Using the EM algorithm and Gaussian Processes to solve a gene expression problem

Sebastian Ciobanu

Faculty of Informatics
Alexandru Ioan Cuza University
Iași, Romania
ciobanu.sebastian.adrian@info.uaic.ro

Liviu Ciortuz

Faculty of Informatics
Alexandru Ioan Cuza University
Iași, Romania
ciortuz@info.uaic.ro

The problem we are dealing with here concerns the analysis of *gene expression* (from bioinformatics) using machine learning techniques. Specifically, we will use the EM algorithm for Gaussian Process Mixture Models (GPMM) to solve this problem. By doing so, we will be able to perform two tasks simultaneously: regression (via *Gaussian Process Regression - GPR*) and clustering. The learning algorithm also involves the *gradient ascent* method and some empirical results are presented.

1 Introduction

The problem tackled here is related to *gene expression* (bioinformatics). Gene expression represents the process by which the information contained within a gene (our DNA) becomes a useful product, such as a protein. Our goal is to understand how the expression levels of genes (in a cell or a tissue) change with time. The expression level of a gene indicates the amount of gene-product (i.e., messenger RNA) in the cell.

The *data* was provided by prof. Tommi Jaakkola [5, 6]. It consists of 90 instances, i.e., genes whose expression levels were measured at 30 timepoints. (As one may notice, this is a time series problem.) We would like to find for each gene a function that goes through its expression levels, therefore we will perform *regression*. We can do 90 separate regression tasks, but since preliminary analysis suggests that the expression curve of each of those 90 genes follows, roughly, one of three patterns, it is natural to think that the genes can be grouped, and therefore we will also involve *clustering* in the process. Gene clustering indeed corresponds to biological processes inside the cell and therefore can lead to valuable insights. We add the following observation regarding the gene expression curves: these three patterns are present in this dataset, but in other datasets the expression curves can (and do) look differently.

In order to perform *regression*, we will use *Gaussian Process Regression (GPR)* [8]. To perform *clustering*, we will use *the EM algorithm* [2] for the Gaussian Process Mixture Model, henceforth denoted EM/GPMM. The **Expectation-Maximization** (EM) algorithm can be used to fit a mixture distribution to the data we work with. An instance generated from a probability mixture model can be described as follows:

- throw a (not necessarily fair) die with k sides and let the result be i
- generate a number (or a vector if the distribution is multivariate) according to the i-th distribution.

The number of distributions (k) must be known beforehand. In fact, the EM algorithm is a general algorithmic schema, i.e., the specific algorithm differs from problem to problem (only the general structure remains the same). It is an iterative algorithm: the two iterative steps are E (Expectation) and M (Maximization). It is used at maximizing a function called log-likelihood of observed data. As suggested, there

Submitted to:
FROM 2019

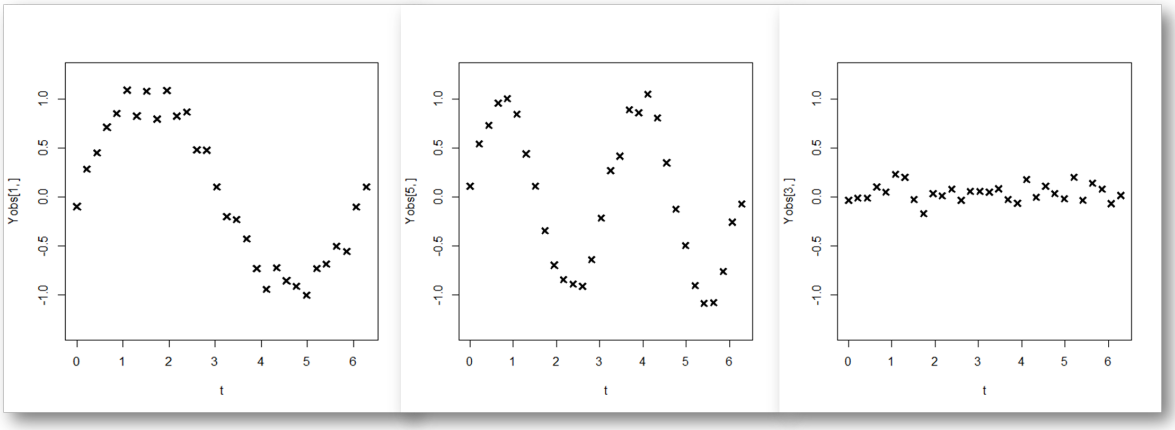

Figure 1: The three types of expression curves in our data

is another type of data besides observed data: latent data. So, the problem to solve must be modelled in such a way that there are data of both types.

