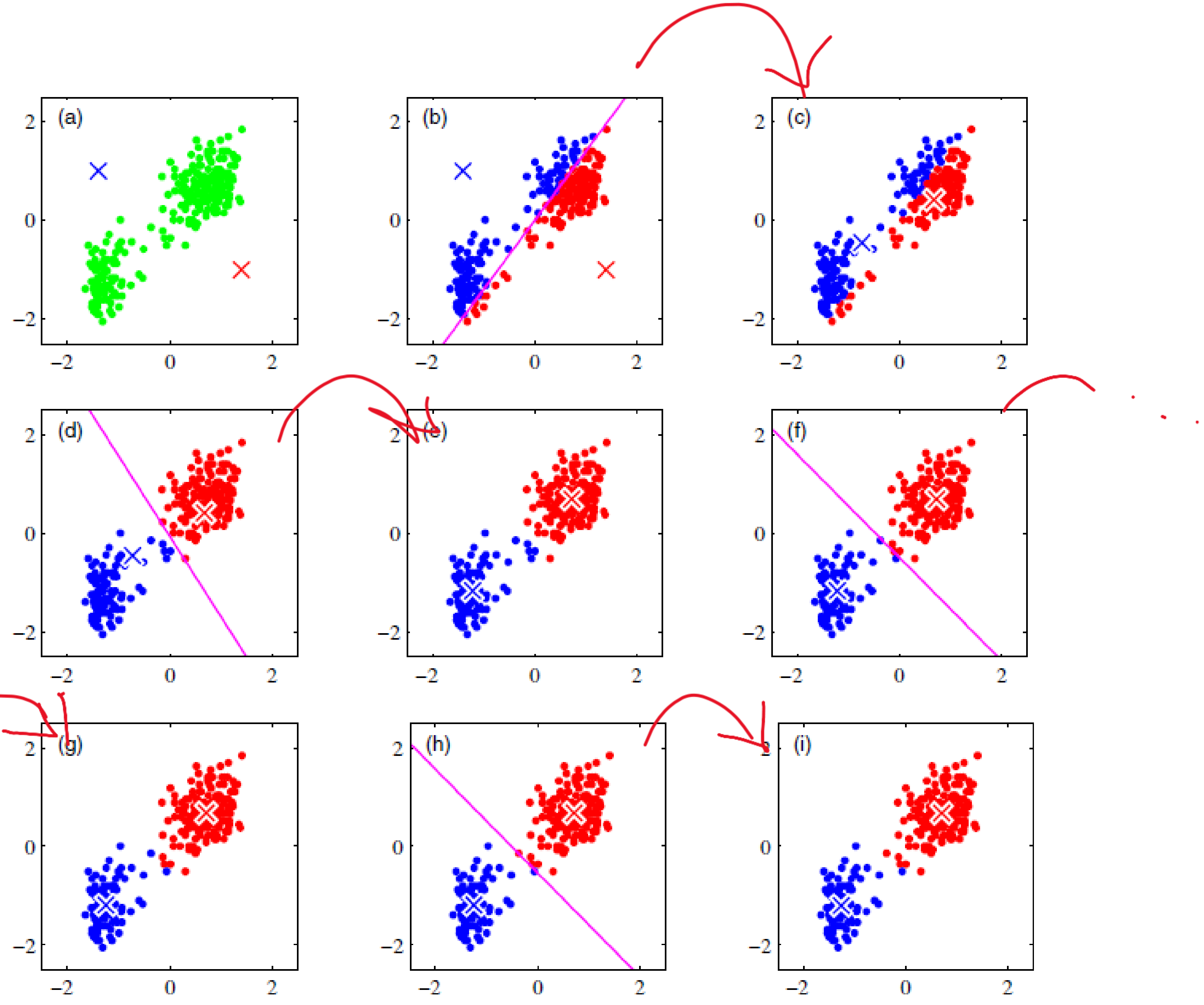
K-means clustering (in one slide!)

Problem: Find K cluster centers that minimize the sum of squared distance of each data point to the nearest cluster center.

Algorithm (Lloyd's Heuristic): Starting with K centers (means) initialized in some way, iterate until convergence:

a) ***[Centers -> Clusters] Assign*** each data point to the nearest center.
b) ***[Clusters -> Centers] Update*** centers to sample means of data points they are responsible for.

[DJM]



[CMB]

K-means clustering problem and notation

Input: N objects: $\{\, x^{(n)} \,\} = \; x^{(1)}, x^{(2)}, \ldots \,, \; x^{(N)}$

Output: K centers/means: $\{\, m^{(k)} \,\} = \; m^{(1)}, m^{(2)}, \ldots, m^{(K)}$

$m^{(k)}, x^{(n)} \in \mathbb{R}^{I}$

(Squared) Distance measure: $d(x, m) = \Sigma_{i=1}^{I} (x_i - \; m_i)^2$, where

I is dimensionality of data (2 in our examples).

Problem: Find K means that minimize $\Sigma_{n=1}^{N} \; d(x^{(n)}, m^{(k^*(n))})$, where

$k^*(n) = \text{argmin}_k \; d(x^{(n)}, m^{(k)})$

[DJM]

K-means clust. pseudocode: Lloyd's heuristic

Initialization: Set K means $\{m^{(k)}\}$ to random values.

Assignment: $r_k^{(n)}$ = 1 if $m^{(k)}$ is the closest mean to datapoint $x^{(n)}$

0 otherwise

Update:

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}.$$

[DJM]

K-means clust. pseudocode: Lloyd's heuristic

Initialization: Set K means $\{m^{(k)}\}$ to random values.

Assignment: $r_k^{(n)}$ = 1 if $m^{(k)}$ is the closest mean to datapoint $x^{(n)}$

0 otherwise

Update:

$$\mathrm{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

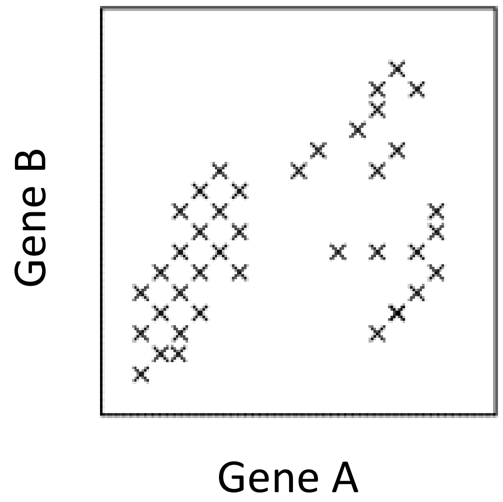
where $R^{(k)}$ is the total responsibility of mean k,

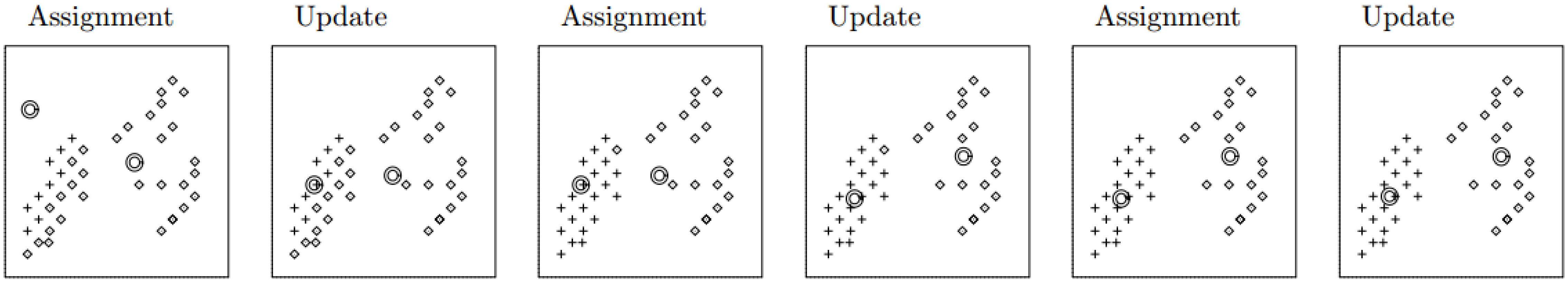
$$R^{(k)} = \sum_n r_k^{(n)}.$$

$\sum_k R^{(k)} = N$

[DJM]

Example run (K=2)



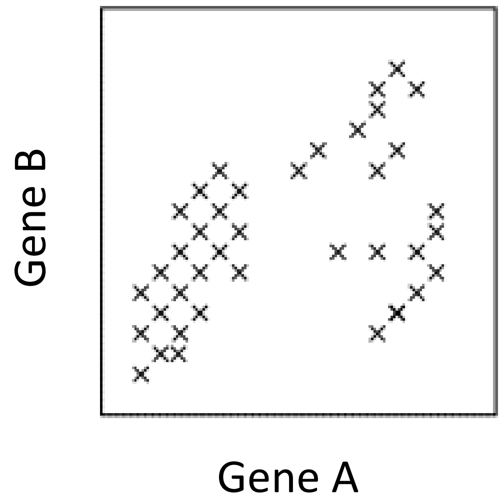


[DJM]

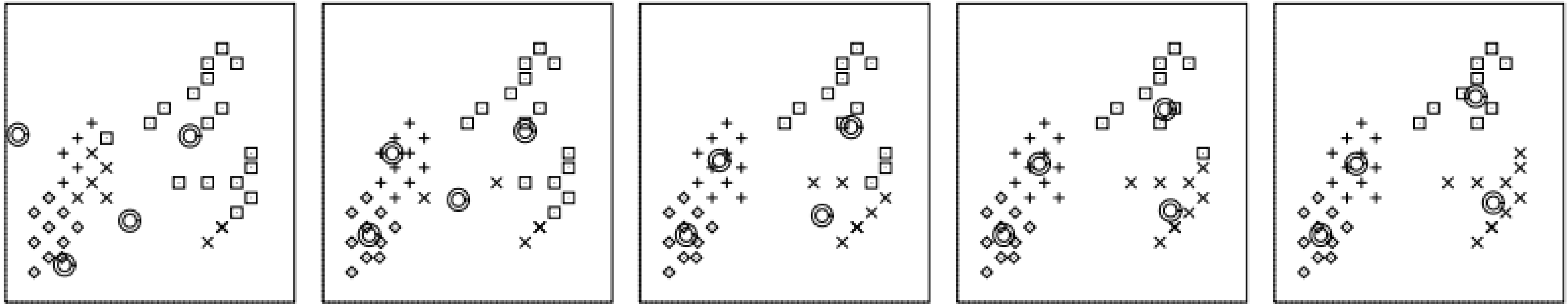
Let’s try K=4 clusters

with two different starting points!

