

LTAT.02.004 MACHINE LEARNING II

## **Expectation-maximisation algorithm**

Sven Laur  
University of Tartu

## Quick recap of hard clustering

**Model.** A hard-clustering algorithm is based on a probabilistic model

$$p[\mathbf{x}_1, \dots, \mathbf{x}_n | \mathbf{z}, \Theta]$$

where  $\mathbf{z}$  denotes *unknown* labels and  $\Theta$  captures all model parameters.

**Optimisation task.** The aim of hard-clustering algorithm is to minimise

$$F(\mathbf{z}, \Theta) = -\log p[\mathbf{x}_1, \dots, \mathbf{x}_n | \mathbf{z}, \Theta] \ .$$

If all data points are independent form each other then we can simplify

$$F(\mathbf{z}, \Theta) = -\sum_{i=1}^n \log p[\mathbf{x}_i | z_i, \Theta_{z_i}]$$

where  $\Theta_{z_i}$  denotes parameters for the  $z_i$ th data source.

## Two-step minimisation algorithm

**M1 step.** Find a new labelling  $z$  that minimises  $F(z, \Theta)$  for fixed  $\Theta$ . Due to the form of  $F(z, \Theta)$  the label can be sought for each data point separately:

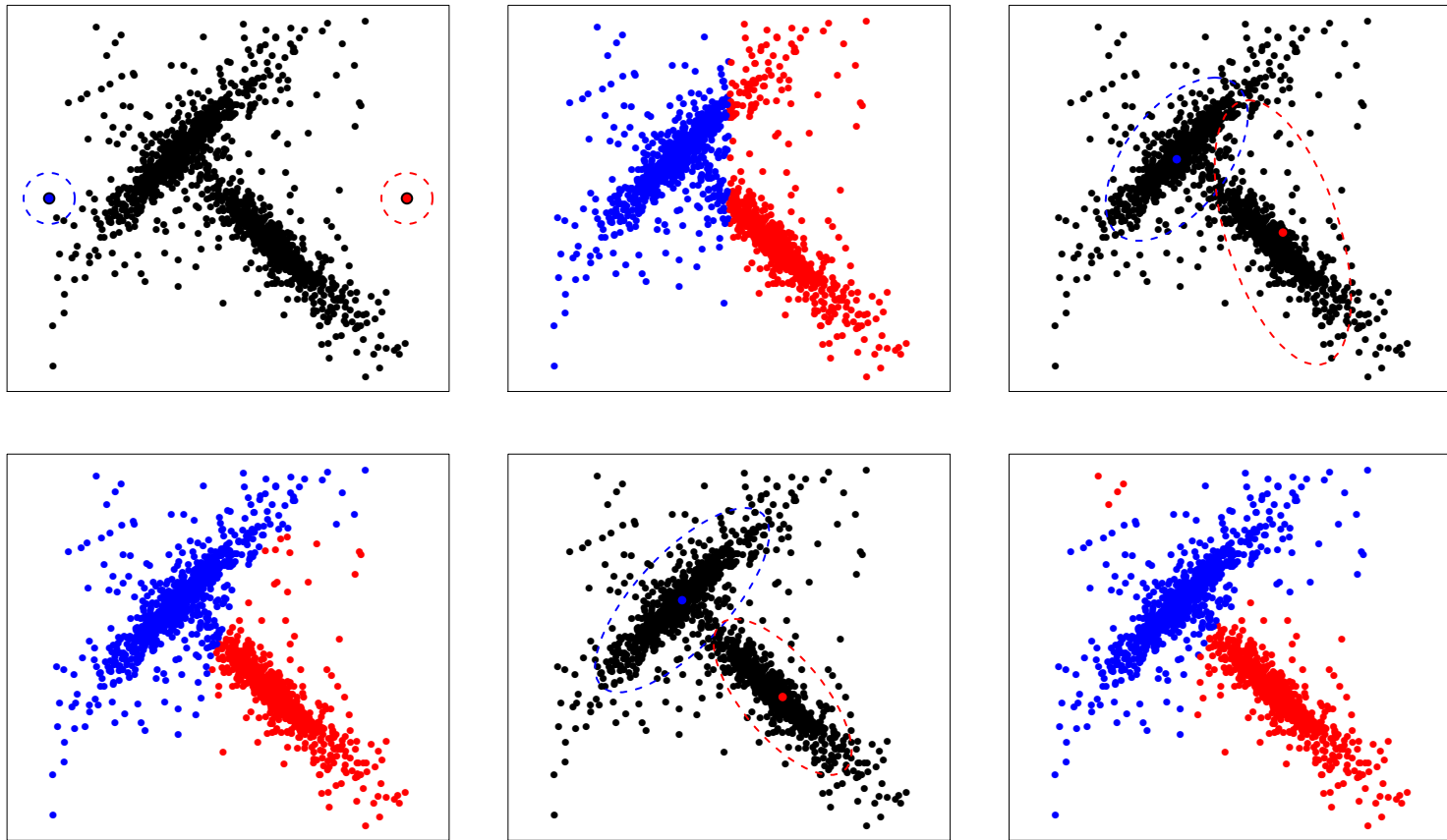
$$z_i = \operatorname{argmax}_{z \in \{1, \dots, k\}} p[\mathbf{x}_i | z_i = z, \Theta_z] .$$