We also care about the parameters of each Gaussian process (specifically, they will be the output of the M-step of the EM/GPMM algorithm), because with them we carry out regression. Given the expression levels of a gene at timepoints $t_1, \ldots, t_{30}$, we find the cluster this instance would belong to (we look at the cluster that has the highest probability of containing this instance), and we carry out Gaussian Process Regression on this data with the parameters corresponding to the cluster we discovered this instance belongs to. In other words, by post-processing the output of the E-step, we perform clustering and by post-processing the output of the M-step, we perform regression. So, the two tasks can be performed quite naturally with EM/GPMM and can be seen as being, up to a certain degree, interdependent (in the while loop, the output of the E-step goes as input for the M-step). Note that we could not use just EM for Multivariate Gaussian Mixture Models [7] (abbreviated EM/GMM) instead of EM/GPMM because the output from the M-step of the EM/GMM algorithm does not help us in carrying out regression.

The expression curves of all 90 genes follow one of the three patterns shown in Figure 1.

Some rows of the **data** available for this problem are shown in Figure 2.

2 Method: EM/GPMM

	Expression level at t_1	Expression level at t_2	...	Expression level at t_{30}
y_1	-0.09900893	0.2818237	...	0.1033819
y_2	-0.00857957	0.1534053	...	0.08532405
y_3	-0.03709729	-0.00954268	...	0.01441636
...	...	...	...	...

t_1	t_2	...	t_{30}
0.0000000	0.2166616	...	6.2831853

Figure 2: The data we work with

A **stochastic (random) process** is an indexed collection of random variables. More formally: $f : \mathscr{X} \to \mathscr{F}(\Omega, \mathbb{R})$ - stochastic process; $\mathscr{X}$ is an index set; $\mathscr{F}(\Omega, \mathbb{R})$ is a set of random variables.

A **gaussian process** is a stochastic process with the following property: $\forall x_1, \ldots, x_n \in \mathscr{X} : (f(x_1), \ldots, f(x_n))$ has a multivariate normal distribution.

A regression task can be modelled as follows: $y_i = f(x_i) + \varepsilon_i, i = 1, \ldots, n$ together with some assumptions. The assumptions of **Gaussian Process Regression** (GPR) are: $f(\cdot) \sim \mathscr{GP}(0, k_{ini}(\cdot, \cdot))$, $\varepsilon_i \sim \mathscr{N}(0, \sigma_n^2)$, where

$\mathscr{GP}(0, k_{ini}(\cdot,\cdot))$ represents a Gaussian process with 0 as the mean function and k_{ini} as the covariance function, or we can combine both into one assumption if the data we work with does not have duplicates in terms of input (x_i): $f(\cdot) \sim \mathscr{GP}(0, k(\cdot,\cdot))$, where

$$k(x,x') = k_{ini}(x,x') + \sigma_n^2 \delta(x,x') \text{ and } \delta(x,x') = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

The mean function does not have to meet any constraints. The covariance function must be a **kernel** function [1].

