MID-SEMESTER TEST

STAT8178: Statistical Computing

Student: Francesco Palermo
Student-id: 45539669

The assignment goal is to fit a "mixed" variant of an AR (1) process. Let's define the following rules:

$$X_{n+1} \sim \begin{cases} f_1\text{: N } (\frac{1}{2} X_n \text{ , } \sigma) & \text{if } Z_n = 1 \\ f_2\text{: N } (X_n \text{ , } \sigma) & \text{if } Z_n = 0 \end{cases}$$

where,

$$Z_n = \begin{cases} 1 \text{ with probability } p \\ 0 \text{ with probability } 1\text{- } p \end{cases}$$

In this scenario X_n is observed, while Z_n is unobserved.

Here, the missing data scenario is present and the EM algorithm will help us finding the unknown parameters:

$\Phi = \{ p, \sigma \}$

a) State the density of the conditional distribution of $\boldsymbol{X_{n+1} = x_{n+1}}$ given that $\boldsymbol{X_n = x_n}$ and $\boldsymbol{Z_n = z_n}$.

The following conditional distribution $f_{X_{n+1}|\, X_n, Z_n}(\boldsymbol{X_{n+1} = x_{n+1} \mid X_n = x_n, Z_n = z_n})$ depends on the unknown value of Z_n and therefore, can be split into:

f_1: f $(X_{n+1} \mid X_n, Z_n{=}1) = \frac{1}{\sigma}$ f $\left(\frac{X_{n+1} - \frac{1}{2} X_n}{\sigma} \right)$

f_2: f $(X_{n+1} \mid X_n, Z_n{=}0) = \frac{1}{\sigma}$ f $\left(\frac{X_{n+1} - X_n}{\sigma} \right)$

It is clear that according to the value of Z_n, we end up with two different normal distributions.

Next, we define a mixture model as follow:

f (X_{n+1}) = $p\frac{1}{\sigma}$ f ($\frac{X_{n+1}-\frac{1}{2}X_n}{\sigma}$) + (1- p) $\frac{1}{\sigma}$ f ($\frac{X_{n+1}- X_n}{\sigma}$)

or in a more concise way:

f (X_{n+1}) = p N ($\frac{1}{2}X_n$, σ) + (1- p) N (X_n , σ)

where p represents the probability that an observation comes from f_1, and σ is the standard deviation of both normal distributions in the mixture model.

b) Suppose a sequence (x_1 ,…, x_{n+1}) is observed from the above model. State the complete data log-likelihood.

We are assuming that the above observations from the sample are independent to each other and identically distributed under the mixture model distribution.
Since X_i i.i.d the joint probability density can be rewritten as a product of individual density function.

Next, we define the complete data which is { $X_i, Z_{i1,} Z_{i2}$ } where X_i is the observed data, Z_{i1} and Z_{i2} are the missing data.
For commodity, the complete data can be written as:

COMPLETE DATA

f (X, Z) = f (X|Z) g(Z) = $f_1(\mathrm{X})^{Z_1}$ $f_2(\mathrm{X})^{Z_2}$ g(Z)

where,

g(Z) = $\begin{cases} p^{Z_1} & \text{if } X_i \in f_1, \text{ z=1} \\ (1-p)^{Z_2} & \text{if } X_i \in f_2, \text{ z=0} \end{cases}$

If Z_{ij} were available, we could split the observed data into two different samples, and then simply infer the corresponding parameters in each sample (in the traditional MLE approach).

LIKELIHOOD FUNCTION OF COMPLETE DATA

$L_c(\Phi) = \prod_{i=1}^{n} f_1(x_i)^{z_{i1}} f_2(x_i)^{z_{i2}} \, g(z_i)$

LOG-LIKELIHOOD FUNCTION OF COMPLETE DATA

$\ell_c(\Phi) = \sum_{i=1}^{n}\{ z_{i1} \log f_1(x_i) + z_{i2} \log f_2(x_i)\} + \sum_{i=1}^{n}\{ z_{i1} \log(p) + z_{i2} \log(1\text{-}p)\} =$

$= \sum_{i=1}^{n}\{ z_{i1} \log \frac{1}{\sigma} f_1\left(\frac{x_{i+1} - \frac{1}{2}x_i}{\sigma}\right) + z_{i2} \log \frac{1}{\sigma} f_2\left(\frac{x_{i+1} - x_i}{\sigma}\right)\} + \sum_{i=1}^{n}\{ z_{i1} \log(p) +$

$+ z_{i2} \log(1\text{-}p)\}$

c) Letting e_n^k abbreviate E $[Z_n \mid X_n = x_n, X_{n+1} = x_{n+1}, p = p^k, \sigma = \sigma^k]$, write an expression for the "Q" function, $Q^k(p, \sigma)$ = E $[\ell_c(p, \sigma) \mid x_1, \ldots, x_{n+1}, p^k, \sigma^k]$.