Example run1 (K=4)

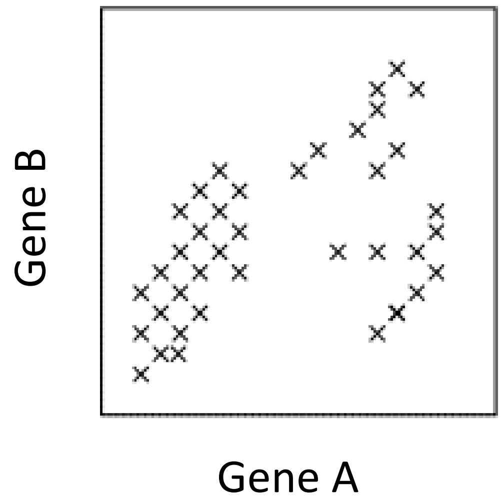


Run 1

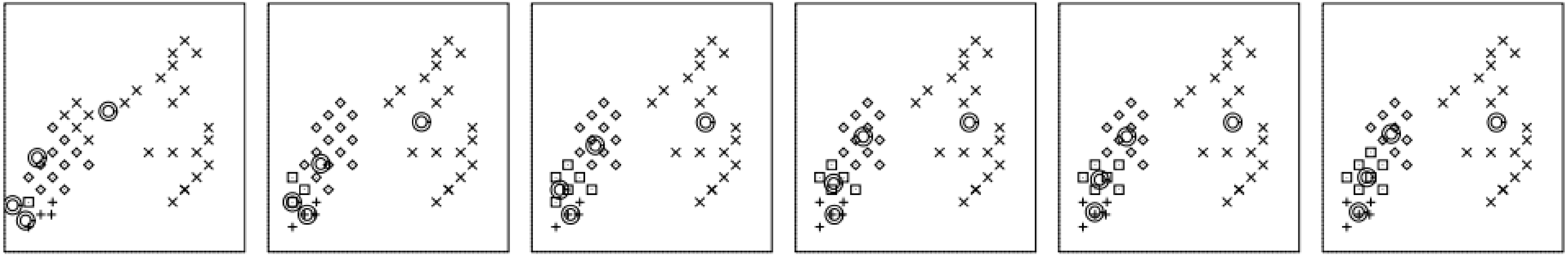


[DJM]

Example run2 (K=4)



Run 2



[DJM]

Analysis of algorithm – Questions

Problem: Find K cluster centers that minimize the sum of squared distance of each data point to the nearest cluster center.

1) Are the means (centroids) the best (and **only**) cluster centers for a given partition?

Is $\mu^{(k)} = \arg\min_{p} \sum_{n: r_k^{(n)}=1} d(x^{(n)}, p)$?

2) Does the algorithm converge always for any dataset?

3) Does the algorithm actually (globally) minimize the sum of squared distances?

K-means clust. pseudocode: Lloyd's heuristic

Initialization: Set K means $\{m^{(k)}\}$ to random values.

Assignment: $r_k^{(n)}$ = 1 if $m^{(k)}$ is the closest mean to datapoint $x^{(n)}$

0 otherwise

$\boxed{\text{objfn} = \sum_n \sum_k r_k^{(n)} \cdot d(x^{(n)}, m^{(k)})}$

Update:

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

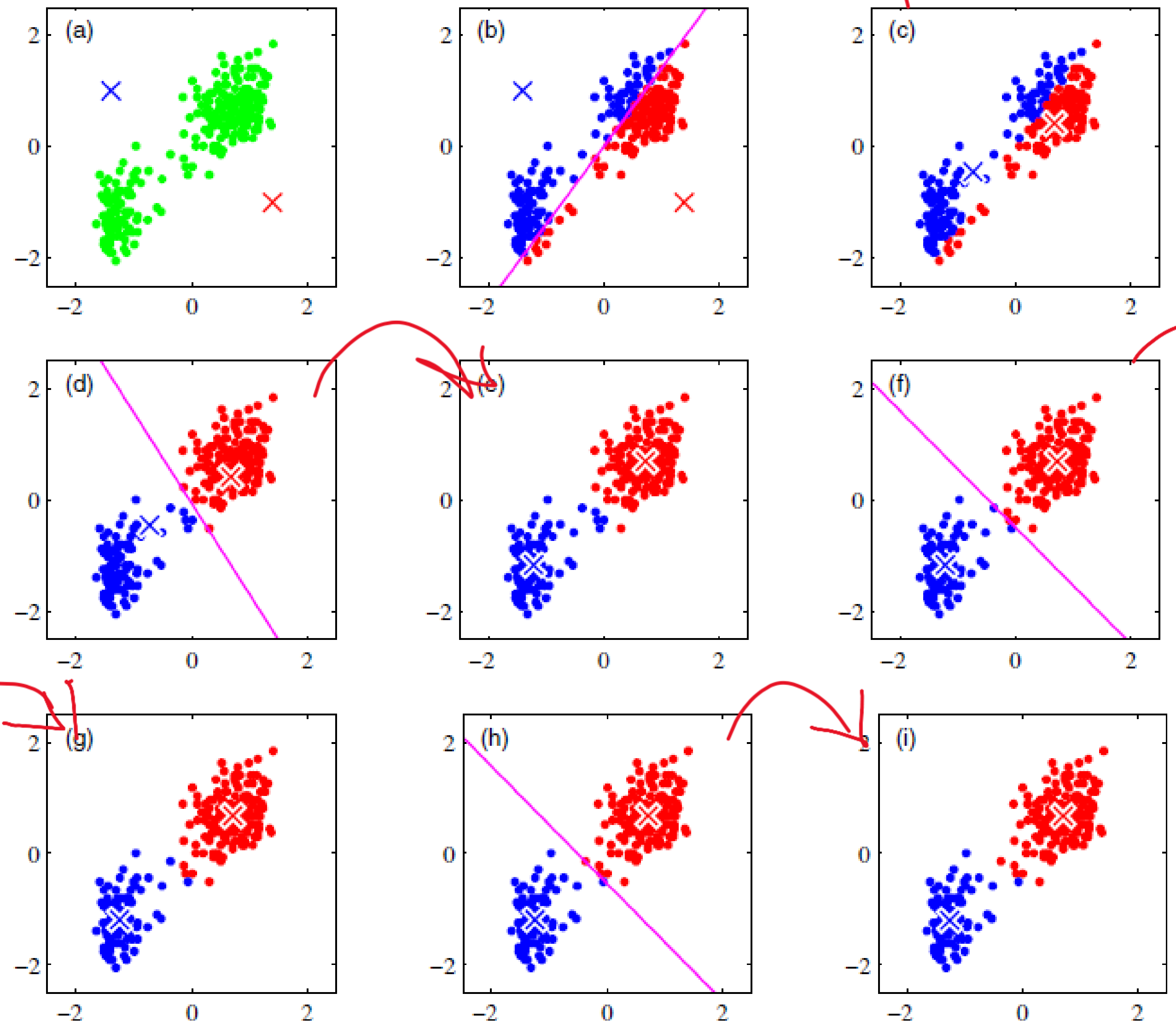
(centroid or center of gravity of cluster k) is the only minimizer

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}.$$

[DJM]

obj fn decreases



[CMB]

Analysis of algorithm – Answers

Problem: Find K cluster centers that minimize the sum of squared distance of each data point to the nearest cluster center.

1) Are the means (centroids) the best (and only) cluster centers for a given partition?

Yes, cluster k centroid is the only minimizer of cluster k cost, given a partition. (Prove it!)

2) Does the algorithm converge always for any dataset?

Yes, because:

a) Every iteration reduce obj. fn. [by (1) above]

b) Obj. fn is lower-bounded by 0.

c) There are only finite # of ways to partition N datapts into K clusters.

3) Does the algorithm actually (globally) minimize the sum of squared distances?

No, only reaches local minima as seen in K=4 cluster runs above. (NP-hard to find global minima)

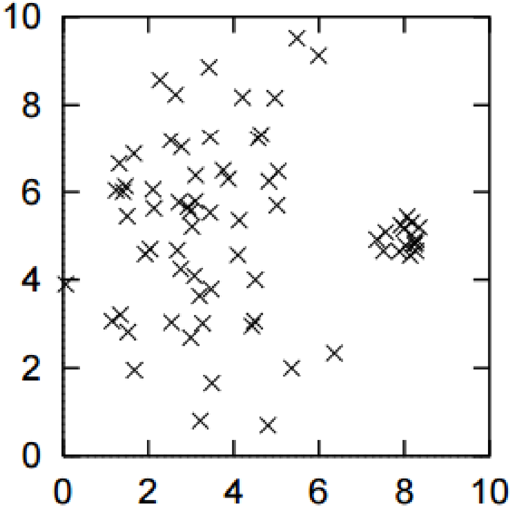
Food for thought – Exercise questions

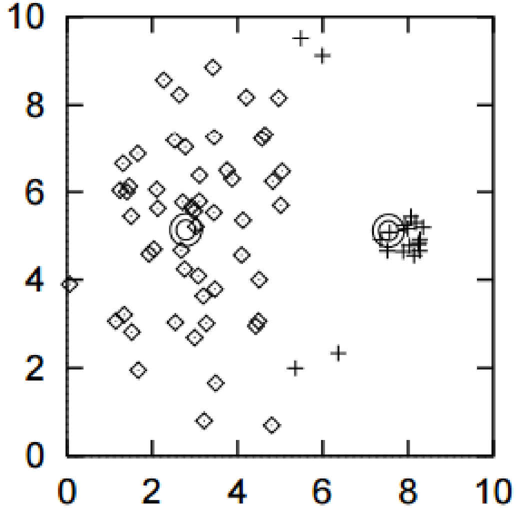
- Prove that (hard) k-means yields clusters that are each convex sets!