**M2 step.** Find parameters  $\Theta$  that minimise  $F(z, \Theta)$  for fixed  $z$ . Again, parameters can separately be sought for each data source:

$$\Theta_j^* = \operatorname{argmax}_{\Theta_j} \sum_{i \in \mathcal{I}_j} \log p[\mathbf{x}_i | z_i = j, \Theta_j]$$

where  $\mathcal{I}_j = \{i : z_i = j\}$  denotes elements coming from the  $j$ th data source.

## Illustrative example



## Hard clustering is brittle against outliers



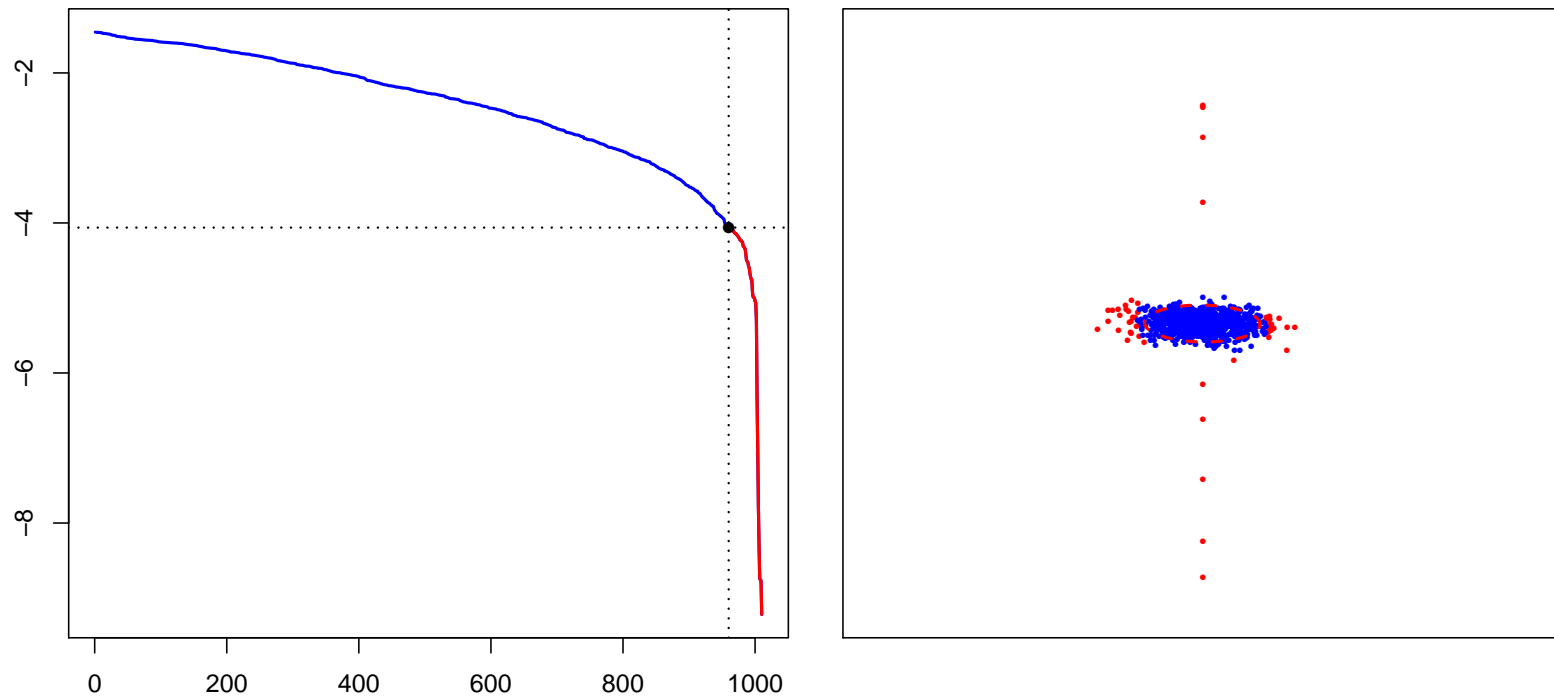
A few data points not belonging to the cluster can completely offset the shape estimation procedure. We need to reduce the impact of outliers.

