MACHINE LEARNING IN BIOINFORMATICS

Part 3: Clustering

František Mráz
KSVI MFF UK

Density Estimation

- We have already seen
  1. **Parametric**: Assume a single model for $p(x \mid C_i)$ (previous lecture)
  2. **Semiparametric**: $p(x \mid C_i)$ is a mixture of densities

     Multiple possible explanations/prototypes:

     Different handwriting styles, accents in speech
  3. **Nonparametric**: No model; data speaks for itself (next lecture)

Mixture Densities

$$p(x) = \sum_{i=1}^{k} p(x \mid G_i) P(G_i)$$

- where G_i are the components/groups/clusters,
  $P(G_i)$ mixture proportions (priors),
  $p(x \mid G_i)$ component densities

- Gaussian mixture where$p(x \mid G_i) \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$ where parameters $\Phi = \{P(G_i), \mu_i, \mathbf{\Sigma}_i\}_{i=1}^{k}$ must be estimated from the unlabeled samples (unsupervised learning) $X = \{x_t\}_{t=1}^{N}$

Classes vs. Clusters

- **Supervised**: $X = \{x_t, r_t\}_{t=1}^{N}$
- Classes $C_i, i = 1, \dots, K$

$$p(x) = \sum_{i=1}^{K} p(x|C_i)\, P(C_i)$$

where

$$p(x|C_i) \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$$

$$\Phi = \{\mathrm{P}(C_i), \mu_i, \mathbf{\Sigma}_i\}_{i=1}^{K}$$

$$\hat{P}(C_i) = \frac{\sum_t r_t^{(i)}}{N} \qquad m_i = \frac{\sum_t r_t^{(i)} x_t}{\sum_t r_t^{(i)}}$$

$$\mathrm{S}_i = \frac{\sum_t r_t^{(i)} (x_t - m_i)(x_t - m_i)^{\mathrm{T}}}{\sum_t r_t^{(i)}}$$

Sample covariance matrix

- **Unsupervised**: $X = \{x_t\}_{t=1}^{N}$
- Clusters $G_i, i = 1, \dots, k$

$$p(x) = \sum_{i=1}^{K} p(x|G_i)\, P(G_i)$$

where

$$p(x|G_i) \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$$

$$\Phi = \{\mathrm{P}(G_i), \mu_i, \mathbf{\Sigma}_i\}_{i=1}^{K}$$

Labels, $r_t^{(i)}$?

$$r_t^{(i)} = \begin{cases} 1 & \text{if } x_t \in C_i \\ 0 & \text{if } x_t \notin C_i \end{cases}$$

k-Means Clustering (a nonparametric algorithm)

- Find k **reference vectors** (prototypes / codebook vectors / codewords) which best represent data
- Reference vectors, m_j , $j = 1, \ldots, k$
- Use nearest (most similar) reference:

