EM Algorithm and Classification

Amitangshu Dasgupta (161082)

MTH 511A

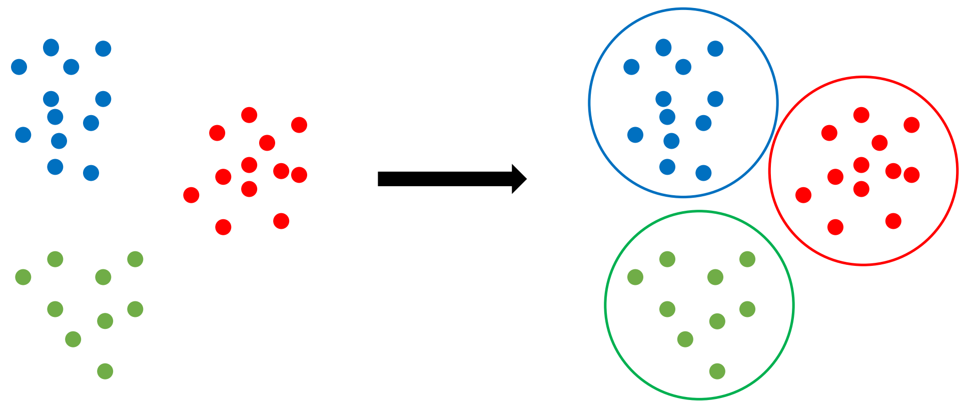
Indian Institute of Technology, Kanpur

Contents

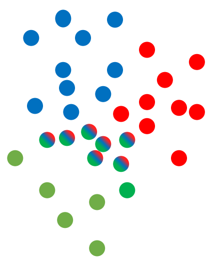
<u>Section A: Introduction and Problem Discussion</u>

EM Algorithm for Gaussian Mixture Model (Model Based Clustering)

Why use Gaussian Mixture Models? The task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.is known as clustering. In the process of clustering, we assign each data point to a cluster where the number of clusters to be formed is predetermined. The following illustration shows a collection of bivariate data points are grouped into 3 clusters using the k-means algorithm. It is to be noted that the shape and diameter of each of the circles denoting the clusters in the grouped data (right hand image) are the same. This is because the k-means algorithm uses the notion of Euclidean distance from the center of each cluster.



Most clustering algorithms (such as the k-means algorithm above) use hard assignments, which means that it is assumed that each data point can belong to, i.e. can have the properties of exactly one cluster. But this is not always true. For example the visualization of the dataset below shows a few data points that can be clustered into either of the two or three groups. In such cases, hard assignments are meaningless and thus a probabilistic approach is needed where we try to compute a probability for each data point to belong to each cluster. It is to be noted that we do not have any initial labels of the data points and the color is provided in the image for illustrative purposes.



Obtaining Maximum Likelihood Estimates of Parameters (Univariate)

We assume that the data points follow a mixture of k Gaussian distributions, where k is the number of clusters to be formed. Say $k = 2$. Let our data set be $\{x_1, x_2, \dots x_n\}$.

$$f(x) = \pi f_1(x; \mu_1, \sigma_1^2) + (1-\pi) f_2(x; \mu_2, \sigma_2^2)$$

where $f_w(x, \mu_w, \sigma_w^2)$ is the normal(μ_1, σ_1^2) pdf at x, $w = 1{,}2$.

Goal: To estimate $\pi, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2$.

Log-likelihood function

$$L(\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2) = \prod_{i=1}^{n} \pi f_1(x_i; \mu_1, \sigma_1^2) + (1-\pi) f_2(x_i; \mu_2, \sigma_2^2)$$

We are to maximize L w.r.t the unknown parameters which would be solving a system of 5 non-linear equations to obtain the MLE's.

We can alternatively use the EM Algorithm which treats the problem as a missing value problem. We have two populations and the information that x_i is coming from which population.

Let's consider the complete set of observations $\{(x_1, \delta_1), (x_2, \delta_2), \dots (x_n, \delta_n)\}$ to come from (X, Δ) where $\delta_i = \begin{cases} 1 \text{ if } x_i \text{ comes from population 1.} \\ 0 \text{ if } x_i \text{ comes from population 2.} \end{cases}$

Note that,

$P(\Delta = 1) = \pi$

$P(\Delta = 0) = 1 - \pi$

And

$f(x|\Delta = 1) = f_1(x, \mu_1, \sigma_1^2)$

$f(x|\Delta = 0) = f_2(x, \mu_2, \sigma_2^2)$.