## Quick fix

Assuming that current parameter estimates are not far off, we can make parameter fitting more robust by throwing out improbable points.

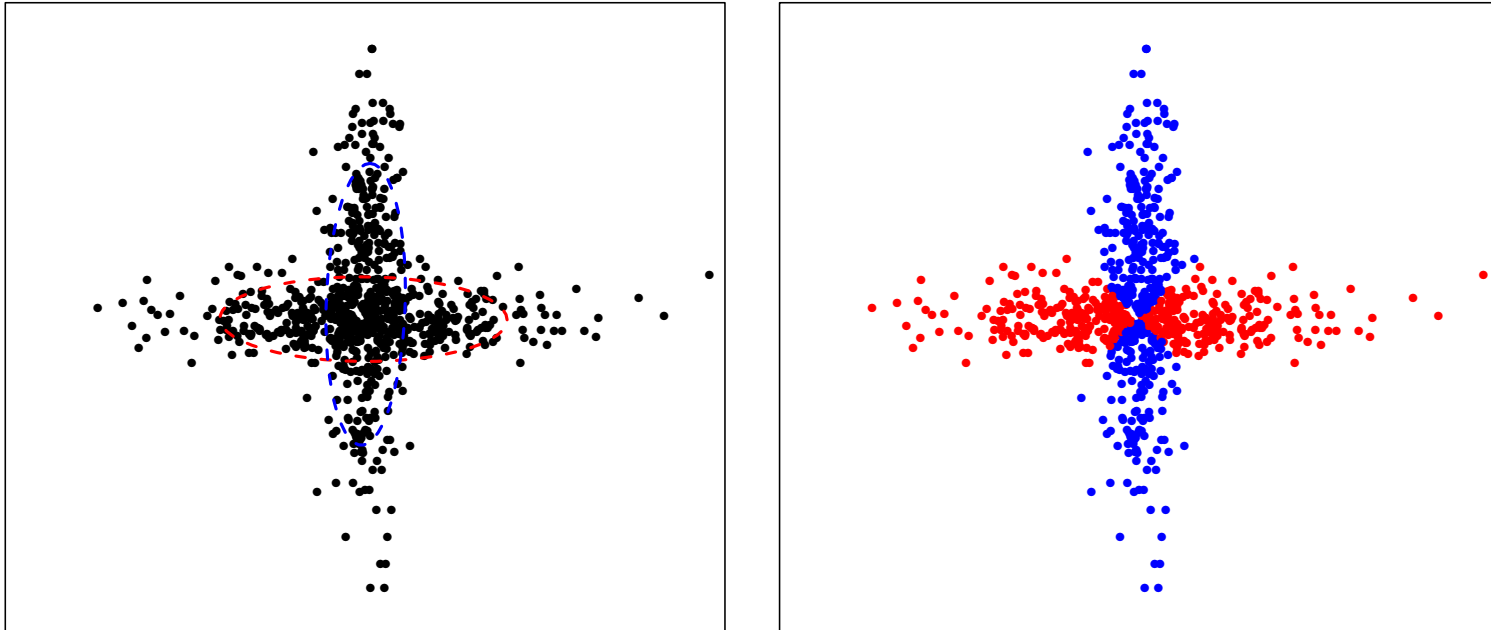
- ▷ Compute likelihood  $p[\mathbf{x}_i | \Theta_j]$  for each data point  $\mathbf{x}_i$  in the cluster
- ▷ Throw out 5-10% cluster points with lowest likelihood scores
- ▷ Model parameters based on the reduced dataset