We will not derive the E and M steps here for EM/GPMM, but we will give the *final formulas*, obtained based on references [5, 10]. Note that we employed a Gaussian (RBF) kernel function for each Gaussian Process. Therefore a *hypothesis* (i.e., model) h_j for a given Gaussian process was identified by the parameters $(\sigma_f)_j$ (from RBF; from function f), ρ_j (from RBF; from function f), $(\sigma_n)_j$ (from ε_i; from noise). k_{ini} and k represent two kernel functions applied on $x \in \mathbb{R}$ and $x' \in \mathbb{R}$. When writing parameter$^{(t)}$, it will signify the value of that parameter at iteration t.

$$k_{ini}(x,x') = \sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\rho^2}\right);\ k(x,x') = \sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\rho^2}\right) + \sigma_n^2 \delta(x,x')$$

$$k_j^{(t)}(x,x') = ((\sigma_f)_j^{(t)})^2 \exp\left(-\frac{(x-x')^2}{2((\rho)_j^{(t)})^2}\right) + ((\sigma_n)_j^{(t)})^2 \delta(x,x')$$

$$K_j^{(t)} \overset{\text{not.}}{=} K(h_j^{(t)}) \overset{\text{not.}}{=} K_j^{(t)}(X,X), \text{ where } X \text{ is a matrix representing the training set}$$

E step:

$\gamma_{ij}^{(t+1)} \overset{\text{not.}}{=} \dfrac{\pi_j^{(t)} \mathscr{N}(x_i; 0, K_j^{(t)}))}{\sum_{l=1}^{k} \pi_l^{(t)} \mathscr{N}(x_i; 0, K_l^{(t)}))}$ - the a posteriori probability that the instance i belongs to the cluster j

M step:

$$\pi_j^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{(t+1)}}{n} \text{ - the a priori probability that an instance belongs to the cluster } j$$

Let $\theta_j \in \{(\sigma_f)_j, (\sigma_n)_j, \rho_j\}$ in the next expression:

$$\frac{\partial Q}{\partial \theta_j} = \frac{1}{2} \sum_{i=1}^{n} \left(\gamma_{ij}^{(t+1)} x_i^T (K_j^{(t)})^{-1} \left(\frac{\partial K_j}{\theta_j} \right)^{(t)} (K_j^{(t)})^{-1} x_i \right) - \frac{\sum_{i=1}^{n} \gamma_{ij}^{(t+1)}}{2} \text{Tr}\left((K_j^{(t)})^{-1} \left(\frac{\partial K_j}{\theta_j} \right)^{(t)} \right)$$

$$\frac{\partial k(x,x')}{\partial \sigma_f} = 2\sigma_f \exp\left(-\frac{(x-x')^2}{2\rho^2}\right);\ \frac{\partial k(x,x')}{\partial \sigma_n} = 2\sigma_n \delta(x,x');\ \frac{\partial k(x,x')}{\partial \rho} = \sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\rho^2}\right) \frac{(x-x')^2}{\rho^3}$$

Unlike to the well-known case of EM/GMM, here the maximization of the "auxiliary" function Q (which is an upper bound of the log-likelihood function of the data) does not give rise to a closed-form solution. This is why a numerical method had to be used in order to increase the function Q. Therefore, in order to find $\theta_l^{(t+1)}$, we used the *gradient ascent* optimization method. Note that we just locally maximize the function Q, and not necessarily globally maximize it. It works because the function Q does not need to be globally maximized, it is enough to increase it at each iteration.

3 Results

A natural question that comes to mind is whether the solution makes the regression task (which was our main goal) better in some way or not. As seen in Figure 3, the deviations from the mean are decreased when working with EM/GPMM and so we have more confidence in what the values on the real expression curve are.

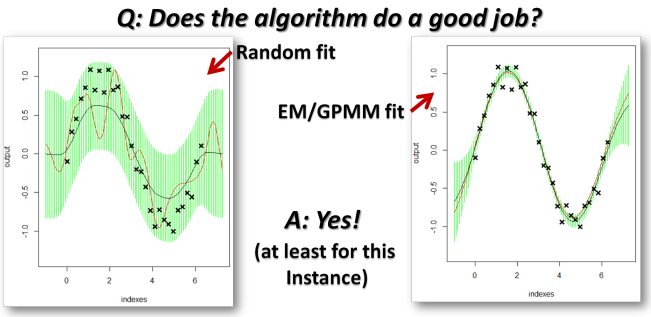

Figure 3: Random fit vs. EM/GPMM fit

For a given number of clusters k, we ran for multiple times the algorithm and, in the end, took only the result with the best (highest) value of the log-likelihood of the observed data. Because we knew a priori that the expression curves of all 90 genes follow one of three patterns, the values of k we tried were only: 2, 3, 4, 5 (i.e., around 3). We visually present the results in **two versions** (Figure 4 and Figure 5 for the first version; Figure 6 and Figure 7 for the second version). In both versions, we have a plot for each cluster. In the first version, what is plotted is one sample function per type of curve. In the second version, all the mean functions resulted from performing GP Regression on a cluster are put on the same plot.