Then the likelihood function would be

$$L(\pi, \mu_1, \sigma_1^2 \mu_2, \sigma_2^2) = \prod_{i:\delta_i=1} \pi f_1(x_i; \mu_1, \sigma_1^2) \times \prod_{i:\delta_i=0} (1-\pi) f_2(x_i; \mu_2, \sigma_2^2)$$

$$= \prod_{i=1}^{n} \{\pi f_1(x_i; \mu_1, \sigma_1^2)\}^{\delta_i} \times \{(1-\pi) f_2(x_i; \mu_2, \sigma_2^2)\}^{1-\delta_i}$$

$$l = \ln(L) = \sum_{i=1}^{n} \delta_i \ln\left(\pi f_1(x_i; \mu_1, \sigma_1^2)\right) + (1-\delta_i) \ln\left([1-\pi] f_2(x_i; \mu_2, \sigma_2^2)\right)$$

$$= \ln(\pi) \sum_{i=1}^{n} \delta_i + \sum_{i=1}^{n} \delta_i \ln(f_1(x_i; \mu_1, \sigma_1^2)) + \ln(1-\pi) \sum_{i=1}^{n} (1-\delta_i) + \sum_{i=1}^{n} (1-\delta_i) \ln(f_2(x_i; \mu_2, \sigma_2^2))$$

Maximizing the above w.r.t parameters gives the MLE's

$$\hat{\pi} = \frac{1}{n}\sum_{i=1}^{n} \delta_i$$

$$\hat{\mu}_1 = \frac{1}{\sum \delta_i}\sum_{i=1}^{n} \delta_i x_i$$

$$\hat{\sigma}_1^2 = \frac{1}{\sum \delta_i}\sum_{i=1}^{n} \delta_i (x_i - \hat{\mu}_1)^2$$

$$\hat{\mu}_2 = \frac{1}{\sum(1-\delta_i)}\sum_{i=1}^{n} (1-\delta_i) x_i$$

$$\hat{\sigma}_2^2 = \frac{1}{\sum(1-\delta_i)}\sum_{i=1}^{n} (1-\delta_i)(x_i - \hat{\mu}_2)^2$$

Now, the assignment vector $\{\delta_1, \delta_2, \dots \delta_n\}$ is missing and thus the EM Algorithm comes into play. We replace the assignment δ_i by the responsibility $r_i = \mathrm{E}(\delta_i | X, \theta)$ where $\theta = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$

It can be shown that,

$$r_i = \frac{\pi f_1(x_i; \mu_1, \sigma_1^2)}{\pi f_1(x_i; \mu_1, \sigma_1^2) + (1-\pi) f_2(x_i; \mu_2, \sigma_2^2)}$$

In naïve terms, r_i represents the likelihood of the data point x_i to belong to population 1, while $1 - r_i$ naturally represents the likelihood of x_i to belong to population 2.

Replacing δ_i with their expected values, we obtain the pseudo log-likelihood function

$$\dot{l} = \ln(\pi)\sum_{i=1}^{n} r_i + \sum_{i=1}^{n} \delta_i \ln(f_1(x_i; \mu_1, \sigma_1^2)) + \ln(1-\pi)\sum_{i=1}^{n}(1-r_i) + \sum_{i=1}^{n}(1-\delta_i)\ln(f_2(x_i; \mu_2, \sigma_2^2))$$

The EM algorithm has two iterative steps.

E-step: Estimating the responsibility vector using the estimates of the parameters viz. $\hat{\pi}, \hat{\mu}_1, \hat{\sigma}_1^2 \hat{\mu}_2, \hat{\sigma}_2^2$.

M-step: Maximizing l w.r.t the parameters using the responsibility vector and obtain the parameter maximum likelihood estimates for the next iteration.

Let $(\pi^{(k)}, \mu^{(k)}{}_1, \sigma_1^{2\,(k)}, \mu^{(k)}{}_2, \sigma_2^{2\,(k)})$ be the estimates of the respective parameters at the kth iteration. Then, we have the recursive relation:

$$\pi^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n} r_i{}^{(k)}$$

$$\mu^{(k+1)}{}_1 = \frac{1}{\sum r_i{}^{(k)}}\sum_{i=1}^{n} r_i{}^{(k)} x_i$$

$$\sigma_1^{2\,(k+1)} = \frac{1}{\sum r_i{}^{(k)}}\sum_{i=1}^{n} r_i{}^{(k)} (x_i - \mu^{(k+1)}{}_1)^2$$

$$\mu^{(k+1)}{}_2 = \frac{1}{\sum (1 - r_i{}^{(k)})}\sum_{i=1}^{n} (1 - r_i{}^{(k)}) x_i$$

$$\sigma_2^{2\,(k+1)} = \frac{1}{\sum (1 - r_i{}^{(k)})}\sum_{i=1}^{n} (1 - r_i{}^{(k)})(x_i - \mu^{(k+1)}{}_2)^2$$

We may start with any random guess for the parameter estimates, but they will ultimately converge to the actual estimates.

Classification based on Maximum Likelihood Estimates Obtained

After we have obtained the MLE's of the parameters, we have essentially obtained the individual Gaussian distributions. Now we may classify an observation into that distribution (cluster) that possesses more similarity. In other words we classify observations accordingly by their likelihood to lie in a particular distribution.

In order to do so, we compute the responsibility

$$r(x) = \frac{\pi f_1(x; \mu_1, \sigma_1^2)}{\pi f_1(x; \mu_1, \sigma_1^2) + (1 - \pi) f_2(x; \mu_2, \sigma_2^2)}$$

for an observation x and assign it to population 1 if $r(x)$ is more than 0.5 or to population 2 if $r(x)$ is less than 0.5. Note that $r(x)$ is essentially the likelihood of the observation x to belong to population 1 given the parameter estimates.