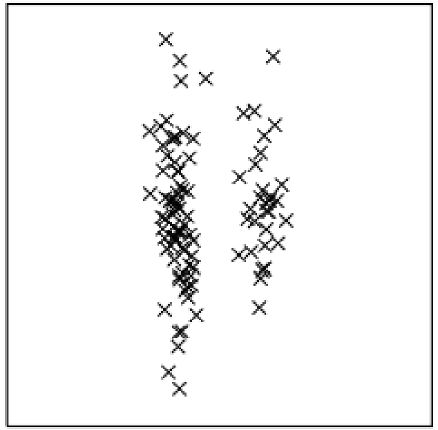
- Prove that k-means is efficiently (polynomial-time) solvable when all datapoints lie on a real line (i.e., finding k centers that minimize k-means cost function when all datapoints are 1D).

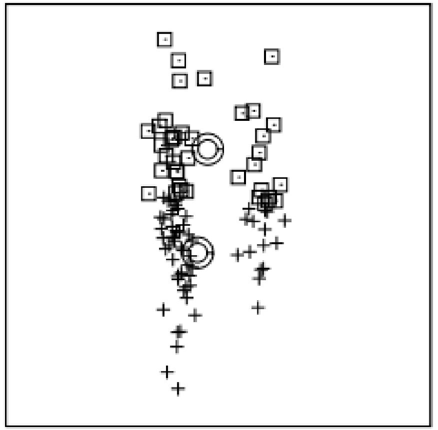
Blank space

K-means pitfalls









[DJM]

K-means pitfalls: solutions?

- How about points in the border between two means? THINK FUZZY.

- What is a better alternative for distance $d(.,.)$? THINK ELLIPSOIDAL.
  - Instead of giving equal weight to all dimensions in a spherical fashion as in squared Euclidean distance, why not choose a distance that adapts to the width and breadth of each cluster?

[DJM]

K-means Clustering Outline

1) hard version: algorithm and analysis

2) soft version vanilla (aided by intuitive formula)

3) soft version vanilla (aided by a vanilla probabilistic model)

4) soft version enhancements (aided by an enhanced probabilistic model)

Q: Can we make $r_k^{(n)}$ fuzzy/soft using an intuitive formula?

Q: Can we make $r_k^{(n)}$ fuzzy/soft using an intuitive formula?

- Intuitive formula (non-probabilistic or heuristic methods):

$r_k^{(n)}$ hard $\Rightarrow \dfrac{d_k}{d_1+d_2+d_3}$ (or) $\dfrac{e^{-\beta d_k}}{e^{-\beta d_1}+e^{-\beta d_2}+e^{-\beta d_3}}$

soft

Vanilla Soft K-means Clustering - pseudocode

Initialization: Set K means $\{m^{(k)}\}$ to random values.

Assignment:

(E-step)

$$r_k^{(n)} = \frac{\exp\left(-\beta\, d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})\right)}{\sum_{k'} \exp\left(-\beta\, d(\mathbf{m}^{(k')}, \mathbf{x}^{(n)})\right)}.$$

Update:

(M-step)

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}.$$

Blank space for illustrations: What is the role of the *stiffness/hardness* parameter β?

Does this formula have any connection to a probabilistic model, and can it be exploited?

- Intuitive formula:

$\gamma_k^{(n)} \Rightarrow$

hard

$$\frac{e^{-\beta d_k}}{e^{-\beta d_1}+e^{-\beta d_2}+e^{-\beta d_3}}$$

soft

$$\gamma_k^{(n)} = P(Z^{(n)}=k \mid X=x^{(n)};\theta)$$

- Yes, this formula corresponds to an inference calculation for a probabilistic mixture model.
- Yes, understanding this connection can result in systematic design of new k-means algo. variants!

K-means Clustering Outline

1) hard version: algorithm and analysis

2) soft version vanilla (aided by intuitive formula)

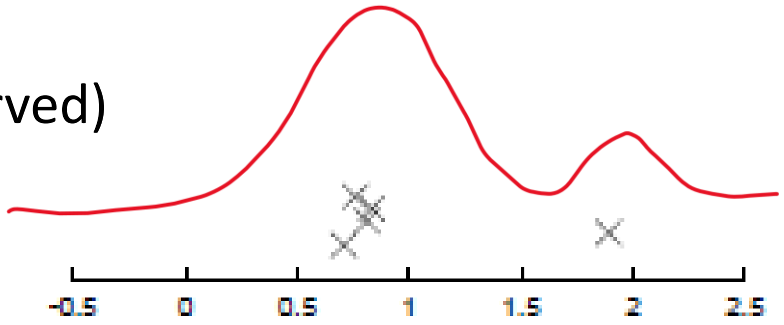
3) soft version vanilla (aided by a vanilla probabilistic model)

- **3a) (Brief) Background: Mixture density estimation via EM algorithm**
- 3b) Methodology: Soft k-means heuristic/algorithm

4) soft version enhancements (aided by an enhanced probabilistic model)

Recall: Example mixture density (very brief mention)

Mixture Model (MM) or Latent Variable Model (LVM)



$Z \sim Bernoulli(\pi)$ (latent variable, not observed)

$(X \mid Z = z) \sim \mathcal{N}(\mu_z, \sigma_z^2)$ (X is observed)

MLE of $\theta = (\boldsymbol{\pi}, \ \mu_1, \sigma_1, \ \mu_2, \sigma_2)$ based on:

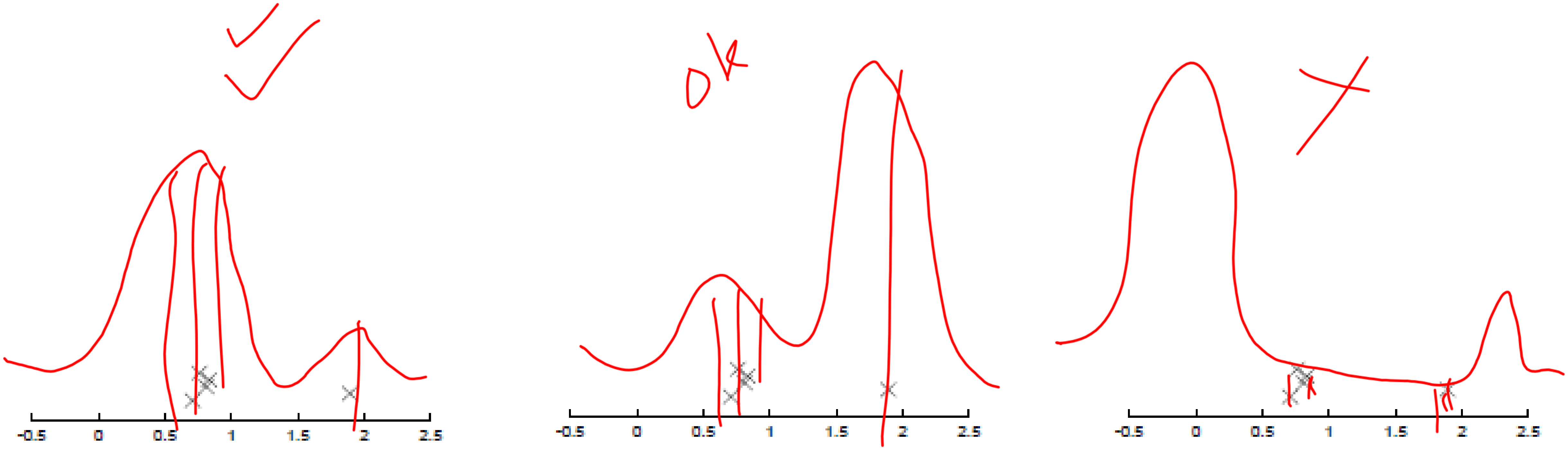
$$\mathcal{L}(\theta \mid D_N) = \prod_{n=1}^{N} \left(\boldsymbol{\pi}\, \mathcal{N}(x_n \mid \mu_1, \sigma_1^2) + (\mathbf{1} - \boldsymbol{\pi})\mathcal{N}(x_n \mid \mu_2, \sigma_2^2)\right)$$

$\hat{\theta}_{MLE} = \arg\max_{\theta} \mathcal{L}(\theta \mid D_N)$ (using numerical methods like Newton-Raphson, or ExpectationMaximization (EM) algorithm)

What is optimized in the MLE of a mixture model?