$$\|x_t - m_i\| = \min_j \|x_t - m_j\|$$

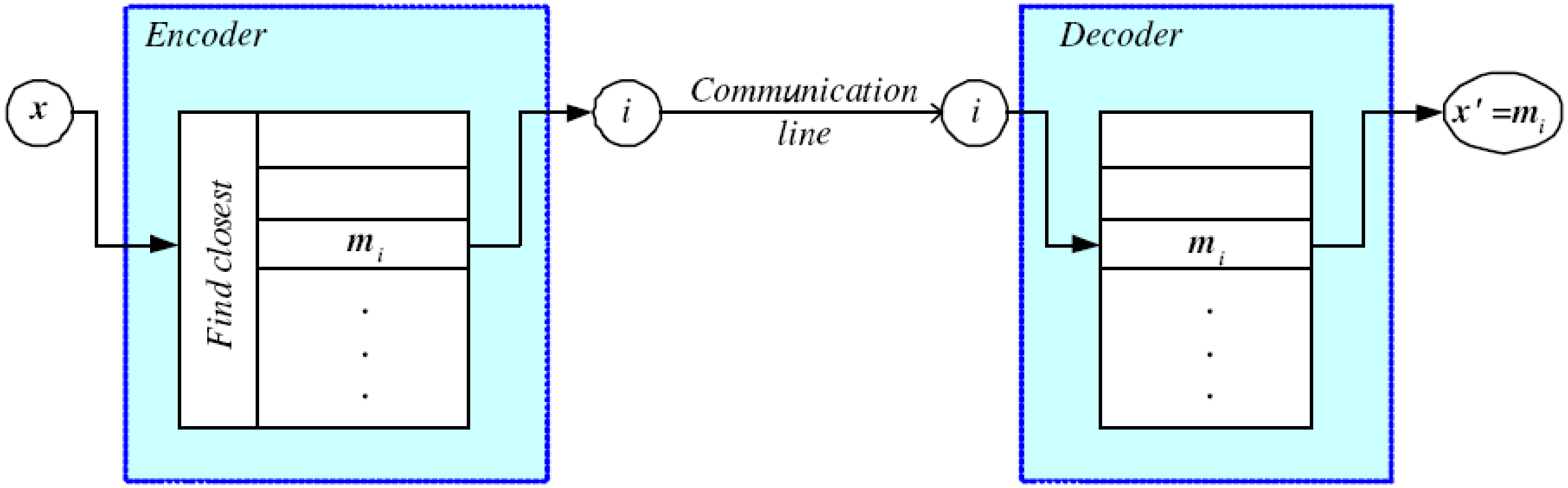
- Reconstruction error

error

$$E\left(\{m_i\}_{i=1}^{k} \middle| \mathbf{X}\right) = \sum_t \sum_i b_t^{(i)} \|x_t - m_i\|$$