Problem Discussion

The problem we have at hand:

We have a data set of measurements of original and fake currencies. We know for each note whether it belongs to the population of fake currencies or the population of real currencies. But, any hard assignment algorithm based on this knowledge would be meaningless since firstly the populations (clusters) clearly overlap, as we will see in Sec C, and secondly a hard assignment algorithm would randomly assign observations that lie in the intersection of multiple clusters. Hence, an EM algorithm would be an appropriate classification method to use in this problem.

Section B: EM Algorithm for Multivariate Gaussian Mixture Model

Consider the p-variate data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_n\}$.

$$f(\boldsymbol{x}) = \pi f_1(\boldsymbol{x}; \boldsymbol{\theta}_1) + (1-\pi) f_2(\boldsymbol{x}; \boldsymbol{\theta}_2)$$

Goal: To estimate $\pi, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2$.

We assume

$$f_w(\boldsymbol{x}; \boldsymbol{\theta}_w) \equiv \frac{1}{(2\pi)^{p/2} |\Sigma_w|^{1/2}} e^{-\frac{1}{2}(x-\boldsymbol{\mu}_w)' \Sigma^{-1} (x-\boldsymbol{\mu}_w)}, \qquad w = 1,2;\ \boldsymbol{\theta}_w = (\boldsymbol{\mu}_w, \Sigma_w)$$

where

$$\boldsymbol{\mu}_w = (\mu_1, \mu_2 \dots \mu_p)'$$

$$\Sigma_w = \begin{matrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{21} \\ \vdots & \vdots & & \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{matrix}$$

Likelihood function,

$$L(\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \prod_{i=1}^{n} \pi f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1) + (1-\pi) f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \Sigma_2)$$

Thus, to obtain the MLE's we need to solve a system of equations with $2p + p(p+1) + 1$ unknowns.

Let the complete data be $\{(\boldsymbol{x}_1, \delta_1), (\boldsymbol{x}_2, \delta_2), \dots (\boldsymbol{x}_n, \delta_n)\}$. Proceeding as before, we obtain the likelihood and the log-likelihood function,

$$L(\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \prod_{i:\delta_i=1} \pi f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1) \times \prod_{i:\delta_i=0} (1-\pi) f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \Sigma_2)$$

$$= \prod_{i=1}^{n} \{\pi f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1)\}^{\delta_i} \times \{(1-\pi) f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \Sigma_2)\}^{1-\delta_i}$$

$$l = \ln(L) = \sum_{i=1}^{n} \delta_i \ln\big(\pi f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1)\big) + (1-\delta_i) \ln\big([1-\pi] f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_2)\big)$$

$$= \ln(\pi) \sum_{i=1}^{n} \delta_i + \sum_{i=1}^{n} \delta_i \ln(f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1)) + \ln(1-\pi) \sum_{i=1}^{n} (1-\delta_i) + \sum_{i=1}^{n} (1-\delta_i) \ln(f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \Sigma_2))$$

Maximizing the above w.r.t parameters gives the MLE's

$$\hat{\pi} = \frac{1}{n}\sum_{i=1}^{n} \delta_i$$

$$\hat{\boldsymbol{\mu}}_1 = \frac{1}{\sum_i \delta_i}\sum_{i=1}^{n} \delta_i \boldsymbol{x}_i$$

$$\hat{\Sigma}_1 = \frac{1}{\sum_i \delta_i}\sum_{i=1}^{n} \delta_i (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_1)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_1)'$$

$$\hat{\boldsymbol{\mu}}_2 = \frac{1}{\sum_i (1-\delta_i)}\sum_{i=1}^{n} (1-\delta_i) \boldsymbol{x}_i$$

$$\hat{\Sigma}_2 = \frac{1}{\sum_i (1-\delta_i)}\sum_{i=1}^{n} (1-\delta_i) (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_2)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_2)'$$

Just as in the univariate case, we replace the unknown δ_i's with their expectations.

$$\mathrm{E}(\delta_i | \boldsymbol{X}, \boldsymbol{\theta}) = r_i = \frac{\pi f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1)}{\pi f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1) + (1-\pi) f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \Sigma_2)} \quad (*)$$

Pseudo log-likelihood function,

$$\dot{l} = \ln(\pi)\sum_{i=1}^{n} r_i + \sum_{i=1}^{n} r_i \ln(f_1(\boldsymbol{x}_i; \boldsymbol{\mu}_1, \Sigma_1)) + \ln(1-\pi)\sum_{i=1}^{n}(1-r_i) + \sum_{i=1}^{n}(1-r_i)\ln(f_2(\boldsymbol{x}_i; \boldsymbol{\mu}_2, \Sigma_2))$$

which gives the MLE's for the $(k+1)^{\text{th}}$ iteration of the algorithm:

$$\left.\begin{aligned}
\pi^{(k+1)} &= \frac{1}{n}\sum_{i=1}^{n} r_i^{(k)} \\
\boldsymbol{\mu}^{(k+1)}{}_1 &= \frac{1}{\sum r_i^{(k)}}\sum_{i=1}^{n} r_i^{(k)} \boldsymbol{x}_i \\
\Sigma_1{}^{(k+1)} &= \frac{1}{\sum r_i^{(k)}}\sum_{i=1}^{n} r_i^{(k)} (\boldsymbol{x}_i - \boldsymbol{\mu}^{(k+1)}{}_1)(\boldsymbol{x}_i - \boldsymbol{\mu}^{(k+1)}{}_1)' \\
\boldsymbol{\mu}^{(k+1)}{}_2 &= \frac{1}{\sum (1-r_i^{(k)})}\sum_{i=1}^{n} (1-r_i^{(k)}) \boldsymbol{x}_i \\
\Sigma_2{}^{(k+1)} &= \frac{1}{\sum (1-r_i^{(k)})}\sum_{i=1}^{n} (1-r_i^{(k)}) (\boldsymbol{x}_i - \boldsymbol{\mu}^{(k+1)}{}_2)(\boldsymbol{x}_i - \boldsymbol{\mu}^{(k+1)}{}_2)'
\end{aligned}\right\} \quad (\#)$$