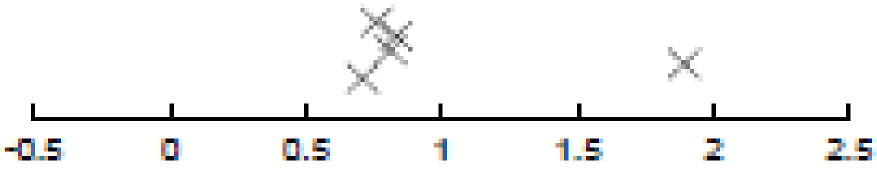
$$\theta = (\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, \pi_1)$$

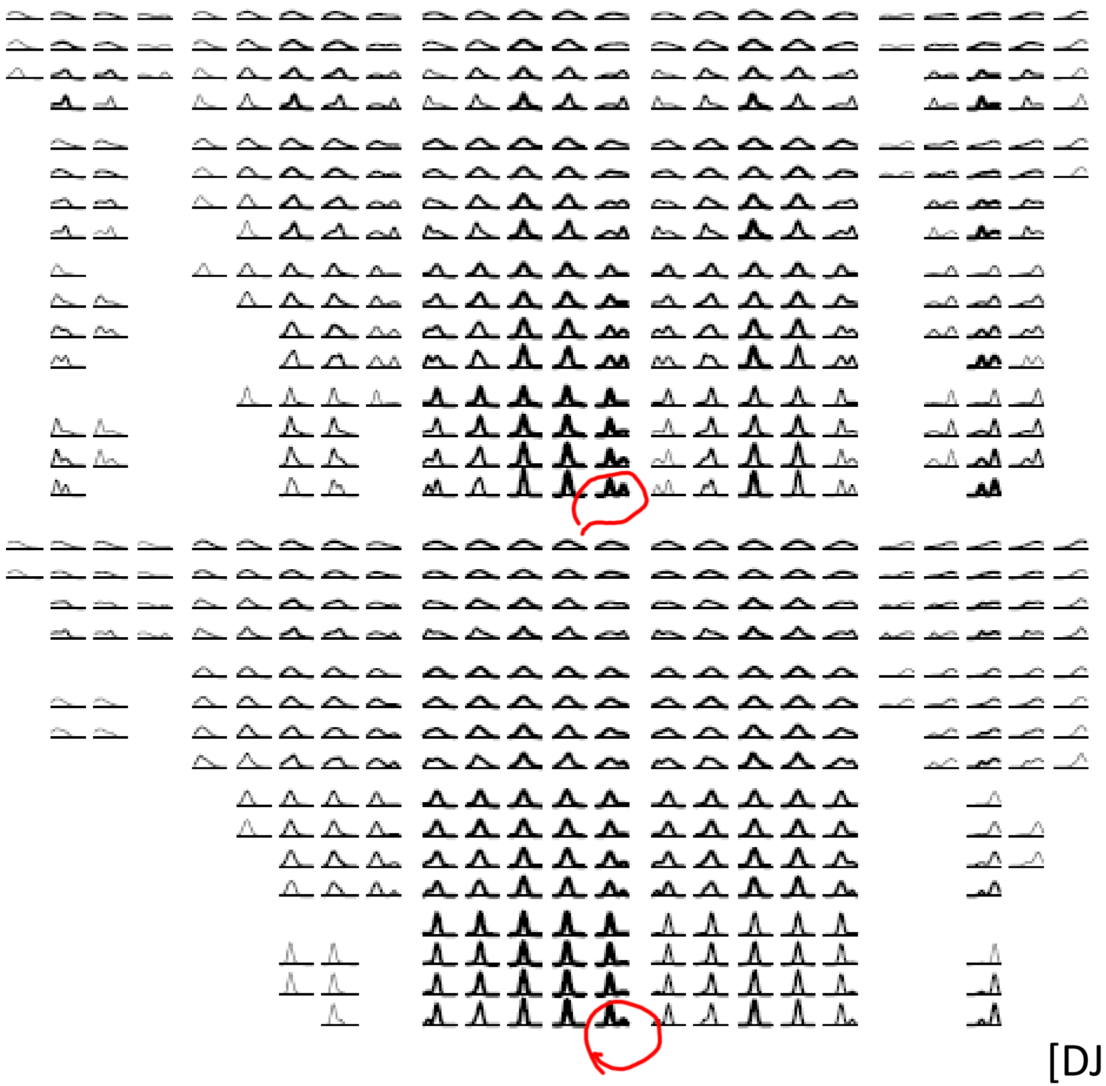
5 params.



Goal: MLE for a mixture of two 1D Gaussians ("Visual" soln.)

Estimate 5 params. now compared to 2 parameters before!

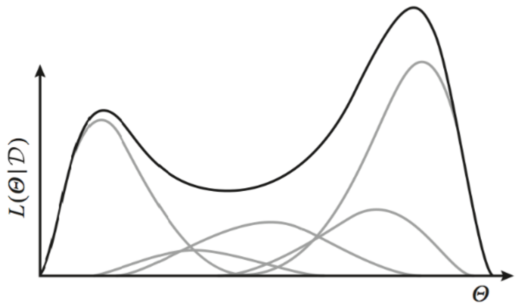




[DJM]

From visual to automated learning of a mixture model: Why is it hard?

$$\hat{\theta}_{ML} = \arg\max_{\theta} \ \mathbf{log} \ (\ P(\boldsymbol{x} \, ; \, \theta) \ )$$
$$= \arg\max_{\theta} \ \mathbf{log} \ (\ \textstyle\sum_{\boldsymbol{z}} \ P(\boldsymbol{x}, \boldsymbol{z} \, ; \, \theta) \ )$$



- "log of sum"
  - necessitates an approximate learning approach*
    - *(fineprint: though GMMs are a tractable special case studied by theoretical CS community as well, we use them to motivate a generic algo. for mixture model density estimation.)
  - EM algo. is one such approach that pushes the log across the sum!
    - E-step: do inference
    - M-step: do estimation of params. using the inferred probabs.

[ECL]

Example (for GMM vs. 1D Gaussian): Why is it hard to optimize using basic calculus/optn.?

$f : \underline{LL(\theta; D)}$

$\frac{\partial f}{\partial \mu_1} = 0$

GMM: coupled eqns

$\frac{\partial f}{\partial \mu} = 0 \quad \frac{\partial f}{\partial \sigma} = 0$

$\hat{\mu}_{ML} = \frac{\sum x_n}{N}$

$g(\hat{\sigma}, \hat{\mu}) = 0$

1D Gauss.

A **Newton-Raphson** algorithm for function optimization is possible for GMM estimation -- see Exercise 22.5 (Pg. 302 of [DJM]) – but it yields the same E/M steps of the EM algo! So, we will look at the EM algo.

[CMB]

Blank space for illustrations

- Inference probability? $r_k^{(n)} = p(z^{(n)} = k \mid x = x^{(n)}; \theta^t)$

Recall: MLE for one 1D Gaussian (closed-form soln.) – ***<u>How to change it to learn mixture of Gaussians?</u>***

- Log likelihood:

$$\ln P\left(\left\{x^{(n)}\right\}_{n=1}^{N} \mid \mu, \sigma\right) = -N \ln(\sqrt{2\pi}\sigma) - \sum_{n} \left(x^{(n)} - \mu\right)^2 / (2\sigma^2)$$

- MLE estimates:

$$\hat{\mu} = \sum_{n=1}^{N} \frac{x^{(n)}}{N}, \quad \hat{\sigma}_N^2 = \frac{\sum_{n=1}^{N} \left(x^{(n)} - \hat{\mu}\right)^2}{N}$$

Recall: MLE for one 1D Gaussian (closed-form soln.) – ***<u>How to change it to learn mixture of Gaussians?</u>***

- Log likelihood:

$$\ln P\left(\left\{x^{(n)}\right\}_{n=1}^{N} \mid \mu, \sigma\right) = -N \ln(\sqrt{2\pi}\sigma) - \sum_n \left(x^{(n)} - \mu\right)^2 / (2\sigma^2)$$

θ^t = current params. $r_k^{(n)} = P(z^{(n)} = k \mid x = x^{(n)}; \theta^t)$

$\downarrow$

$\theta^{t+1} = ??$

- MLE estimates:

$$\hat{\mu}^{(k)}_N = \sum_{n=1}^{N} \frac{r_k^{(n)} x^{(n)}}{\cancel{N}\ R^{(k)}}, \quad \hat{\sigma}_N^{2\,(k)} = \frac{\sum_{n=1}^{N} r_k^{(n)} \left(x^{(n)} - \hat{\mu}\right)^2}{\cancel{N}\ R^{(k)}}$$

(wted. sample mean & variance)

EM-algorithm in Action: Parameter-learning of a Mixture of 1D Gaussians (aka univariate GMM)

Calculate:

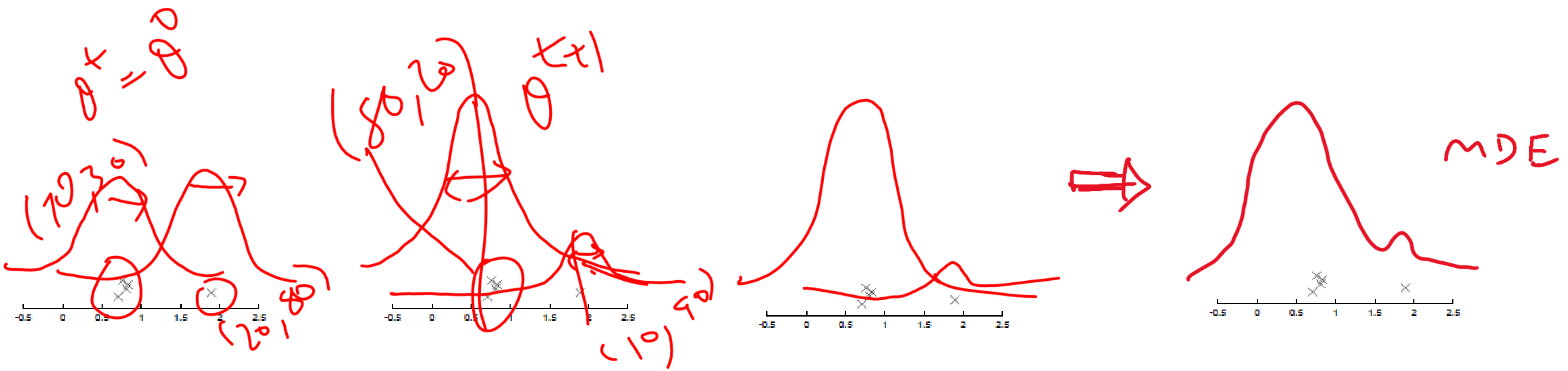
$$r_k(n) = p(Z=k \mid \theta^t, x) = \frac{\pi_k N(x \mid \mu_k, \sigma_k^2)}{\sum_{k'=1}^{K} \pi_{k'} N(x \mid \mu_{k'}, \sigma_{k'}^2)}$$