Derive the "Q" function is the first step of the iterative EM algorithm. It is actually the outcome function obtained from the E (expectation) step. It can be defined as the expected value of the complete data log-likelihood. The following steps will illustrate this function from a broader version into a more manageable one.

THE "Q" FUNCTION

$Q^k(p, \sigma)$ = E $[\ell_c(p, \sigma) \mid x_1, \ldots, x_{n+1}, p^k, \sigma^k]$ =

$= \sum_{i=1}^{n}\{E(z_{i1} | x_1, \ldots, x_{n+1}, p^k, \sigma^k) \log f_1(x_i) + E(z_{i2} | x_1, \ldots, x_{n+1}, p^k, \sigma^k) \log f_2(x_i)\} +$

$+ \sum_{i=1}^{n}\{E(z_{i1} | x_1, \ldots, x_{n+1}, p^k, \sigma^k) \log(p) + E(z_{i2} | x_1, \ldots, x_{n+1}, p^k, \sigma^k) \log(1\text{-}p)\}$

For convenience, let's abbreviate e_n^k = E $[Z_n \mid X_n = x_n, X_{n+1} = x_{n+1}, p = p^k, \sigma = \sigma^k]$ from the above formula and after grouping some factors we obtain:

$Q^k(p, \sigma) = \sum_{i=1}^{n} e_{i1}^k(\log f_1(x_i) + \log(p)) + e_{i2}^k(\log f_2(x_i) + \log(1\text{-}p)) =$

$= \sum_{i=1}^{n} e_{i1}^k\left(\log \frac{1}{\sigma} f_1\left(\frac{x_{i+1} - \frac{1}{2}x_i}{\sigma}\right) + \log(p)\right) + e_{i2}^k\left(\log \frac{1}{\sigma} f_2\left(\frac{x_{i+1} - x_i}{\sigma}\right) + \log(1\text{-}p)\right)$

d) Taking e_n^k and the sequence (x_n) as fixed, determine

$$(p^{k+1}, \sigma^{k+1}) = \text{argmax } Q^k(p, \sigma)$$

This is actually the M (maximization) step. We are looking for the *estimates* of *p* and σ that maximize/minimize the Q function. The resulted *estimates* will be the essential part of the iterative method. Let's rewrite the Q function in order to highlight (in orange) the fixed terms as specified above.

$$Q^k(p, \sigma) = \sum_{i=1}^{n} e_{i1}^k \left(\log \frac{1}{\sigma} f_1 \left(\frac{x_{i+1} - \frac{1}{2} x_i}{\sigma} \right) + \log (p)\right) + e_{i2}^k \left(\log \frac{1}{\sigma} f_2 \left(\frac{x_{i+1} - x_i}{\sigma} \right) + \log (1\text{-}p)\right)$$

Let's start to maximize the Q function with respect of *p* first and then an identical approach will be taken for σ. In layman's term a derivate of the Q function will be taken with respect of the parameter of interest and then by assigning it to zero we will solve for the involved parameter.

$$\frac{\partial Q^k(p,\sigma)}{\partial p} = \frac{\partial}{\partial p} \left(\sum_{i=1}^{n} e_{i1}^k \log (p) + e_{i2}^k \log (1\text{-}p)\right) = \sum_{i=1}^{n} \left(\frac{e_{i1}^k}{p} - \frac{e_{i2}^k}{1-p} \right)$$

Now, $\frac{\partial Q^k(p,\sigma)}{\partial p} = 0$

$$\sum_{i=1}^{n} \left(\frac{e_{i1}^k}{p} \right) = \sum_{i=1}^{n} \left(\frac{e_{i2}^k}{1-p} \right) \iff \frac{\sum_{i=1}^{n} e_{i1}^k - p \sum_{i=1}^{n} e_{i1}^k}{p} = \sum_{i=1}^{n} \left(e_{i2}^k \right)$$

$$\iff \sum_{i=1}^{n} e_{i1}^k = p \left(\sum_{i=1}^{n} e_{i1}^k + \sum_{i=1}^{n} e_{i2}^k\right) \iff p^{k+1} = \frac{\sum_{i=1}^{n} e_{i1}^k}{\sum_{i=1}^{n} e_{i1}^k + \sum_{i=1}^{n} e_{i2}^k} \qquad (1)$$

We can now evaluate and define some elements of this division:

- $\sum_{i=1}^{n} e_{i1}^k$ is the estimated proportion of the sample that is from f_1
- $\sum_{i=1}^{n} e_{i2}^k$ is the estimated proportion of the sample that is from f_2

Finally, the sum of these estimated proportions (denominator of *(1)*) is the sample size and therefore we have the expected proportion of the sample that comes from the f_1 distribution.