## Illustrative example



Left pane shows ordered log likelihood of points and corresponding cut-off point. The right pane shows how the outlier elimination alter parameters.

## Choice of labels can be ambiguous



Sometimes the likelihoods for different sources are almost equal

- ▷ Label assignment is almost arbitrary
- ▷ Data points that belong to the cluster are not counted



## Fractional weights as an alternative

Data points should have different weights based on the plausibility

- ▷ Potential outliers should have low weights to limit their impact
- ▷ Ambiguous points should impact the parameters of both clusters
- ▷ We should not prefer some data points to others in the algorithm

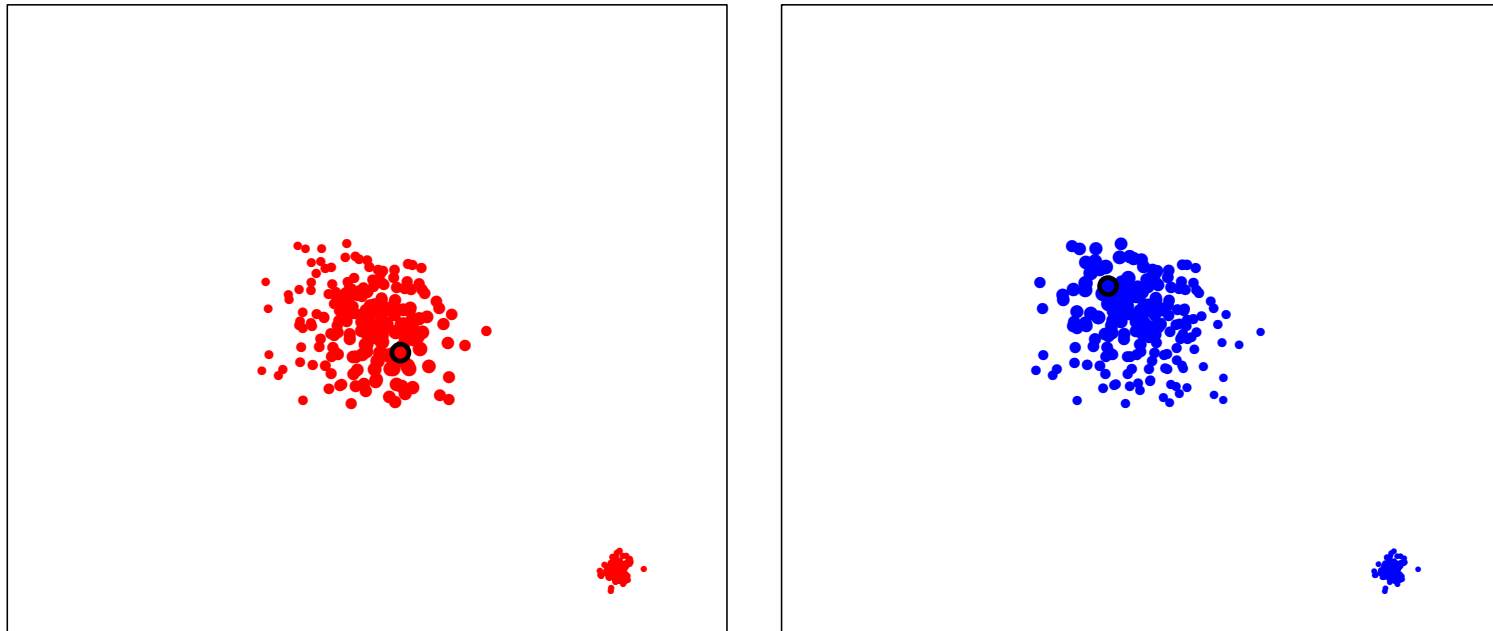
**First try.** The most obvious choice for the weights are likelihoods