E-step (Assign)

$$\hat{\mu}_k = \sum_{n=1}^{N} \frac{r_k(n)\, x_n}{R_k} \; ; \quad \hat{\sigma}_k^2 = \sum_{n=1}^{N} \frac{r_k(n)}{R_k} (x_n - \hat{\mu}_k)^2$$

$$\pi_k = \frac{R_k}{\sum_{k'} R_{k'}} = \frac{R_k}{N} \quad (\theta_{t+1})$$

M-step (Update)



[ECL]

Blank space for illustrations

K-means Clustering Outline

1) hard version: algorithm and analysis

2) soft version vanilla (aided by intuitive formula)

3) soft version vanilla (aided by a vanilla probabilistic model)

  3a) (Brief) Background: Mixture density estimation via EM algorithm

  3b) Methodology: Soft k-means heuristic/algorithm

4) soft version enhancements (aided by an enhanced probabilistic model)

Soft K-means motivation: clustering == mixture density estimation

- Soft K-means: Views clustering problem as inferring hidden cluster labels in a mixture model of two Gaussians.
- Inference task: What is $P(Z = k \mid X = x, \theta)$
  - **Answers our final question of interest, given that $\boldsymbol{\theta}$ (its optimal/MLE value) is known**
  - Also, a subroutine for the important task of MLE below!

$\gamma_k^{(n)} = P(Z^{(n)} = k \mid X = x^{(n)}, \theta)$

- Parameter learning: Use MLE using EM algorithm to find the optimal value of θ:

$$\hat{\theta}_{ML} = argmax_{\theta}\, P(Data \mid Model_{\theta}) = \prod_{n=1}^{N} \left(\boldsymbol{\pi}\, \mathcal{N}(x_n \mid \mu_1, \sigma_1^2) + (\mathbf{1} - \boldsymbol{\pi}) \mathcal{N}(x_n \mid \mu_2, \sigma_2^2)\right)$$

[DJM]

Blank space for illustrations

Simplifying assumptions abt. GMM

Assume:

1) MM of 1D Gaussians $(I=1)$ $(x^{(n)} \in \mathbb{R}^I)$

2) $\sigma_1 = \sigma_2 = \cdots = \sigma_K = \sigma$ (known)

3) $\pi_1 = \pi_2 = \cdots = \pi_K = \frac{1}{K}$ (known)

EM-algorithm in Action: Parameter-learning of a Mixture of 1D Gaussians (aka univariate GMM) – **BEFORE SIMPLIFICATION**

Calculate:

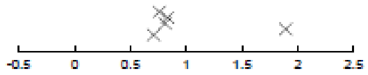
$$r_k(n) = p(Z=k \mid \theta^t, x) = \frac{\pi_k N(x \mid \mu_k, \sigma_k^2)}{\sum_{k'=1}^{K} \pi_{k'} N(x \mid \mu_{k'}, \sigma_{k'}^2)}$$

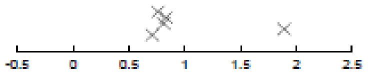
E-step (Assign)

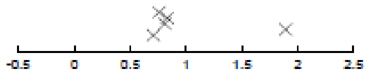
$$\hat{\mu}_k = \sum_{n=1}^{N} \frac{r_k(n) x_n}{R_k} \; ; \; \hat{\sigma}_k^2 = \sum_{n=1}^{N} \frac{r_k(n)}{R_k} (x_n - \hat{\mu}_k)^2$$

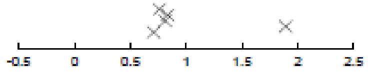
$$\pi_k = \frac{R_k}{\sum_{k'} R_{k'}} = \frac{R_k}{N} \qquad (\theta_{t+1})$$

M-step (Update)









[ECL]

EM-algorithm in Action: Parameter-learning of a Mixture of 1D Gaussians (aka univariate GMM) – **AFTER SIMPLIFICATION**

Calculate:

$$r_k(n) = p(Z=k \mid \theta^t, x) = \frac{N(x \mid \mu_k, \sigma^2)}{\sum_{k'=1}^{K} N(x \mid \mu_{k'}, \sigma^2)}$$

E-step (Assign)

$$\hat{\mu}_k = \sum_{n=1}^{N} \frac{r_k(n)\, x_n}{R_k} \;;$$

2

For all $k = 1, \dots, K$

$\sigma_k = \sigma$ (known)

$\pi_k = \frac{1}{K}$ (known)

(θ_{t+1})

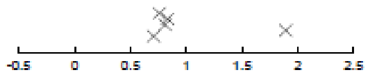
M-step (Update)

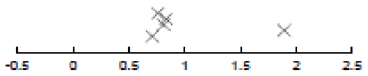
$=$

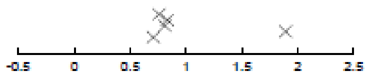
$\beta := \frac{1}{2\sigma^2}$

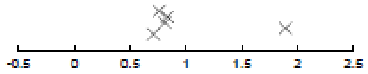
$$= \frac{e^{-\beta d(x, \mu_k)}}{\sum_{k'} e^{-\beta(x, \mu_{k'})}}$$

1









[ECL]

Blank space for illustrations

EM algo. – GMM version
(aka) **Vanilla Soft K-means**

Initialization: Set K means $\{m^{(k)}\}$ to random values.

Assignment:
(E-step) $m^{(k)} := \mu_k$

$$r_k^{(n)} = \frac{\exp\left(-\beta\, d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})\right)}{\sum_{k'} \exp\left(-\beta\, d(\mathbf{m}^{(k')}, \mathbf{x}^{(n)})\right)}. \quad (1) \quad \text{(here } \beta = 1/(2\sigma^2)\text{)}$$

Update:
(M-step)

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}} \quad (2)$$

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}.$$

[DJM]

Details: Same algo. works for multiple dimensions (I > 1)!

Assuming Gaussians across the I dimensions are **indept.** for each cluster, i.e.,
$P(x \mid Z = k', \theta) = \prod_{i=1}^{I} P(x_i \mid Z = k', \theta)$, for any cluster k'; then

1) Likelihood = ? (1D vs. I >1D)

$$P\Big(X = x \mid \theta = \{\pi_1, \ldots, \pi_K,\ m^{(1)}, \ldots, m^{(K)},\ \sigma\}\Big) = \sum_{k'=1}^{K} \pi_{k'} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{\left(x - m^{(k')}\right)^2}{2\sigma^2}\right\} = \sum_{k'} \pi_{k'} \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\beta\, d(x, m^{(k')})\Big)$$

$$P(X = x \mid \theta) = \sum_{k'=1}^{K} \pi_{k'} \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{I} \exp\left\{-\frac{\sum_{i=1}^{I}\left(x_i - m_i^{(k)}\right)^2}{2\sigma^2}\right\} = \sum_{k'} \pi_{k'} \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{I} \exp\Big(-\beta\, d(x, m^{(k')})\Big)$$

2) Responsibilities = ? (if all $\pi_{k'}$ are the same)

$$r_k^{(n)} = \frac{\exp\left(-\beta\, d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})\right)}{\sum_{k'} \exp\left(-\beta\, d(\mathbf{m}^{(k')}, \mathbf{x}^{(n)})\right)}.$$

Recall: Probabilistic mixture modelling based formula ***matches*** our intuitive formula for $r_k^{(n)}$

$r_k^{(n)}$

hard

$$\frac{e^{-\beta d_k}}{e^{-\beta d_1}+e^{-\beta d_2}+e^{-\beta d_3}}$$

soft

Blank space for illustrations

Soft K-means progress check!

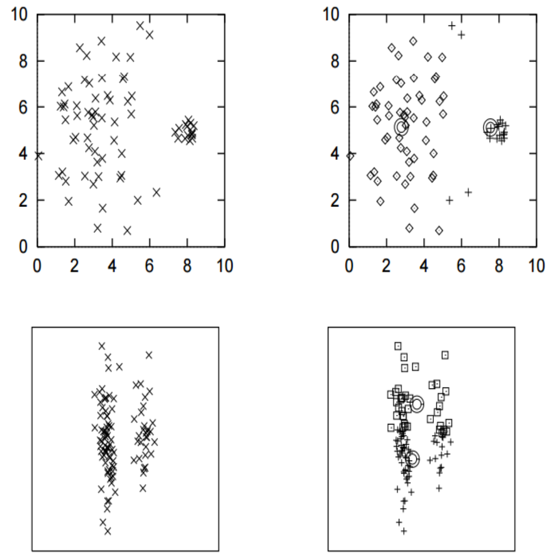
Fixes:

FUZZY ASSIGNMENTS

Doesn't fix:

"Large and small" dataset.

Long and narrow "lozenges" dataset



[DJM]

K-means Clustering Outline

1) hard version: algorithm and analysis

2) soft version vanilla (aided by intuitive formula)

3) soft version vanilla (aided by a vanilla probabilistic model)

   3a) Background: Inference+Parameter-learning of a probabilistic mixture model

   3b) Methodology: Soft k-means heuristic/algorithm

4) soft version enhancements (aided by an enhanced probabilistic model)

Enhancing soft kmeans – the idea!