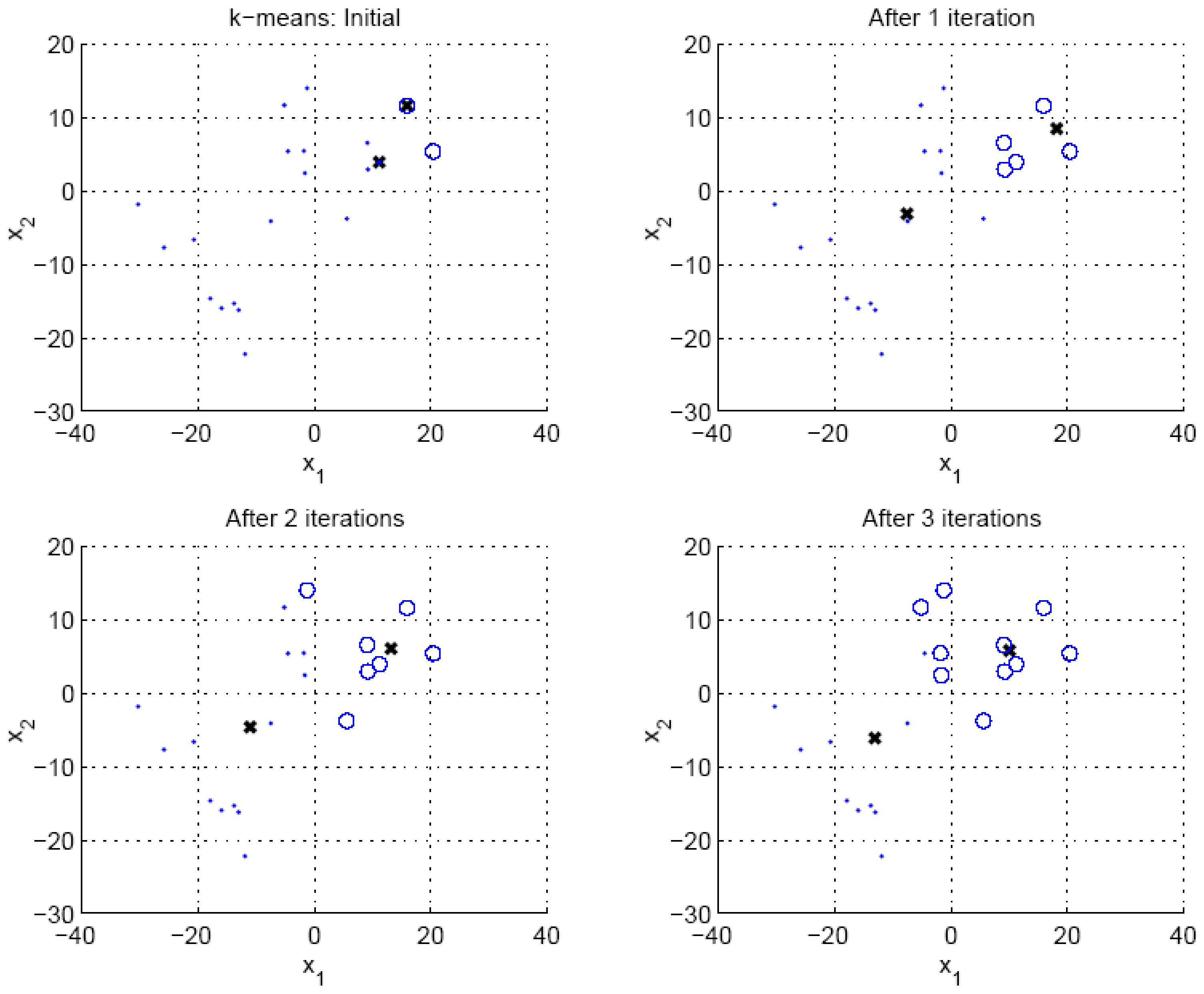
$$b_t^{(i)} = \begin{cases} 1 & \text{if } \|x_t - m_i\| = \min_j \|x_t - m_j\| \\ 0 & \text{otherwise} \end{cases}$$