$$w_{ij} = p[\mathbf{x}_i | z_i = j, \Theta_j]$$

but some data points have very low likelihoods for all clusters

- ▷ These data points would be largely ignored by the algorithm

## The weighting scheme hides data



The overall contribution of cluster points in south-east direction is small for both clusters and thus the cluster is never found by the the algorithm.

## Fractional weights as an alternative

Data points should have different weights based on the plausibility

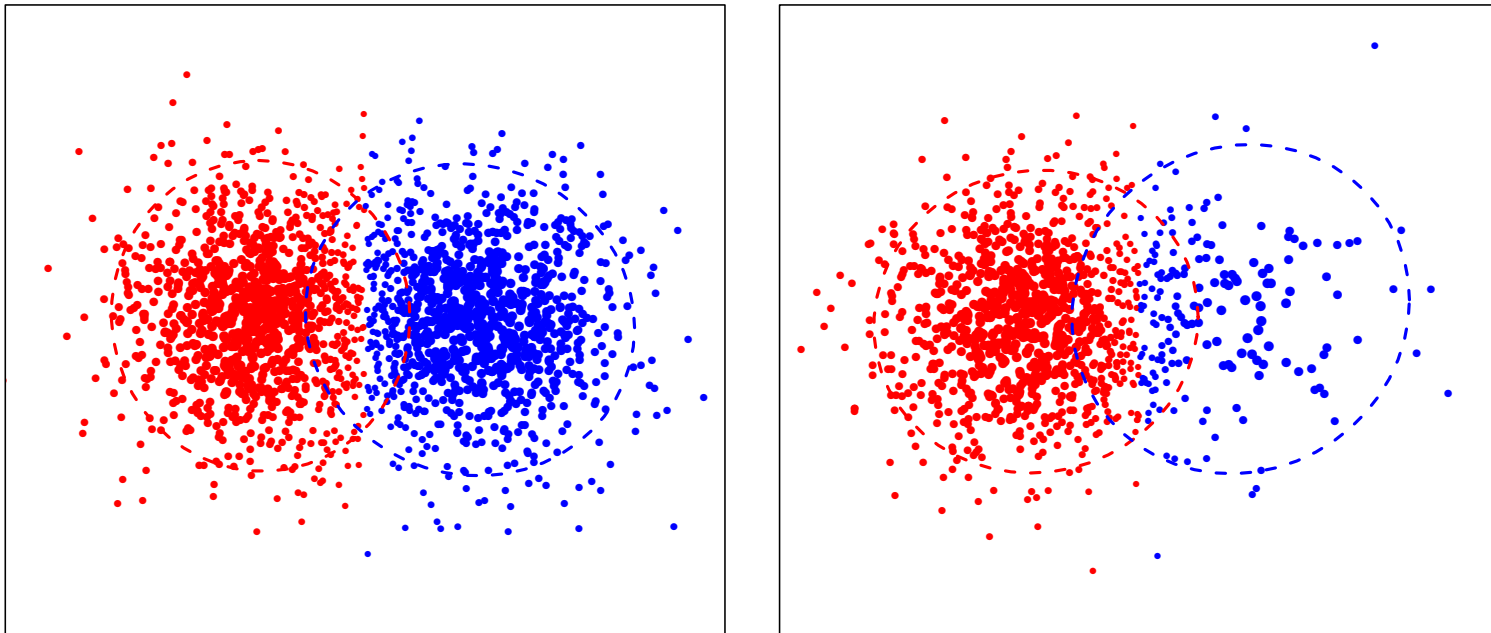
- ▷ Potential outliers should have low weights to limit their impact
- ▷ Ambiguous points should impact the parameters of both clusters
- ▷ We should not prefer some data points to others in the algorithm

**Second try.** We can normalise the weights so that the sum up to one

$$w_{ij} = \frac{p[\mathbf{x}_i | z_i = j, \Theta_j]}{p[\mathbf{x}_i | z_i = 1, \Theta_1] + \dots + p[\mathbf{x}_i | z_i = k, \Theta_k]}$$

This weighting does not work if getting data points from some cluster is much more probable than from the other clusters.