ESTIMATED PROPORTION

$$p^{k+1} = \frac{\sum_{i=1}^{n} e_{i1}^k}{n}$$

Next, we maximize/minimize the Q function with respect to σ. Note, that really few steps have been omitted such as derivate of constants or grouping some factors together, so it is easier to go through the process.

$$\frac{\partial Q^k(p,\sigma)}{\partial \sigma} = \frac{\partial}{\partial \sigma}\left(\sum_{i=1}^{n} e_{i1}^k \log \frac{1}{\sigma} f_1\left(\frac{x_{i+1}-\frac{1}{2}x_i}{\sigma}\right) + e_{i2}^k \log \frac{1}{\sigma} f_2\left(\frac{x_{i+1}- x_i}{\sigma}\right)\right)$$

$$= \frac{\partial}{\partial \sigma}\left(\sum_{i=1}^{n} e_{i1}^k \log \frac{1}{\sigma} + e_{i1}^k \left(-\frac{1}{2}\left(\frac{x_{i+1}-\frac{1}{2}x_i}{\sigma}\right)^2 \left(-\frac{1}{2}\log(2\pi)\right) +\right.\right.$$

$$\left.\left. + e_{i2}^k \log \frac{1}{\sigma} + e_{i2}^k \left(-\frac{1}{2}\left(\frac{x_{i+1}- x_i}{\sigma}\right)^2 \left(-\frac{1}{2}\log(2\pi)\right)\right)\right.\right.$$

$$= \frac{\partial}{\partial \sigma}\left(\sum_{i=1}^{n} e_{i1}^k \log \frac{1}{\sigma} - \frac{1}{2} e_{i1}^k \left(\frac{x_{i+1}-\frac{1}{2}x_i}{\sigma}\right)^2 + e_{i2}^k \log \frac{1}{\sigma} - \frac{1}{2} e_{i2}^k \left(\frac{x_{i+1}- x_i}{\sigma}\right)^2\right)$$

$$= \frac{\partial}{\partial \sigma}\left(\sum_{i=1}^{n} \log \frac{1}{\sigma} \left(e_{i1}^k + e_{i2}^k\right) - \frac{1}{2} e_{i1}^k \left(x_{i+1} - \frac{1}{2} x_i\right)^2 \left(\frac{1}{\sigma^2}\right) +\right.$$

$$\left. - \frac{1}{2} e_{i2}^k \left(x_{i+1} - x_i\right)^2 \left(\frac{1}{\sigma^2}\right)\right)$$

$$= \sum_{i=1}^{n} - \left(\frac{e_{i1}^k + e_{i2}^k}{\sigma}\right) + \frac{e_{i1}^k \left(x_{i+1}-\frac{1}{2}x_i\right)^2}{\sigma^3} + \frac{e_{i2}^k \left(x_{i+1}- x_i\right)^2}{\sigma^3}$$

Now, $\frac{\partial Q^k(p,\sigma)}{\partial \sigma} = 0$

$$\sum_{i=1}^{n} \frac{-\sigma^2\left(e_{i1}^k + e_{i2}^k\right) + e_{i1}^k \left(x_{i+1}-\frac{1}{2}x_i\right)^2 + e_{i2}^k \left(x_{i+1}- x_i\right)^2}{\sigma^3} = 0$$

$$\sum_{i=1}^{n} \sigma^2\left(e_{i1}^k + e_{i2}^k\right) = \sum_{i=1}^{n} e_{i1}^k \left(x_{i+1} - \frac{1}{2} x_i\right)^2 + \sum_{i=1}^{n} e_{i2}^k \left(x_{i+1} - x_i\right)^2$$

$$\sigma^{k+1} = \pm \sqrt{\frac{\sum_{i=1}^{n} e_{i1}^k \left(x_{i+1}-\frac{1}{2}x_i\right)^2 + \sum_{i=1}^{n} e_{i2}^k \left(x_{i+1}- x_i\right)^2}{\sum_{i=1}^{n} \left(e_{i1}^k + e_{i2}^k\right)}}$$

The standard deviation is a non-negative quantity, so we only take the positive root.