Encoding/Decoding



k-means Clustering

```
Initialize $m_i, i = 1, \ldots, k$, for example to $k$ random $x_t$
Repeat
    for all $x_t \in \{x_t\}_{t=1}^N$
        $b_t^{(i)} = \begin{cases} 1 & \text{if } \|x_t - m_i\| = \min_j \|x_t - m_j\| \\ 0 & \text{otherwise} \end{cases}$
    for all $m_i, i = 1, \ldots, k$
        $m_i := \frac{\sum_t b_t^{(i)} x_t}{\sum_t b_t^{(i)}}$
Until $m_i$ converge
```


k−means: Initial
After 1 iteration
After 2 iterations
After 3 iterations
x1
x2

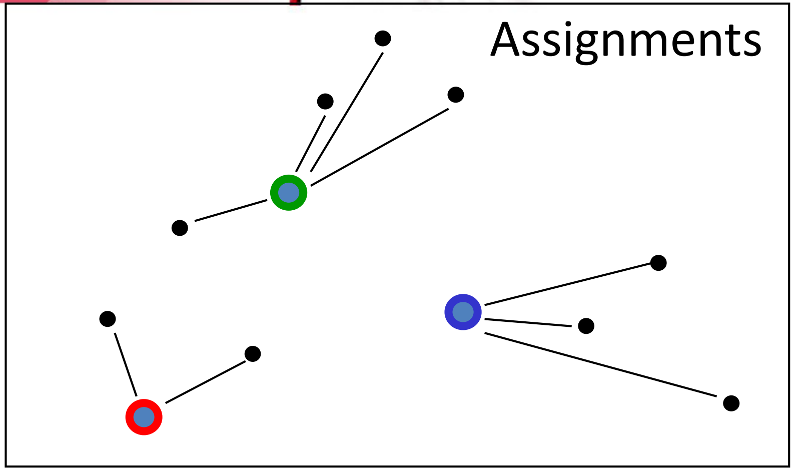
The k-Means Algorithm

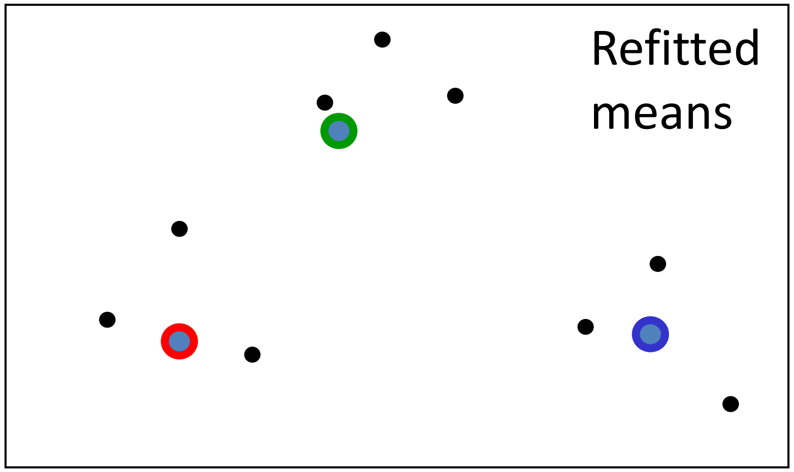
- Assume the data lives in a Euclidean space.
- Assume we want k classes.
- Assume we start with randomly located cluster centers

The algorithm alternates between two steps:

1. **Assignment step**: Assign each datapoint to the closest cluster.

2. **Refitting step**: Move each cluster center to the center of gravity of the data assigned to it.





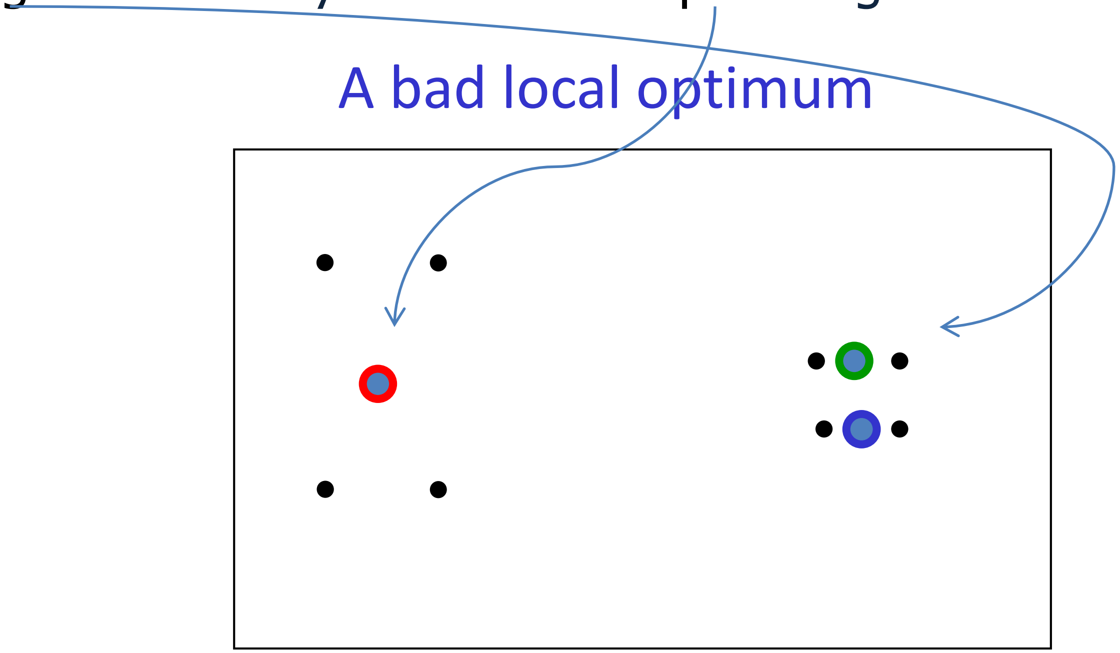
Why K-Means Converges

- Whenever an assignment is changed, the sum squared distances of datapoints from their assigned cluster centers is reduced.
- Whenever a cluster center is moved the sum squared distances of the datapoints from their currently assigned cluster centers is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged.