## The weighting scheme ignores frequency



If the elements from one data source are rare then the elements form the ambiguous area are more likely to originate from the abundant source.

## Fractional weights as an alternative

Data points should have different weights based on the plausibility

- ▷ Potential outliers should have low weights to limit their impact
- ▷ Ambiguous points should impact the parameters of both clusters
- ▷ We should not prefer some data points to others in the algorithm

**Standard solution.** We must take mixture proportions into account

$$\begin{aligned} w_{ij} &= \frac{\lambda_j \cdot p[\mathbf{x}_i | z_i = j, \Theta_j]}{\lambda_1 \cdot p[\mathbf{x}_i | z_i = 1, \Theta_1] + \dots + \lambda_k \cdot p[\mathbf{x}_i | z_i = k, \Theta_k]} \\ &= \frac{p[\mathbf{x}_i, z_i = j | \Theta]}{p[\mathbf{x}_i | \Theta]} = p[z_i = j | \mathbf{x}_i, \Theta] \end{aligned}$$

and thus the weight  $w_{ij}$  is equal to label probability given  $\mathbf{x}_i$  and  $\Theta$ .

How to handle fractional  
weights?

## Naive implementation of fractional weights

Under the independence assumption hard-clustering minimises

$$F(\mathbf{z}, \Theta) = - \sum_{i=1}^n \log p[\mathbf{x}_i | z_i, \Theta_{z_i}]$$

To implement weights with precision  $\frac{1}{\ell}$  we duplicate each data point  $\ell$  times and modify label assignment step **M1**. The step **M2** remains same.

### E1 step

- ▷ Find weights for each  $w_{ij} = \Pr[z_i = j | \mathbf{x}_i, \Theta]$
- ▷ Compute integer counts  $c_{ij} = \lfloor w_{ij} \cdot \ell \rfloor$  so that they add up to  $\ell$ .
- ▷ For each data point  $\mathbf{x}_i$  assign  $c_{ij}$  copies to the cluster  $j$ .

## Implementation without data duplication

**E1 step.** Compute fractional weights  $w_{ij} = p[z_i = j | \mathbf{x}_i, \Theta]$  and assign proportional number of point instances  $\hat{w}_{ij}$  to each cluster.

**M2 step.** Find parameters  $\Theta$  that minimise

$$F_*(\mathbf{w}, \Theta) = - \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p[\mathbf{x}_i | z_i = j, \Theta]$$

### Convergence

- ▷ M2 step clearly reduces the objective function.
- ▷ It is ~~not~~ evident that E1 step does not increase the objective function.
- ▷ We need another way to establish convergence.



## Update steps for Gaussian mixture model

**E1 step.** Compute fractional weights  $w_{ij} = p[z_i = j | \mathbf{x}_i, \Theta]$  for each point

$$w_{ij} = \frac{\lambda_j \cdot p[\mathbf{x}_i | z_i = j, \Theta_j]}{\lambda_1 \cdot p[\mathbf{x}_i | z_i = 1, \Theta_1] + \cdots + \lambda_k \cdot p[\mathbf{x}_i | z_i = k, \Theta_k]}$$

**M2 step.** Find parameters  $\Theta$  that minimise  $F_*(\mathbf{w}, \Theta)$  for fixed  $\mathbf{w}$  where  $\mathbf{w}$  denotes fractional multiplicity of labels:

$$n_j = \sum_{i=1}^n w_{ij}$$

$$\lambda_j = \frac{n_j}{n_1 + \cdots + n_k}$$

$$\mu_j = \frac{1}{n_j} \cdot \sum_{i=1}^n w_{ij} \mathbf{x}_i$$

$$\Sigma_j = \frac{1}{n_j} \cdot X^t \text{diag}(\mathbf{w}_{*j}) X$$

Why does this algorithm  
work?