ESTIMATED STANDARD DEVIATION

$$\sigma^{k+1} = \sqrt{\frac{\sum_{i=1}^{n} e_{i1}^k \left(x_{i+1}-\frac{1}{2}x_i\right)^2 + \sum_{i=1}^{n} e_{i2}^k \left(x_{i+1}- x_i\right)^2}{n}}$$

e) Determine the value of e_n^k = E [$Z_n \mid X_n = x_n, X_{n+1} = x_{n+1}, p, \sigma$].

First at all, let us define the following two values:

- e_{i1}^k: probability that the i-*th* observation comes from the first normal distribution f_1 given Φ^k.
- e_{i2}^k: probability that the i-*th* observation comes from the second normal distribution f_2 given Φ^k.

It is clear that $e_{i1}^k + e_{i2}^k = 1$. Therefore, we can define those values as follow:

$$e_{i1}^k = \frac{p^k f_1(X_i;\boldsymbol{\sigma})}{p^k f_1(X_i;\boldsymbol{\sigma})+(1-p^k) f_2(X_i;\boldsymbol{\sigma})} = \frac{p^k \frac{1}{\sigma} f_1\left(\frac{x_{i+1}-\frac{1}{2}x_i}{\sigma}\right)}{p^k \frac{1}{\sigma} f_1\left(\frac{x_{i+1}-\frac{1}{2}x_i}{\sigma}\right)+(1-p^k) \frac{1}{\sigma} f_1\left(\frac{x_{i+1}- x_i}{\sigma}\right)}$$

$$e_{i2}^k = \frac{(1-p^k) f_2(X_i;\boldsymbol{\sigma})}{p^k f_1(X_i;\boldsymbol{\sigma})+(1-p^k) f_2(X_i;\boldsymbol{\sigma})} = \frac{(1-p^k) \frac{1}{\sigma} f_1\left(\frac{x_{i+1}- x_i}{\sigma}\right)}{p^k \frac{1}{\sigma} f_1\left(\frac{x_{i+1}-\frac{1}{2}x_i}{\sigma}\right)+(1-p^k) \frac{1}{\sigma} f_1\left(\frac{x_{i+1}- x_i}{\sigma}\right)}$$

f) Write code to fit this model via the EM algorithm. Report and interpret the results.

```
function [iter,pkp1,sigmakp1] = Em_mixture(file_path,pk,sigmak,maxiter)
    %PRE-PROCESSING STEPS
    %load the file into a variable of str type
    xnp1_str=load(file_path);
    %Transform the str type into a manageable matrix called xnp1
    xnp1=cell2mat(struct2cell(xnp1_str));
    %Create the xn matrix
    xn=circshift(xnp1,1);
    %delete the first observation from both xnp1 and xn
    xnp1=xnp1(2:length(xnp1));
    xn=xn(2:length(xn));

    %EM ALGORITHM
    for iter = 0:maxiter

        diff1=(xnp1-(1/2)*xn);
        diff2=(xnp1-xn);
        f1 = (1/sqrt(2*pi))*(exp(-(diff1.^2)./(2.*sigmak.^2))./sigmak);
        f2 = (1/sqrt(2*pi))*(exp(-(diff2.^2)./(2.*sigmak.^2))./sigmak);
        %create/update eik
        ei1=(pk*f1)./(pk*f1 +(1-pk)*f2);
        ei2=((1-pk)*f2)./(pk*f1 +(1-pk)*f2);
        %updating the new value of p and sigma
        pkp1=sum(ei1)/length(xnp1);
        sigmakp1=sqrt(sum(ei1.*(diff1.^2) + ei2.*(diff2.^2))./length(xn));

        if all(abs([pkp1-pk; sigmakp1-sigmak])<1e-4)
            break;
        else
            pk = pkp1; sigmak = sigmakp1;
        end
    end
end
```

Finally, we can show the Matlab script (see above).

Let us first describe the function definition

```
function [iter,pkp1,sigmakp1] = Em_mixture(file_path,pk,sigmak,maxiter)
```

The following function takes 4 input parameters and return 3 output values back to the user. In particular, the input parameters are:

- *file_path* = It describes the path where we can find our given file. It needs to be specified as string.
- *pk* = This is the initial value for the proportion. It is a floating number between 0 and 1.
- *sigmak* = This is the initial value for the standard deviation. It is a non-negative floating number.
- *maxiter = This* represents the maximum number of the iteration the EM algorithm can run.