Algorithm steps:

Step 1: Initiate $(\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = (\pi^{(0)}, \boldsymbol{\mu}^{(0)}{}_1, \boldsymbol{\mu}^{(0)}{}_2, \Sigma_1{}^{(0)}, \Sigma_2{}^{(0)})$ by guess.

Step 2: E-step – Compute $r_1{}^{(0)}, r_2{}^{(0)} \dots r_n{}^{(0)}$ using $(\pi^{(0)}, \boldsymbol{\mu}^{(0)}{}_1, \boldsymbol{\mu}^{(0)}{}_2, \Sigma_1{}^{(0)}, \Sigma_2{}^{(0)})$ and the relation $(*)$.

Step 3: M-step – Compute l and the MLE's using (#).

Continue repeating Steps 2 and 3 until l stabilizes at some value.

Section C: Data Analysis, Methodology and Numerical Results

We obtain the summary statistics and covariance matrices for the two groups:

Group 0 (Fake currencies)

Summary statistics

```
    length          left           right          bottom          top           diagonal
 Min.   :213.8   Min.   :129.0   Min.   :129.0   Min.   : 7.200   Min.   : 7.700   Min.   :139.6
 1st Qu.:214.7   1st Qu.:129.7   1st Qu.:129.4   1st Qu.: 7.900   1st Qu.: 9.775   1st Qu.:141.2
 Median :215.0   Median :129.9   Median :129.7   Median : 8.250   Median :10.200   Median :141.5
 Mean   :215.0   Mean   :129.9   Mean   :129.7   Mean   : 8.305   Mean   :10.168   Mean   :141.5
 3rd Qu.:215.2   3rd Qu.:130.2   3rd Qu.:130.0   3rd Qu.: 8.800   3rd Qu.:10.600   3rd Qu.:141.8
 Max.   :215.9   Max.   :131.0   Max.   :131.1   Max.   :10.400   Max.   :11.700   Max.   :142.4
```

Covariance matrix

	length	left	right	bottom	top	diagonal
length	0.150	0.058	0.057	0.057	0.014	0.005
left	0.058	0.133	0.086	0.057	0.049	-0.043
right	0.057	0.086	0.126	0.058	0.031	-0.024
bottom	0.057	0.057	0.058	0.413	-0.263	0.000
top	0.014	0.049	0.031	-0.263	0.421	-0.075
diagonal	0.005	-0.043	-0.024	0.000	-0.075	0.200

Group 1 (Real currencies)

Summary statistics

```
    length          left           right          bottom         top           diagonal
 Min.   :213.9   Min.   :129.6   Min.   :129.3   Min.   : 7.40   Min.   : 9.10   Min.   :137.8
 1st Qu.:214.6   1st Qu.:130.1   1st Qu.:130.0   1st Qu.: 9.90   1st Qu.:10.68   1st Qu.:139.2
 Median :214.8   Median :130.3   Median :130.2   Median :10.60   Median :11.10   Median :139.5
 Mean   :214.8   Mean   :130.3   Mean   :130.2   Mean   :10.53   Mean   :11.13   Mean   :139.4
 3rd Qu.:215.0   3rd Qu.:130.5   3rd Qu.:130.4   3rd Qu.:11.40   3rd Qu.:11.53   3rd Qu.:139.8
 Max.   :216.3   Max.   :130.8   Max.   :131.1   Max.   :12.70   Max.   :12.30   Max.   :140.6
```

Covariance matrix

	length	left	right	bottom	top	diagonal
length	0.124	0.032	0.024	-0.101	0.019	0.012
left	0.032	0.065	0.047	-0.024	-0.012	-0.005
right	0.024	0.047	0.089	-0.019	0.000	0.034
bottom	-0.101	-0.024	-0.019	1.281	-0.490	0.238
top	0.019	-0.012	0.000	-0.490	0.404	-0.022
diagonal	0.012	-0.005	0.034	0.238	-0.022	0.311

We shall be using these to determine which group is the population of fake currencies and which group is that of real currencies.

Methodology:

Stage 1: Partitioning

a. We partition the data randomly into two groups viz. train.data (85% of observations) and test.data (remaining 15% of observations) ignoring the assignments (column1)
b. We further partition train.data randomly into two groups viz. population1 (50%) and poplation2 (50%)

Stage 2: Computation of initial parameter values

We will be using the mean vector and covariance matrix obtained from these two groups to initiate parameter values in Step 1 of the EM algorithm. We shall initiate $\pi = 0.5$

Stage 3: Computation of responsibility vector

We compute the responsibility vector $\boldsymbol{r}$ using the parameter values obtained in Stage 2 and the relation $(*)$

Stage 4: Computing the pseudo log-likelihood function

We obtain the value of the pseudo log-likelihood $\dot{l}$ using the responsibility vector $\boldsymbol{r}$ from Stage 3 and the parameter values from Stage 2.