Local Minima

- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves: Simultaneously merge two nearby clusters and split a big cluster into two.

A bad local optimum



Soft k-Means

- Instead of making hard assignments of datapoints to clusters, we can make soft assignments. One cluster may have a responsibility of 0.7 for a datapoint and another may have a responsibility of 0.3.
  - Allows a cluster to use more information about the data in the refitting step.
  - What happens to our convergence guarantee?
  - How do we decide on the soft assignments?

Expectation-Maximization (EM)

- In k-means we minimized the total reconstruction error; original data are $X = \{x_t\}_{i=1}^{N}$

- Next we will we look for the component density parameters that maximize the likelihood of the sample

- Log likelihood with a mixture model $\mathcal{L}(\Phi|X) = \log \prod_t p(x_t|\Phi) =$

Parameter vector of the model

$$= \sum_t \log \sum_{i=1}^{k} p(x_t|\mathrm{G_i})P(G_i)$$

- Assume hidden variables Z, which when known, make optimization much simpler
- Complete likelihood, $\mathcal{L}_c(\Phi\,|X, Z)$, in terms of X and Z
- Incomplete likelihood, $\mathcal{L}(\Phi\,|X)$, in terms of X

E- and M-steps

Iterate the two steps

1. **E-step:** Estimate Z given X and current Φ
2. **M-step:** Find new Φ' given Z, X, and old Φ:

E-step: $\mathcal{Q}\left(\Phi \middle| \Phi^{(l)}\right) = E[\mathcal{L}_c(\Phi|X,Z)|X,\Phi^{(l)})]$

M-step: $\Phi^{(l+1)} = \arg\max_{\Phi} \mathcal{Q}(\Phi|\Phi^{(l)})$

An increase in $\mathcal{Q}$ (the expected likelihood) increases incomplete likelihood

$$\mathcal{L}\left(\Phi^{(l+1)} \middle| X\right) \geq \mathcal{L}(\Phi^{(l)}|X)$$

EM in Gaussian Mixtures

Indicator variables

- $z_t^{(i)} = 1$ if x_t belongs to G_i, 0 otherwise (it corresponds to labels $r_t^{(i)}$ of supervised learning); assume $p(x|G_i) \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$
- **E-step:**

$$E\left[z_t^{(i)} \middle| X, \Phi^{(l)}\right] = E[z_t^{(i)} | x_t, \Phi^{(l)}]$$

x_t are iid

$$= P\left(z_t^{(i)} = 1 \middle| x_t, \Phi^{(l)}\right)$$

$z_t^{(i)}$ is a 0/1 random variable

$$= \frac{p\left(x_t \middle| z_t^{(i)} = 1, \Phi^{(l)}\right) P(z_t^{(i)} = 1 | \Phi^{(l)})}{p(x_t | \Phi^{(l)})}$$

Bayes' rule

$$= \frac{p\left(x_t \middle| G_i, \Phi^{(l)}\right) P(G_i)}{\sum_j p\left(x_t \middle| G_j, \Phi^{(l)}\right) P(G_j)}$$

$$= P\left(G_i \middle| x_t, \Phi^{(l)}\right) =: h_t^{(i)}$$

New notation

EM in Gaussian Mixtures

- **M-step:**

$$\Phi^{(l+1)} = \arg\max_{\Phi} Q\left(\Phi \middle| \Phi^{(l)}\right) = \arg\max_{\Phi} E[\mathcal{L}_c(\Phi|X,Z)|X,\Phi^{(l)})]$$

$$= \arg\max_{\Phi} \sum_t \sum_i E\left[z_t^{(i)} \middle| X, \Phi^{(l)}\right] \cdot [\log P(G_i) + \log p(x_t|G_i,\Phi^{(l)}]$$

$$= \arg\max_{\Phi} \sum_t \sum_i h_t^{(i)} \log P(G_i) + \sum_t \sum_i h_t^{(i)} \cdot \log p(x_t|G_i,\Phi^{(i)})$$