On the other hand, this function returns this following values:

- *iter:* The number of iterations the algorithm needed it in order to find the best estimates.
- *pkp1*: This is the estimated proportion given by the EM algorithm.
- *sigmakp1*: This is the estimated standard deviation given by the EM algorithm.

The function *Em_mixture* can be divided into two main parts: **Pre-processing of data** and **EM algorithm.**

PRE-PROCESSING DATA

```
%PRE-PROCESSING STEPS
%load the file into a variable of str type
xnp1_str=load(file_path);
%Transform the str type into a manageable matrix called xnp1
xnp1=cell2mat(struct2cell(xnp1_str));
%Create the xn matrix
xn=circshift(xnp1,1);
%delete the first observation from both xnp1 and xn
xnp1=xnp1(2:length(xnp1));
xn=xn(2:length(xn));
```

Although the code has been heavily commented, I quickly explain some of the most interesting part.

After loading the file into a more manageable matrix called *xnp1*, I created another vector *xn* that will be useful in many mathematical operations. This last vector has been obtained by shifting *xnp1* by one position thanks to the Matlab function *circshift.*

While this function works perfectly, the first observation of *xn* became now useless (since it was assigned by the last observation of *xnp1* due to the circular shifting applied by *circshift*). The following picture helps to better understand this concept and extract the first 3 observation from both *xnp1* and *xn*.

xnp1

1	0.3910
2	0.3658
3	1.1862

xn

1	-30.5735
2	0.3910
3	0.3658

The first subtraction needs to be (*0.3658 – 0.3910*) and those values are evenly found on the second position of those arrays. Finally, we remove the first observation from both vectors. It is important to note that this is a very typical scenario in Time Series when an AR(1) model need to be built.

EM-ALGORITHM

```
for iter = 0:maxiter

    diff1=(xnp1-(1/2)*xn);
    diff2=(xnp1-xn);
    f1 = (1/sqrt(2*pi))*(exp(-(diff1.^2)./(2.*sigmak.^2))./sigmak);
    f2 = (1/sqrt(2*pi))*(exp(-(diff2.^2)./(2.*sigmak.^2))./sigmak);
    %create/update eik
    ei1=(pk*f1)./(pk*f1 +(1-pk)*f2);
    ei2=((1-pk)*f2)./(pk*f1 +(1-pk)*f2);
    %updating the new value of p and sigma
    pkp1=sum(ei1)/length(xnp1);
    sigmakp1=sqrt(sum(ei1.*(diff1.^2) + ei2.*(diff2.^2))./length(xn));

    if all(abs([pkp1-pk; sigmakp1-sigmak])<1e-4)
        break;
    else
        pk = pkp1; sigmak = sigmakp1;
    end
end
```

The EM-ALGORITH is then performed. This is an iterative method for computing ML parameters estimates when there are missing values. This snippet of code is practically everything we manually developed in the previous points, so not too much explanation is needed.

Attention must be given to the following parts:

Diff1, Diff2 is used for subtracting *xnp1 and xn* according to the respectively normal distribution. This helps the reader better evaluate the following density function.

The stopping criteria of this algorithm is shown in the *if statement*. If the condition is true (the estimated parameters do not change significantly) the *break* statement immediately terminates the loop. However, If the condition is false, the loop continues until *maxiter*.

Two function calls will be shown in order to better understand the logic of the algorithm.

```
>> [iter,pkp1,sigmakp1]=Em_mixture("2020MidSem.mat",0.5,3,100)

iter =

    20


pkp1 =

    0.0398


sigmakp1 =

    1.5542
```

Twenty iterations were needed for the algorithm to find the best possible estimates of *p* and σ. Therefore, $[\boldsymbol{p^{k+1}}$ **= 0.0398,** $\boldsymbol{\sigma^{k+1}}$ **= 1.5542]**

The next function call is meant to show that the number of iteration can decrease (not dramatically though, which empower the EM algorithm efficiency) if we had a certain pre-knowledge about the initial.

We are going to insert input parameters that are very close to the actual estimates and see the behaviour.

```
>> [iter,pkp1,sigmakp1]=Em_mixture("2020MidSem.mat",0.04,1.7,100

iter =

     9


pkp1 =

    0.0394


sigmakp1 =

    1.5546
```

Nine iterations were now needed for the algorithm to find the best possible estimates of p and σ. In conclusion, the EM algorithm found the best estimates very quickly even when the initial guess was further to the actual ML estimates.