Stage 5: Updating parameter values

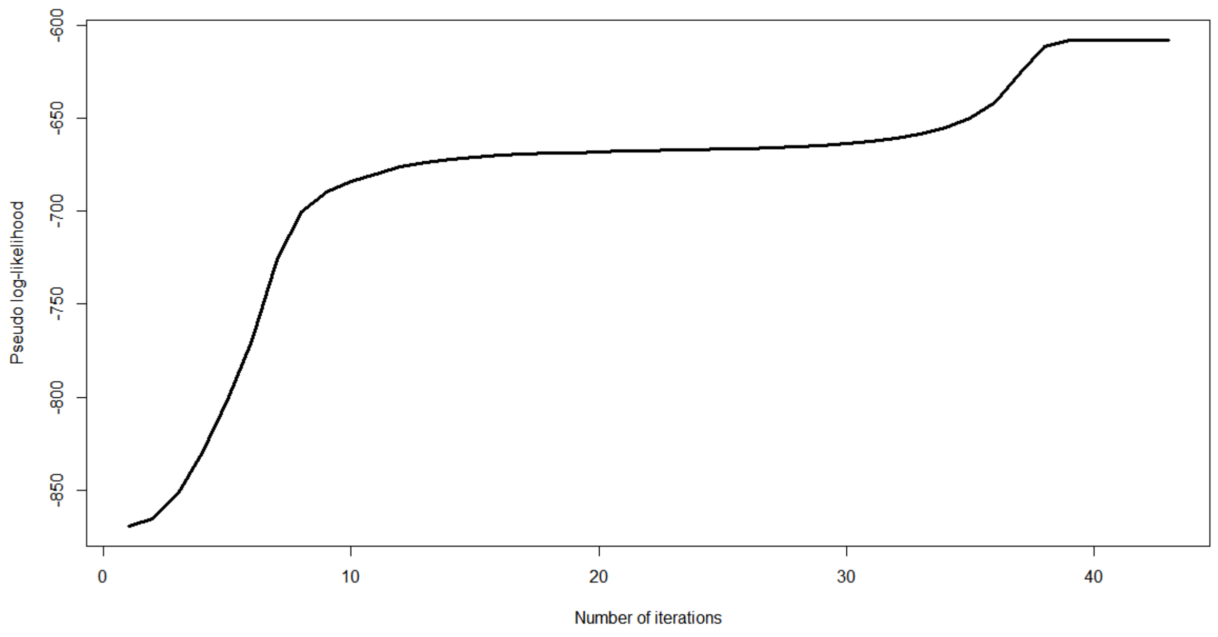
We update the parameter values using the responsibility vector $\boldsymbol{r}$ and the relations (#)

Note that we use the partitions population1 and population2 from Stage 1 only for the computation of the initial parameter values. We perform Stage 3-5 iteratively until the value of the pseudo log-likelihood $\dot{l}$ stabilizes on the whole of train.data

Numerical results

We took the stopping rule to stop at the k^{th} iteration if $\left|\dot{l}^{(k+1)} - \dot{l}^{(k)}\right| < 0.01$

Number of iterations needed: 43



Population 1: Gaussian($\boldsymbol{\mu}_1, \Sigma_1$)

Population 2: Gaussian($\boldsymbol{\mu}_2, \Sigma_2$)

$\boldsymbol{\mu}_1 =$ $(214.783, 130.284, 130.186, 10.891, 11.086, 139.628)$

$\Sigma_1 =$

0.084	0.035	0.036	-0.060	0.044	0.046
0.035	0.060	0.049	0.021	-0.004	0.037
0.036	0.049	0.097	-0.018	0.016	0.060
-0.060	0.021	-0.018	0.716	-0.437	-0.104
0.044	-0.004	0.016	-0.437	0.397	0.037
0.046	0.037	0.060	-0.104	0.037	0.127

$\boldsymbol{\mu}_2 =$ $(214.962, 130.005, 129.803, 8.385, 10.293, 141.131)$

$\Sigma_2 =$

0.179	0.059	0.061	0.071	0.002	-0.011
0.059	0.142	0.106	0.103	0.106	-0.232
0.061	0.106	0.143	0.097	0.097	-0.211
0.071	0.103	0.097	0.484	-0.212	-0.176
0.002	0.106	0.097	-0.212	0.558	-0.469
-0.011	-0.232	-0.211	-0.176	-0.469	1.288

Section D: Classification

Methodology

After we have obtained the separate distributions (i.e MLE's of the parameters) by the EM Algorithm, we can proceed with the classification of our test set, test.data

For each observation x in test.data, we first obtain the responsibility metric $r(x)$ (ignoring the actual assignments $a(x)$, say) by using the relation

$$r(x) = \frac{\pi f_1(x; \boldsymbol{\mu}_1, \Sigma_1)}{\pi f_1(x; \boldsymbol{\mu}_1, \Sigma_1) + (1-\pi) f_2(x; \boldsymbol{\mu}_2, \Sigma_2)}$$

We assign x to Population 1 if $r(x)$ is less than 0.5 or to Population 2 otherwise.