- Solution using Lagrange multiplier method

$$P(G_i) = \frac{\sum_t h_t^{(i)}}{N} \qquad m_i^{(l+1)} = \frac{\sum_t h_t^{(i)} x_t}{\sum_t h_t^{(i)}}$$

Use estimated labels in place of unknown labels

$$\mathbf{S}_t^{(l+1)} = \frac{\sum_t h_t^{(i)} \left(x_t - m_i^{(l+1)}\right)\left(x_t - m_i^{(l+1)}\right)^T}{\sum_t h_t^{(i)}}$$

EM in Gaussian Mixtures

- For Gaussian components in the E-step

$$h_t^{(i)} = \frac{p(G_i)|\mathbf{S}_i|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(x_t - m_i)^T \mathbf{S}_i^{-1}(x_t - m_i)\right]}{\sum_j p(G_j)|\mathbf{S}_j|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(x_t - m_j)^T \mathbf{S}_i^{-1}(x_t - m_j)\right]}$$

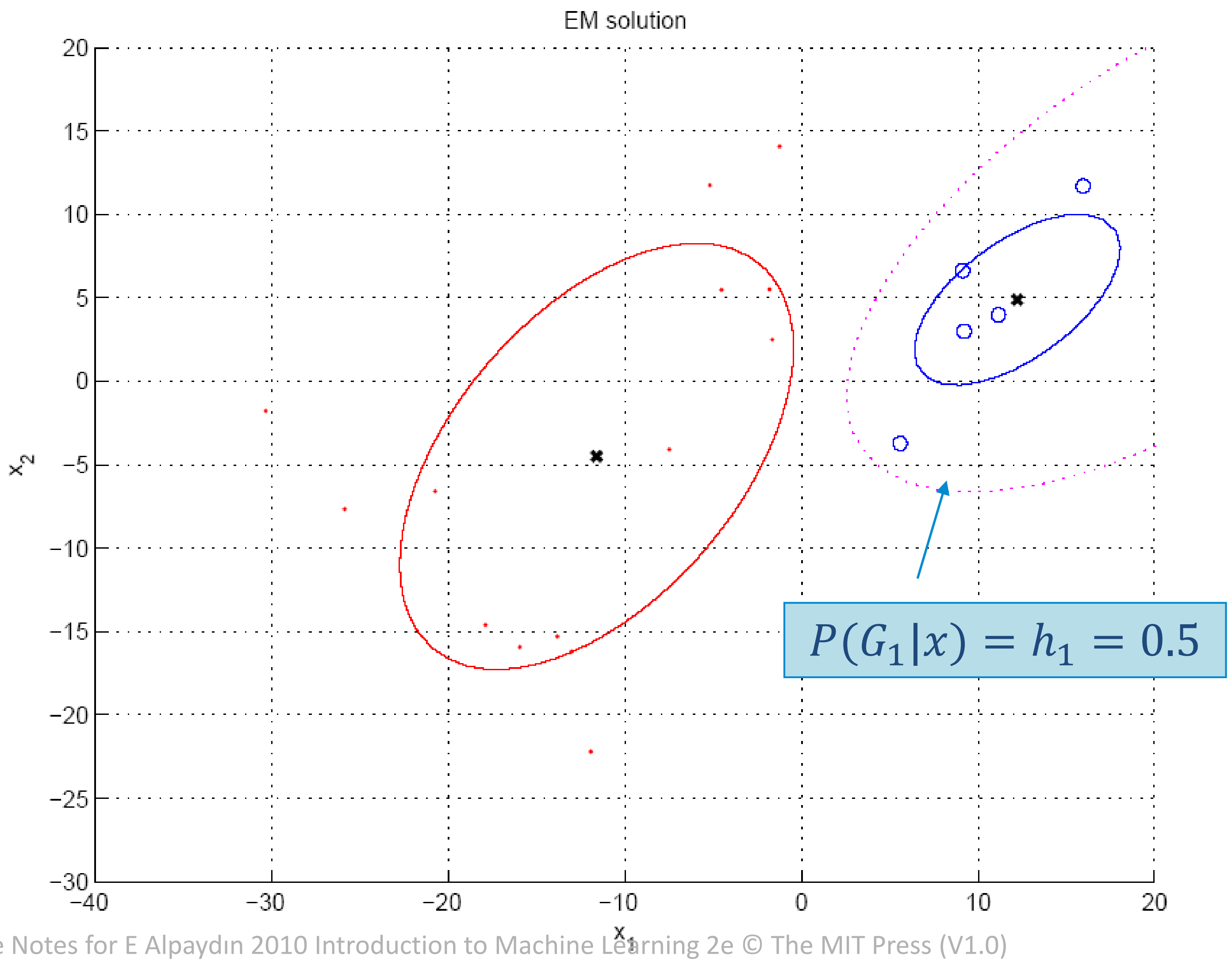
- EM is initialized by k-means – it estimates m_i
- Then we estimate $\mathbf{S}_i$ and compute $p(G_i)$ as