## Hard-clustering is non-optimal in theory

Assume that we can compute all likelihoods  $p[\mathbf{x}_1, \mathbf{x}_2 | \mathbf{z}, \mathcal{M}_1]$  for two models

Model	$z_1$	$z_2$	p
$\mathcal{M}_1$	0	0	0.18
$\mathcal{M}_1$	0	1	0.17
$\mathcal{M}_1$	1	0	0.14
$\mathcal{M}_1$	1	1	0.15
			<b>0.64</b>

Model	$z_1$	$z_2$	p
$\mathcal{M}_2$	0	0	<b>0.24</b>
$\mathcal{M}_2$	0	1	0.04
$\mathcal{M}_2$	1	0	0.04
$\mathcal{M}_2$	1	1	0.04
			0.36

Then the hard clustering algorithm chooses the model  $\mathcal{M}_2$  although the overall plausibility of  $\mathcal{M}_1$  is much higher if we are neutral.

**Fix.** We should choose the model with the highest posterior probability

$$p[\Theta | \mathbf{x}_1, \dots, \mathbf{x}_n] = \sum_{\mathbf{z}} p[\Theta, \mathbf{z} | \mathbf{x}_1, \dots, \mathbf{x}_n]$$

## Further analysis

Now note that

$$p[\Theta, z | \mathbf{x}_1, \dots, \mathbf{x}_n] = \frac{p[\mathbf{x}_1, \dots, \mathbf{x}_n, z | \Theta] \cdot p[\Theta]}{p[\mathbf{x}_1, \dots, \mathbf{x}_n]}$$

and thus if we do *not have preferences over models* we get

$$p[\Theta, z | \mathbf{x}_1, \dots, \mathbf{x}_n] = c(\mathbf{x}_1, \dots, \mathbf{x}_n) \cdot p[\mathbf{x}_1, \dots, \mathbf{x}_n, z | \Theta]$$

Hence, we can solve the optimisation task

$$P(\Theta) = \sum_z p[\mathbf{x}_1, \dots, \mathbf{x}_n, z | \Theta] \rightarrow \max$$

## Decomposition into individual draws

The maximisation task is hard, since  $P(\Theta)$  has no nice form

$$\begin{aligned} P(\Theta) &= \sum_{\mathbf{z}} \prod_{i=1}^n \Pr[z_i | \Theta] \cdot p[\mathbf{x}_i | z_i, \Theta_{z_i}] \\ &= \prod_{i=1}^n \sum_{j=1}^k \Pr[z_i = j | \Theta] \cdot p[\mathbf{x}_i | z_i = j, \Theta_j] \end{aligned}$$

Even the logarithm trick does not help

$$\log P(\Theta) = \sum_{i=1}^n \log \left( \sum_{j=1}^k \Pr[z_i = j | \Theta] \cdot p[\mathbf{x}_i | z_i = j, \Theta_j] \right)$$

## Heuristic minimisation

Lets assign an arbitrary probability distribution over labels  $q(\mathbf{z})$  then we can formally provide a lower bound to the log-likelihood

$$\begin{aligned}\log p[\Theta | \mathbf{x}_1, \dots, \mathbf{x}_n] &= \log \left( \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \frac{p[\Theta, \mathbf{z} | \mathbf{x}_1, \dots, \mathbf{x}_n]}{q(\mathbf{z})} \right) \\ &\geq \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log \left( \frac{p[\Theta, \mathbf{z} | \mathbf{x}_1, \dots, \mathbf{x}_n]}{q(\mathbf{z})} \right) = F(\mathbf{q}, \Theta)\end{aligned}$$

Now if we manage to find  $q(\mathbf{z})$  and  $\Theta$  that maximise  $F(\mathbf{q}, \Theta)$  we have located the set of model parameters where the sought log-likelihood  $P(\Theta)$  must be quite high.