Numerical Results

We apply the classification to our test.data which contained 14 fake and 16 real currencies. We obtain the following result:

Observation x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Actual Assignments a(x)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	$\boxed{1}$	1	1	1	1	1	$\boxed{1}$	1	1	1	1
Responsiility r(x)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	$\boxed{0}$	1	1	1	1	1	$\boxed{0}$	1	1	1	1

We notice that there are 2 misclassifications out of the 30 observations.

Note that, we don't have the knowledge about which of the two population is that of the fake currencies and which one is that of real ones. Therefore, we might get complementary results while classification, i.e in this case we might have got 28 misclassifications. This just means that we are associating the wrong distribution to a given population. In such a case, reverting the association (i.e assigning x to Population 1 if $r(x)$ is more than 0.5 and to Population 2 otherwise) would solve the problem.

Section E: Conclusion

We used the EM Algorithm for different stopping rules, i.e

Rule: Stop at the k^{th} iteration if $\left|\hat{l}^{(k+1)} - \hat{l}^{(k)}\right| < \varepsilon$

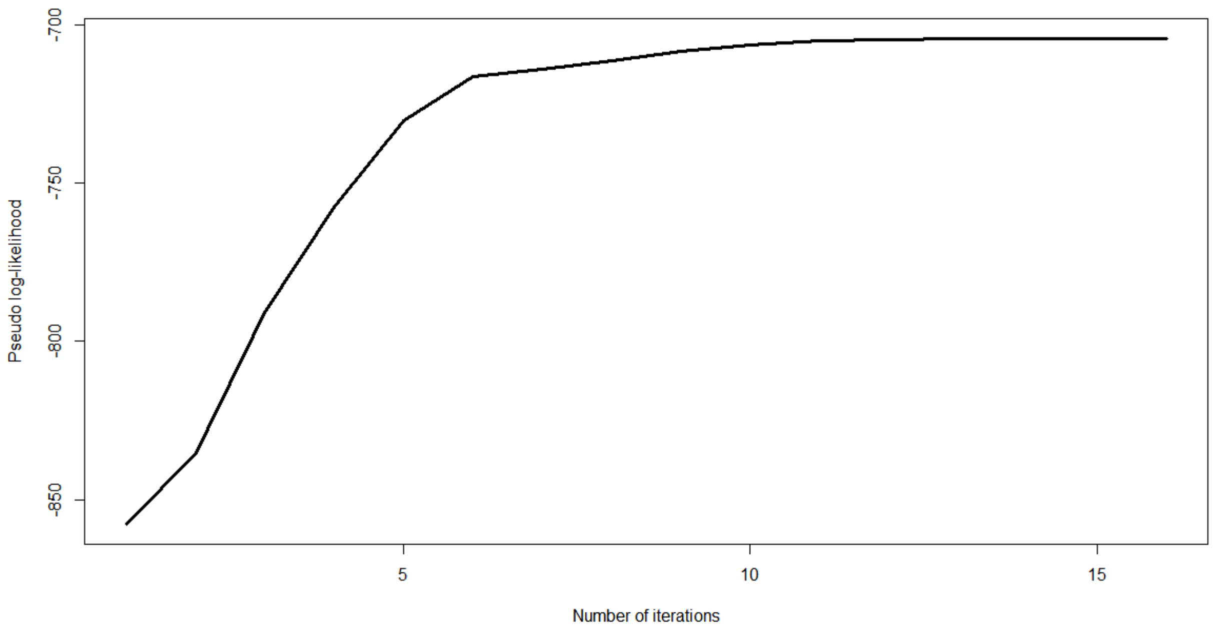
for $\varepsilon \in \{0.01, 0.03, 0.1, 0.3\}$

We provided the complete numerical results for the case $\varepsilon = 0.01$ above.

We shall compare the end results for the other values of ε below.