$$\frac{\sum_t b_t^{(i)}}{N}$$

- By making a few simplifying assumptions (the same diagonal covariance matrix for all clusters) we obtain that k-means clustering is a special case of EM applied to Gaussian mixtures where inputs are assumed independent with equal and shared variances, all components have equal priors, and labels are hardened

EM in Gaussian Mixtures

- k-means thus pave the input density with circles, whereas EM in the general case uses ellipses of arbitrary shapes, orientations, and coverage proportions



$P(G_1|x) = h_1 = 0.5$

Mixtures of Latent Variable Models

- Using full covariance matrices with Gaussian mixtures, even if there is no singularity, can cause overfitting if the input dimensionality is high and the sample is small → regularize clusters to decrease the number of parameters:
  - Assuming a common covariance matrix may not help as clusters may have different shapes.
  - Assuming diagonal covariance matrices is even more risky because it removes all correlations.
  - Use PCA/FA to decrease dimensionality: Mixtures of PCA/FA

$$p(x_t|G_i) = \mathcal{N}(m_i, \mathbf{V}_i\mathbf{V}_i^T + \mathbf{\Psi}_i)$$

Factor loadings of G_i

Specific variance of G_i

Can use EM to learn V_i (Ghahramani and Hinton, 1997; Tipping and Bishop, 1999)

After Clustering

- Dimensionality reduction methods find correlations between features and group features
- Clustering methods find similarities between instances and group instances
- Clustering allows knowledge extraction through

  number of clusters,

  prior probabilities,

  cluster parameters, i.e., center, range of features.
- Example: customer relationship management (CRM)
  - First clustering = customer segmentation
  - Then different strategies for different types of customers

Clustering as Preprocessing

- Estimated group labels $h^{(j)}$ (soft) or $b^{(j)}$ (hard) may be seen as the dimensions of a new k-dimensional space, where we can then learn our discriminant or regressor.
- Local representation (only one $b^{(i)}$ is 1, all others are 0; only few $h^{(j)}$ are nonzero) vs Distributed representation (After PCA; all $z^{(j)}$ are nonzero)

Mixture of Mixtures

- In classification, the input comes from a mixture of classes (supervised).
- If each class is also a mixture, e.g., of Gaussians, (unsupervised), we have a mixture of mixtures:

$$p(x|C_i) = \sum_{j=1}^{k_i} p(x|G_{ij})P(G_{ij})$$

$$p(x) = \sum_{i=1}^{K} p(x|C_i)P(C_i)$$

  - k_i is is the number of components of class C_i
  - G_{ij} is the component j of class C_i