1. Understand the mixture model underlying vanilla soft K-means, and how K-means is simply MLE via EM for this model:

   i) K same-width Gaussian σ for each cluster (including same σ across dimensions i.e., spherical Gaussian)

   ii) Same prior probability for each cluster: $\pi_1 \ = \pi_2 \ = \ \ldots \ = \pi_K = 1/K$

2. Generalize this model by relaxing these two model assumptions.

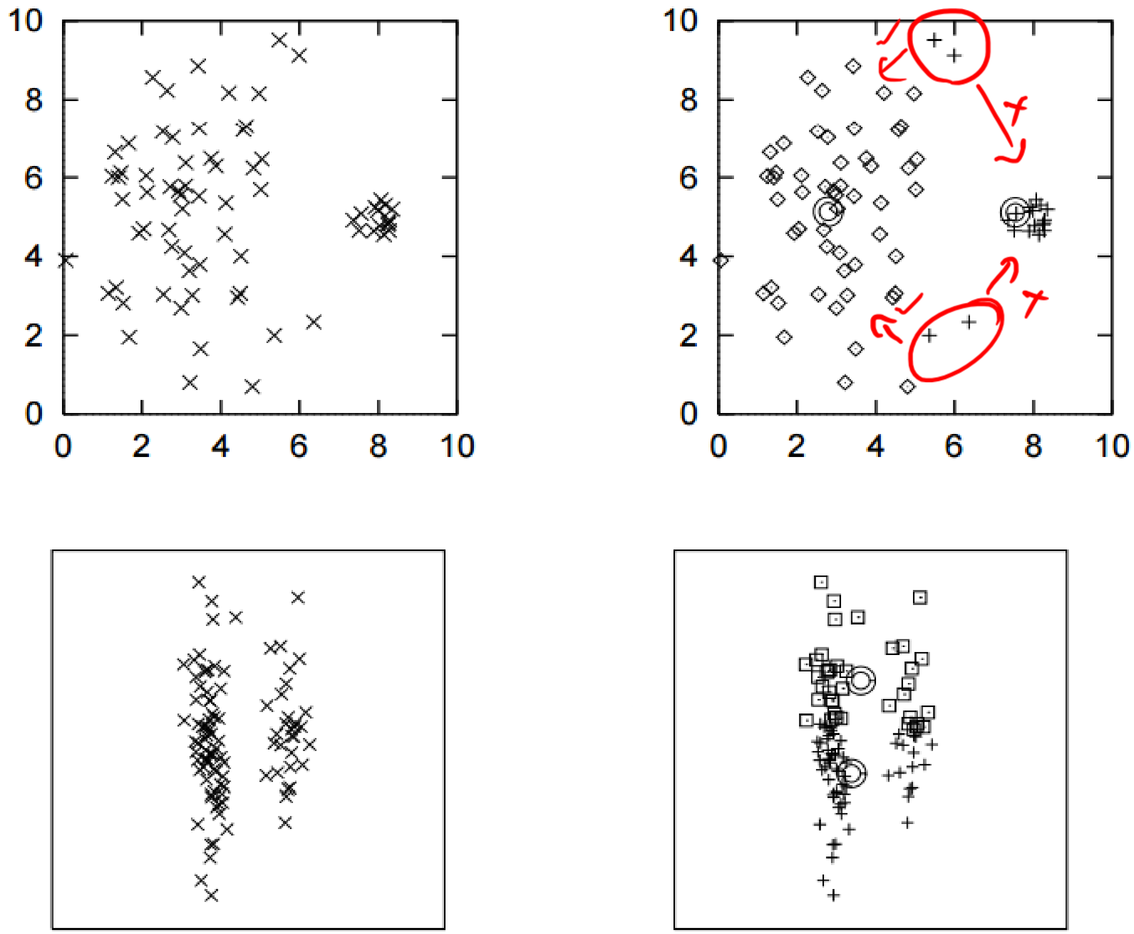
   i) Per-cluster width $\{\sigma_k\}$

   ii) Per-cluster prior $\{\pi_k\}$

Above steps offer a systematic approach to design new variants/enhancements of the K-means algorithm!!!

[DJM]

Cluster width as extra parameter?



[DJM]

Soft K-means version 2

(adapt to different *cluster widths* and *cluster proportions*)

Assignment:

$$r_k^{(n)} = \frac{\pi_k \frac{1}{(\sqrt{2\pi}\sigma_k)^I} \exp\left(-\frac{1}{\sigma_k^2} d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})\right)}{\sum_{k'} \pi_k \frac{1}{(\sqrt{2\pi}\sigma_{k'})^I} \exp\left(-\frac{1}{\sigma_{k'}^2} d(\mathbf{m}^{(k')}, \mathbf{x}^{(n)})\right)}$$

where I is the dimensionality of $\mathbf{x}$.

Update:

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

$$\sigma_k^2 = \frac{\sum_n r_k^{(n)} (\mathbf{x}^{(n)} - \mathbf{m}^{(k)})^2}{I R^{(k)}}$$

$$\pi_k = \frac{R^{(k)}}{\sum_k R^{(k)}}$$

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}.$$

[DJM]

Soft K-means version 2

(adapt to different *cluster widths* and *cluster proportions*)

$\{\sigma_k\}, \{\pi_k\}$

Assignment:

$$r_k^{(n)} = \frac{\pi_k \frac{1}{(\sqrt{2\pi}\sigma_k)^I} \exp\left(-\frac{1}{\sigma_k^2} d(\mathbf{m}^{(k)}, \mathbf{x}^{(n)})\right)}{\sum_{k'} \pi_{k'} \frac{1}{(\sqrt{2\pi}\sigma_{k'})^I} \exp\left(-\frac{1}{\sigma_{k'}^2} d(\mathbf{m}^{(k')}, \mathbf{x}^{(n)})\right)}$$

where I is the dimensionality of $\mathbf{x}$.

Update:

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{R^{(k)}}$$

$$\sigma_k^2 = \frac{\sum_n r_k^{(n)} \underbrace{\|\mathbf{x}^{(n)} - \mathbf{m}^{(k)}\|^2}_{}}{I R^{(k)}} \qquad d(x^{(n)}, m^{(k)})$$

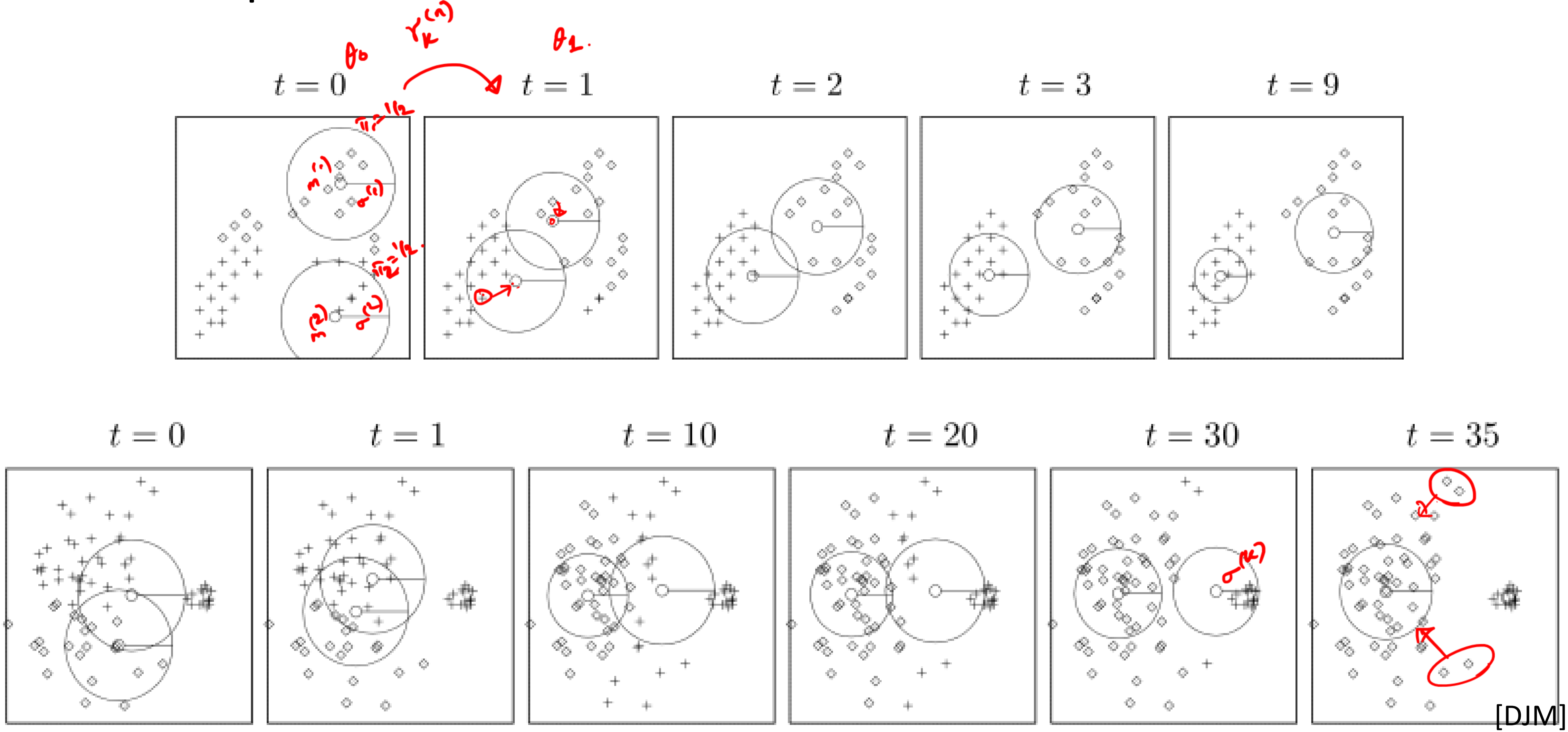
$$\pi_k = \frac{R^{(k)}}{\sum_k R^{(k)}} = \frac{R^{(k)}}{N}$$

where $R^{(k)}$ is the total responsibility of mean k,

$$R^{(k)} = \sum_n r_k^{(n)}.$$

[DJM]

Example runs of Soft K-means, version 2



[DJM]

Final enhancement we will try

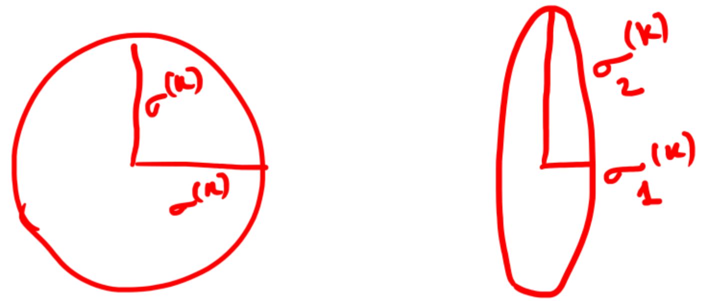
From SPHERICAL to ELLIPSOIDAL axis-aligned (indept.) Gaussians!