Case: $\varepsilon = 0.03$

Number of iterations: 29

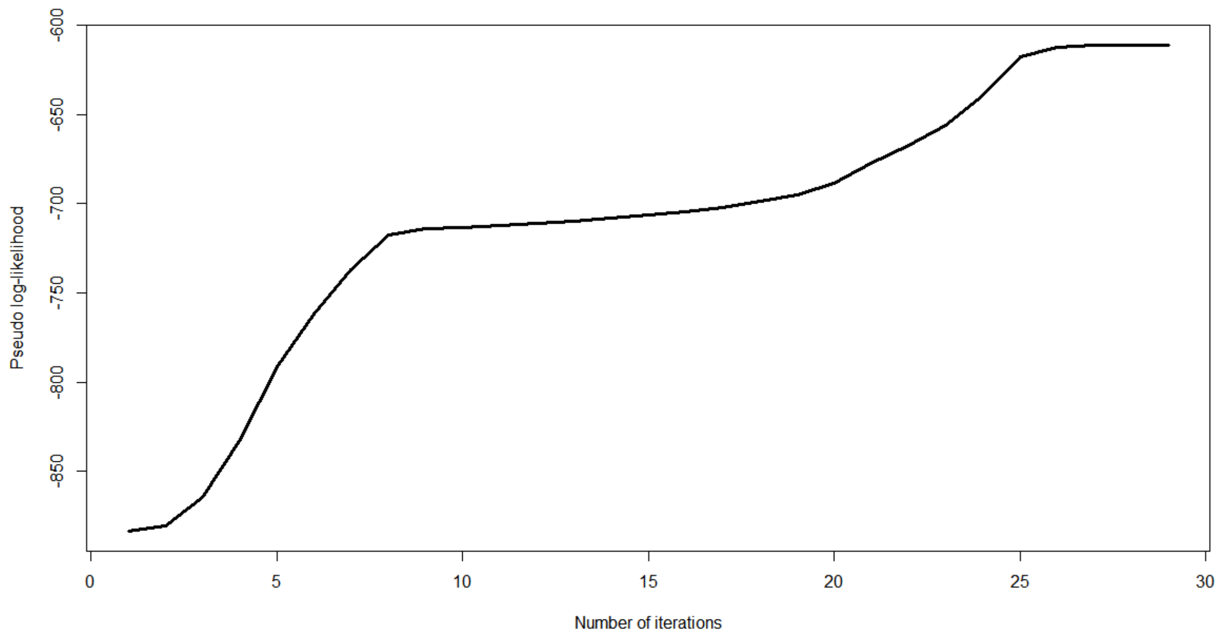


Observation x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Actual Assignments a(x)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	$\boxed{1}$	1	1	1	1	1	1	1	$\boxed{1}$	1	1	1	1	$\boxed{1}$
Responsiility r(x)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	$\boxed{0}$	1	1	1	1	1	1	1	$\boxed{0}$	1	1	1	1	$\boxed{0}$

Number of misclassifications: 3

Case: $\varepsilon = 0.10$

Number of iterations: 16

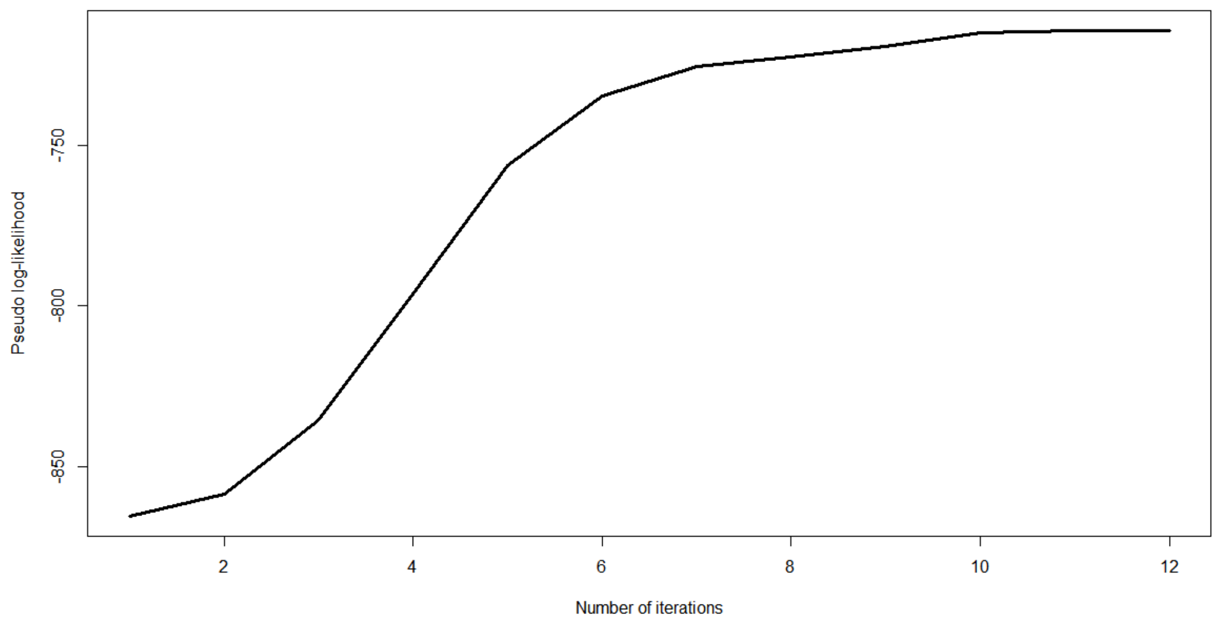


Observation x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Actual Assignments a(x)	0	0	$\boxed{0}$	$\boxed{0}$	0	$\boxed{0}$	0	0	$\boxed{0}$	0	0	1	1	$\boxed{1}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Responsiility r(x)	0	0	$\boxed{1}$	$\boxed{1}$	0	$\boxed{1}$	0	0	$\boxed{1}$	0	0	1	1	$\boxed{0}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Number of misclassifications: 5

Case: $\varepsilon = 0.30$

Number of iterations: 12



Observation x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Actual Assignments a(x)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Responsiility r(x)	1	1	0	1	0	0	1	1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Number of misclassifications: 12

Remark: We clearly see the decrease in the number of iterations with the increase in the number of misclassifications as ε increases.

ε	#misclassifications	#iterations
0.01	2	43
0.03	3	29
0.10	5	16
0.30	12	12

Note: Since the partitioning into train.data and test.data has been done randomly, each time we have a different testing set. Therefore, the table above shows the overall decrease in accuracy of the classification as we loosen the stooping rule.