Hierarchical Clustering

- Cluster based on similarities/distances
- Distance measure between instances x_r and x_s

Minkowski (L_p) (Euclidean for $p = 2$)

$$d_m(x_r, x_s) = \left[\sum_{j=1}^{d} (x_{rj} - x_{sj})^p\right]^{\frac{1}{p}}$$

City-block distance

$$d_{cb}(x_r, x_s) = \sum_{j=1}^{d} \left|x_{rj} - x_{sj}\right|$$

Agglomerative Clustering

- Start with N groups each with one instance and merge two closest groups at each iteration
- Distance between two groups G_i and G_j:
  - Single-link:

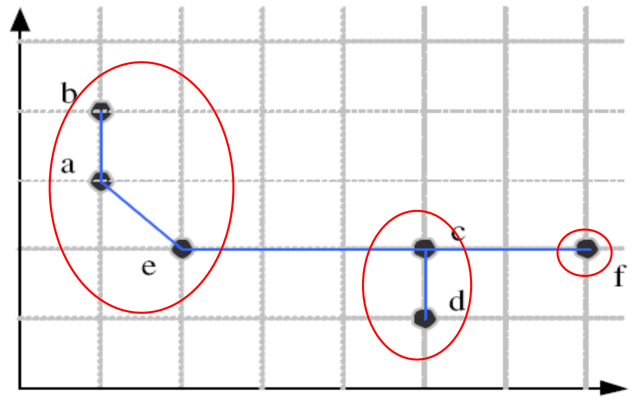
$$d(G_i, G_j) = \min_{x_r \in G_i, x_s \in G_j} d(x_r, x_s)$$

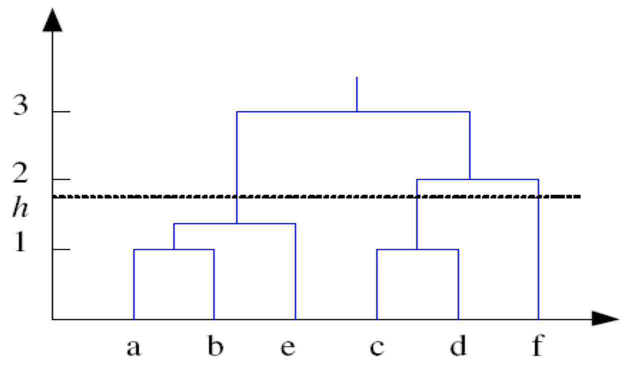
  - Complete-link:

$$d(G_i, G_j) = \max_{x_r \in G_i, x_s \in G_j} d(x_r, x_s)$$

  - Average-link: the average of distances between all pairs
  - centroid distance: the distance between the centroids

Example: Single-Link Clustering





Dendrogram

- Two instances are grouped together at level h *if the* distance between them is less than h*, or if there is an intermediate sequence* of instances between them such that the distance between consecutive instances is less than h*.*
- in the complete-link method, all instances in a group have a distance less than *h between* them

Choosing k

- Defined by the application, e.g., image quantization
- Plot data (after PCA) and check for clusters
- Incremental (leader-cluster) algorithm: Add one at a time until "elbow" (reconstruction error / log likelihood/ intergroup distances)
- Manually check for meaning

Clustering

- We assume that the data was generated from a number of different classes. The aim is to cluster data from the same class together.
  - How do we decide the number of classes?
  - Why not put each data point into a separate class?
- What is the objective function that is optimized by sensible clusterings?

A Generative View of Clustering

- We need a sensible measure of what it means to cluster the data well.
  - This makes it possible to judge different methods.
  - It may make it possible to decide on the number of clusters.
- An obvious approach is to imagine that the data was produced by a generative model.
  - Then we can adjust the parameters of the model to maximize the probability that it would produce exactly the data we observed.