$\left\{ \left\{ \sigma_i^{(k)} \right\}_{i=1}^{I} \right\}_{k=1}^{K}$

So per-cluster, per-dimensionality widths.

$\sigma \longrightarrow \boxed{\sigma^{(k)}}$

π_k



each of I dimension
also get their own ~~cluster~~ width.

[DJM]

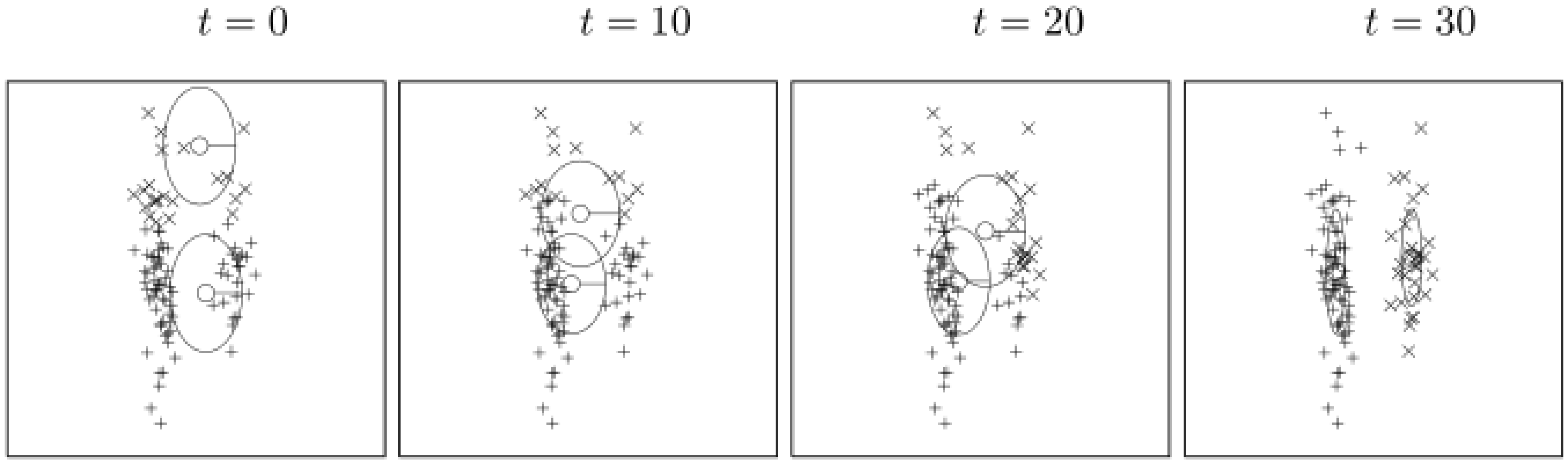
Soft k-means version 3

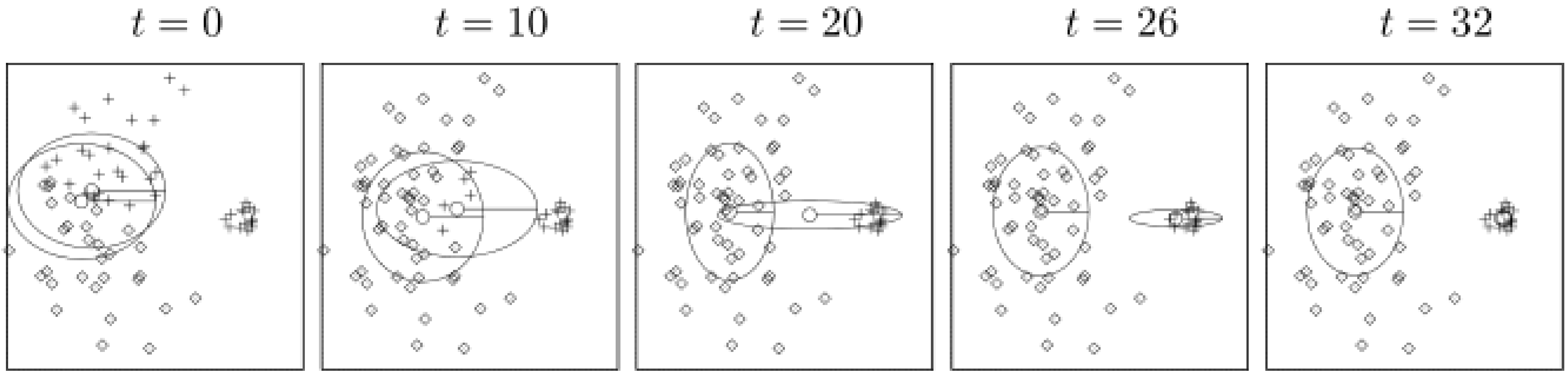
$$r_k^{(n)} = \frac{\pi_k \frac{1}{\prod_{i=1}^{I} \sqrt{2\pi}\sigma_i^{(k)}} \exp\left(-\sum_{i=1}^{I} (m_i^{(k)} - x_i^{(n)})^2 \Big/ 2(\sigma_i^{(k)})^2\right)}{\sum_{k'} (\text{numerator, with } k' \text{ in place of } k)} \quad (22.27)$$

$$\sigma_i^{2\,(k)} = \frac{\sum_n r_k^{(n)} (x_i^{(n)} - m_i^{(k)})^2}{R^{(k)}} \quad (22.28)$$

[DJM]

Example runs of Soft K-means, version 3



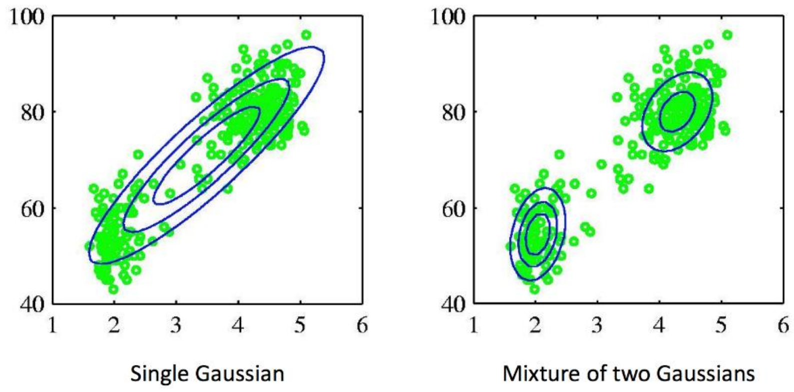


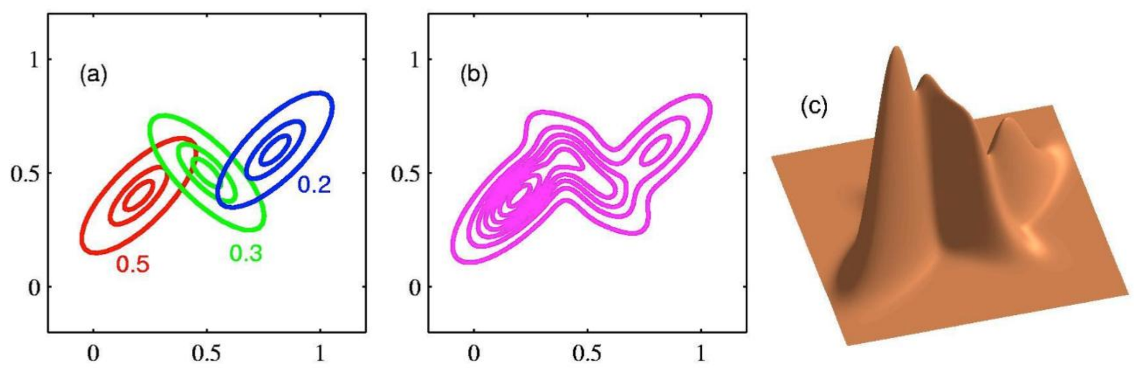
[DJM]

Soft k-means version 4?

Q: What if data is not axis-aligned i.e., the different dimensions are not independent?