Appendix

R Code

```
require(mvtnorm)

######################
####IMPORT DATASET####
######################
swissbank <- read.csv("~/swiss-bank.dat", sep="")


########################
####RANDOM PARTITION####
########################
###training set/testing set
rsamp=array(dim=170)
n=1
while(n<=170){
  x=floor(runif(1,min=0.005,max=1.005)*200)
  if(! x%in%rsamp){
    rsamp[n]=x
    n=n+1
  }
}

train.data<-swissbank[rsamp,-1]
test.data<-swissbank[-rsamp,]

###population 1/population 2
pop1=array(dim=85)
n=1
while(n<=85){
  x=floor(runif(1,min=0.005,max=1.005)*170)
  if(! x%in%pop1){
    pop1[n]=x
    n=n+1
  }
}

population1<-train.data[pop1,]
population2<-train.data[-pop1,]




##########################################
####INITIALISATION OF PARAMETER VALUES####
```

```
##########################################
p1=.5
mu1=as.numeric(sapply(population1,mean))
mu2=as.numeric(sapply(population2,mean))
sgm1=matrix(cov(as.matrix(population1),y=as.matrix(population1)),6,6)
sgm2=matrix(cov(as.matrix(population2),y=as.matrix(population2)),6,6)

#######################################
####COMPUTING RESPONSIBILITY VECTOR####
#######################################
responsibility.vector<-function(train.data){
  n=length(as.matrix(train.data[,1]))
  r<<-array(dim=n)
  for(i in 1:n){
    f1=dmvnorm(as.numeric(train.data[i,]),mean = mu1,sigma = sgm1)
    f2=dmvnorm(as.numeric(train.data[i,]),mean = mu2,sigma = sgm2)
    r[i]<<-p1*f1/(p1*f1+(1-p1)*f2)
  }
}

####################################
####UPDATION OF PARAMETER VALUES####
####################################
parameters<-function(train.data){
  mu1<<-rep(0,6)
  mu2<<-rep(0,6)
  s=1-r
  n=length(r)
  for(i in 1:n){
    mu1<<-mu1+r[i]*as.numeric(train.data[i,])/sum(r)
    mu2<<-mu2+s[i]*as.numeric(train.data[i,])/sum(s)
  }
  sgm1<<-matrix(rep(0,36),nrow=6,ncol=6)
  sgm2<<-matrix(rep(0,36),nrow=6,ncol=6)
  for(i in 1:n){
    sgm1<<-sgm1+r[i]*(as.numeric(train.data[i,])-
mu1)%*%t((as.numeric(train.data[i,])-mu1))/sum(r)
    sgm2<<-sgm2+s[i]*(as.numeric(train.data[i,])-
mu2)%*%t((as.numeric(train.data[i,])-mu2))/sum(s)
  }
  p1<<-sum(r)/n
}

######################################
####PSEUDO LOG LIKELIHOOD FUNCTION####
######################################
pseudo.log.likelihood<-function(train.data){
  s=1-r
  q1=1-p1
  t1=0
  t2=0
  for (i in 1:length(as.matrix(train.data[,1]))) {
    lnf1=dmvnorm(as.numeric(train.data[i,]),mean=mu1,sigma=sgm1,log=T)
    lnf2=dmvnorm(as.numeric(train.data[i,]),mean=mu2,sigma=sgm2,log=T)
    t1=t1+r[i]*lnf1
    t2=t2+s[i]*lnf2
  }
  pllf<<-log(p1)*sum(r)+log(q1)*sum(s)+t1+t2
  print(pllf)
}

####################
####EM ALGORITHM####
####################
responsibility.vector(train.data)
pseudo.log.likelihood(train.data)
pllf.array<-array()
pllf.array[1]<-pllf
parameters(train.data)
pseudo.log.likelihood(train.data)
pllf.array[2]<-pllf
k=2
while(TRUE){
  pllf2=pllf
  k=k+1
  responsibility.vector(train.data)
  parameters(train.data)
  pseudo.log.likelihood(train.data)
  pllf.array[k]<-pllf
  if(abs(pllf2-pllf)<0.01)
    break()
}

plot(pllf.array,type = "l",lwd=3,xlab = "Number of iterations", ylab =
"Pseudo log-likelihood")

######################
####CLASSIFICATION####
######################
r.t<-array()
for(i in 1:length(test.data[,1])){
  f1=dmvnorm(as.numeric(test.data[i,2:7]),mean = mu1,sigma = sgm1)
  f2=dmvnorm(as.numeric(test.data[i,2:7]),mean = mu2,sigma = sgm2)
  r.t[i]<-p1*f1/(p1*f1+(1-p1)*f2)
}
names(test.data)[1]<-c("status")
```

```
s1 = sum(abs(test.data$status-round(r.t)))
s2 = sum(abs(test.data$status-(1-round(r.t))))
if(s1<s2){
  result.vector <- cbind("True class"=test.data$status, "Assigned
class"=round(r.t))
  }else{
  result.vector <- cbind("True class"=test.data$status, "Assigned
class"=(1-round(r.t)))
}
View(result.vector)
```

Bibliography

[1] The Elements of Statistical Learning by Hastie. T, Tibshirani. R and Friedman. J (2nd ed.), Page 272, Section 8.5