## Further analysis

The new minimisation target further decomposes into two separate parts

$$F(\mathbf{q}, \Theta) = - \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log q(\mathbf{z}) + \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log (p[\mathbf{z}, \Theta | \mathbf{x}_1, \dots, \mathbf{x}_n])$$

and we can use two-step minimisation algorithm

**E step.** Fix  $\Theta$  and find optimal weights  $q(\mathbf{z})$  for all labels

**M step.** Fix all weights  $q(\mathbf{z})$  and find parameters  $\Theta$  maximising  $F(\mathbf{q}, \Theta)$

## What are optimal weights?

It is well known fact that the term

$$\sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log \left( \frac{p[\mathbf{z}, \Theta | \mathbf{x}_1, \dots, \mathbf{x}_n]}{q(\mathbf{z})} \right)$$

is maximised only if

$$q(\mathbf{z}) = c_* \cdot p[\mathbf{z}, \Theta | \mathbf{x}_1, \dots, \mathbf{x}_n] = p[\mathbf{z} | \Theta, \mathbf{x}_1, \dots, \mathbf{x}_n]$$

Since the probabilities of labels decompose into the product so does  $q(\mathbf{z})$

$$q(\mathbf{z}) = \prod_{i=1}^n p[z_i | \Theta_{z_i}, \mathbf{x}_i] = \prod_{i=1}^n w_{iz_i}$$

where the weights are the same as were in our *practical algorithm*.



## What are optimal parameters?

For fixed weights  $q(\mathbf{z})$  the first term in  $F(\mathbf{q}, \Theta)$  is constant and thus optimal set of parameters can be found by maximising

$$\sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log (p[\mathbf{z}, \Theta | \mathbf{x}_1, \dots, \mathbf{x}_n])$$

The latter is equivalent to the following maximisation task

$$\sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log (p[\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z} | \Theta])$$

which can be further decomposed into a simpler sum

$$\sum_{i=1}^n \sum_{\mathbf{z} \in \mathcal{Z}} q(\mathbf{z}) \cdot \log (p[\mathbf{x}_i, \mathbf{z}_i | \Theta]) = \sum_{i=1}^n \sum_{\mathbf{z} \in \mathcal{Z}} \prod_{\ell=1}^n w_{\ell \mathbf{z}_{\ell}} \cdot \log (p[\mathbf{x}_i, \mathbf{z}_i | \Theta]) \quad .$$

## Lower-bound regularly coincides with probability

Observe the second factor in the sum after E-step to get further insight

$$\begin{aligned} F(\mathbf{q}, \Theta) &= \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log \left( \frac{p[\Theta, \mathbf{z} | \mathbf{x}_1, \dots, \mathbf{x}_n]}{q(\mathbf{z})} \right) \\ &= \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log \left( \frac{p[\Theta, \mathbf{z} | \mathbf{x}_1, \dots, \mathbf{x}_n]}{p[\mathbf{z} | \Theta, \mathbf{x}_1, \dots, \mathbf{x}_n]} \right) \\ &= \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log \left( \frac{p[\Theta, \mathbf{z} | \mathbf{x}_1, \dots, \mathbf{x}_n]}{p[\mathbf{z} | \Theta, \mathbf{x}_1, \dots, \mathbf{x}_n]} \right) \\ &= \sum_{\mathbf{z}} q(\mathbf{z}) \cdot \log (p[\Theta | \mathbf{x}_1, \dots, \mathbf{x}_n]) \\ &= \log (p[\Theta | \mathbf{x}_1, \dots, \mathbf{x}_n]) \end{aligned}$$

## Connection with the hard-clustering algorithm

The hard-clustering algorithm assigns zero-one probabilities to different labelings  $\mathbf{z}$  which can be formalised as products of zero-one weights  $w_{ij}$ . Thus the parameters are still chosen by maximising

$$\sum_{i=1}^n \sum_{\mathbf{z} \in \mathcal{Z}} \prod_{\ell=1}^n w_{\ell z_{\ell}} \cdot \log (p[\mathbf{x}_i, \mathbf{z}_i | \Theta]) \quad .$$

Soft-clustering (EM-algorithm) is just a more general maximisation task.

Maximisation task can be converted back to hard-clustering problem by rounding fractional weights to integers.

- ▷ This leads to the dataset with repeated data samples.
- ▷ We can still use the update steps for hard clustering.