A: MLE via EM algo. for general GMM





[from ErmongroupStanfordCS228CourseLecNotes]

[CMB]

Example: GMM MLE Learning via EM algo. (aka fully-enhanced Soft k-means algo. or version 4)

Assignment (E) step:

$$r_k^{(n)} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}$$

$\mu_k \in \mathbb{R}^I$

$\Sigma_k \in \mathbb{R}^{I \times I}$ (psd) (Σ_k diag. (version 3))

Update (M) step:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{R^{(k)}} \sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{R^{(k)}} \sum_{n=1}^{N} r_k^{(n)} \left(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}^{(n)} - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

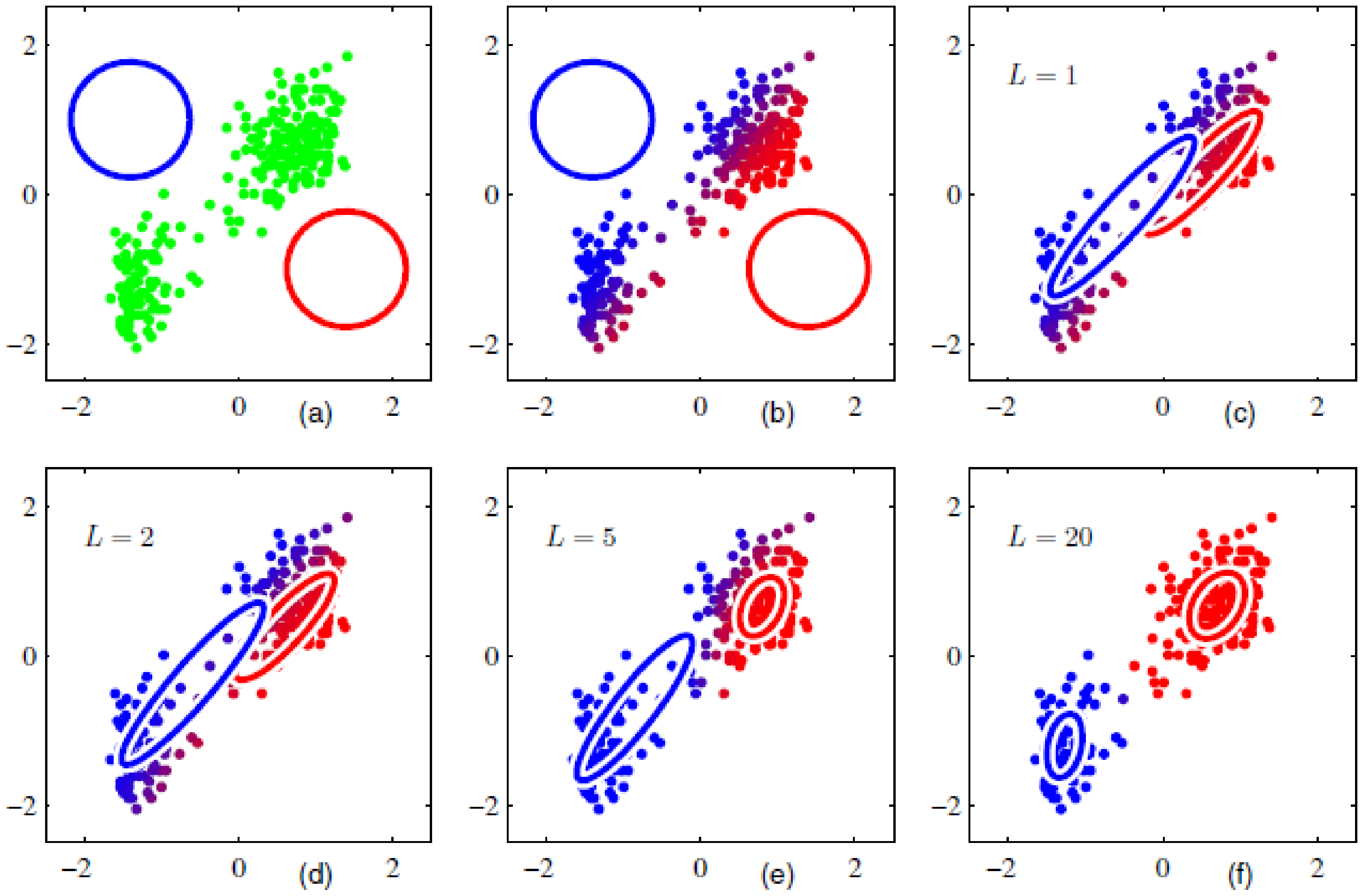
$$\pi_k^{\text{new}} = \frac{R^{(k)}}{N}$$

where

$$R^{(k)} = \sum_{n=1}^{N} r_k^{(n)}$$

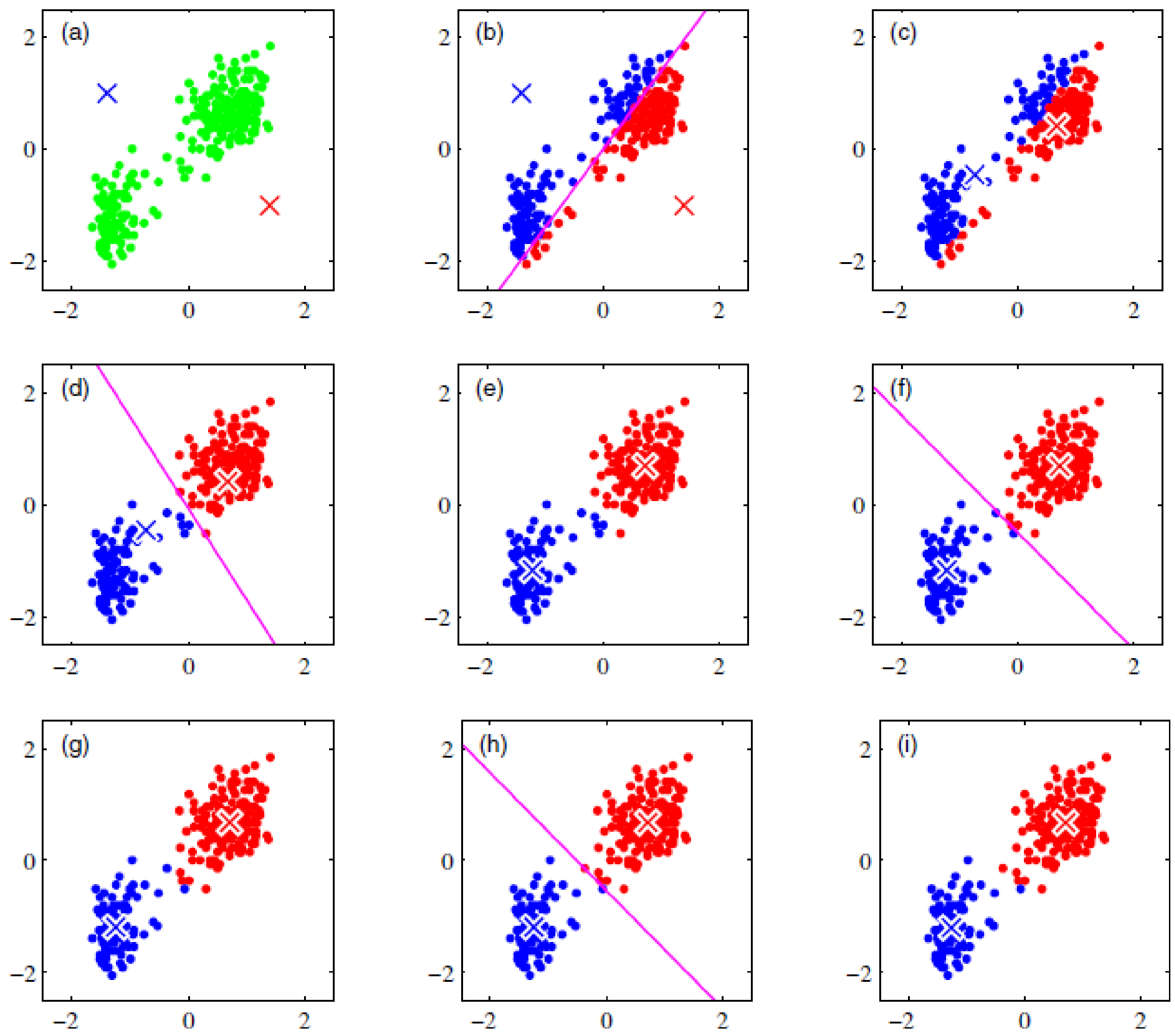
[CMB]

Fully-enhanced Soft K-means



[CMB]

Recall: Hard K-means



[CMB]

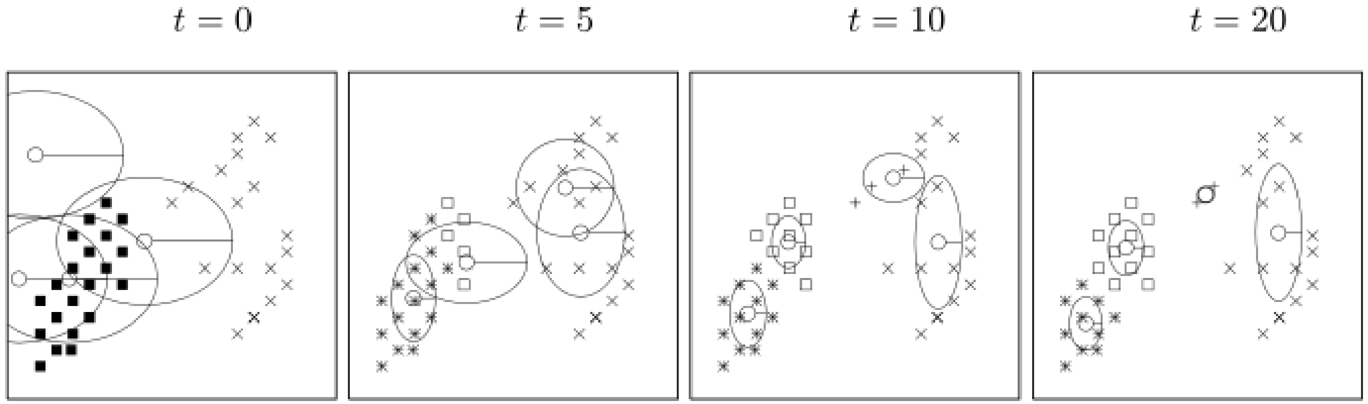
K-means Clustering Outline

1) hard version: algorithm and analysis
2) soft version vanilla (aided by intuitive formula)
3) soft version vanilla (aided by a vanilla probabilistic model)
4) soft version enhancements (aided by an enhanced probabilistic model)

Summary of “Clustering as Probabilistic Modeling”

- Probabilistic mixture modeling and associated statistical inference gives a systematic approach to design and optimize a data science task.

- Soft K-means successful in many applications (e.g. cancer subtyping), though still not perfect.
  - E.g. MLE blows up when mean sits on exactly one data point. THINK BAYESIAN INFERENCE (aka pseudocounts-like ideas)!



Thank you!