It can be noticed that when $k = 2$ a cluster contains only a type of curves, and the other one contains two types of curves. When $k = 3$, the result is the natural one knowing that our data contains only 3 types of expression curves. When $k = 4$, a natural cluster is split into two. When $k = 5$, a natural cluster is split into two and a cluster is empty. We remind that those results were obtained after running the algorithm **multiple** times for a given k. Running the algorithm just once does not, in general, give the results from above (including the case when $k = 3$).

Observation: We also clustered the time series using k-means, a classic algorithm for clustering. When using $k = 3$, each cluster was perfectly found by k-means. An advantage of k-means is that it is faster (time - less than a second in our case) than EM. After that, one could learn/train a GP on each cluster. A disadvantage is that we lose two things: the soft clustering part and the interaction of *all* instances when finding the parameters of GPs.

4 Conclusion

We presented an application of the EM algorithm and Gaussian Processes in bioinformatics. The problem consists of identifying functional patterns in gene expression data, and the solution we explored used the EM/GPMM algorithm in order to perform simultaneously a clustering task and a regression task.

As *future work*, we could try to compare these results with the results from the simple **alternatives**: perform a hard-clustering (using hierarchical clustering algorithms; for k-means, see the observation in *3. Results*) or EM/GMM and then, in each cluster, fit the parameters of a Gaussian process using the principle of maximum likelihood estimation (MLE).

In the literature [4] there are also approaches that do not use Gaussian Processes, but **Dirichlet Processes** [9] and we could try this approach, too. Furthermore, we can compare the results provided

by these methods on **other datasets**, and include a **numerical analysis** of the quality of the clusters we obtained [3].

References

[1] Nello Cristianini & John Shawe-Taylor (2000): *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, 1 edition. Cambridge University Press.

[2] A. P. Dempster, N. M. Laird & D. B. Rubin (1977): *Maximum likelihood from incomplete data via the EM algorithm. JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 39(1), pp. 1–38.

[3] Guojun Gan, Chaoqun Ma & Jianhong Wu (2007): *Data clustering - theory, algorithms, and applications.* SIAM. Available at `http://bookstore.siam.org/sa20/`.

[4] James Hensman, Magnus Rattray & Neil D. Lawrence (2014): *Fast nonparametric clustering of structured time-series*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, doi:10.1109/TPAMI.2014.2318711.

[5] Tommi Jaakkola: *MIT, 6867 Machine Learning, Fall 2006, Problem Set 5, pr. 2*. Available at `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/assignments/hw5.pdf`.

[6] Tommi Jaakkola: *MIT, 6867 Machine Learning, Fall 2006, Problem Set 5, pr. 2, Dataset*. Available at `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/assignments/prob2_data.zip`.

[7] Thomas M. Mitchell (1997): *Machine Learning*, 1 edition. McGraw-Hill, Inc., New York, NY, USA.

[8] Carl Edward Rasmussen & Christopher K. I. Williams (2005): *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[9] Claude Sammut & Geoffrey I. Webb, editors (2017): *Encyclopedia of Machine Learning and Data Mining*. Springer, doi:10.1007/978-1-4899-7687-1.

[10] Volker Tresp (2001): *Mixtures of Gaussian processes*. In: *Advances in Neural Information Processing Systems 13*, MIT Press, pp. 654–660.

A Appendix: Figures with results

The following two pages contain four figures that represent the results of applying EM/GPMM on the dataset we worked with. Those figures were referenced in the *Results* section.

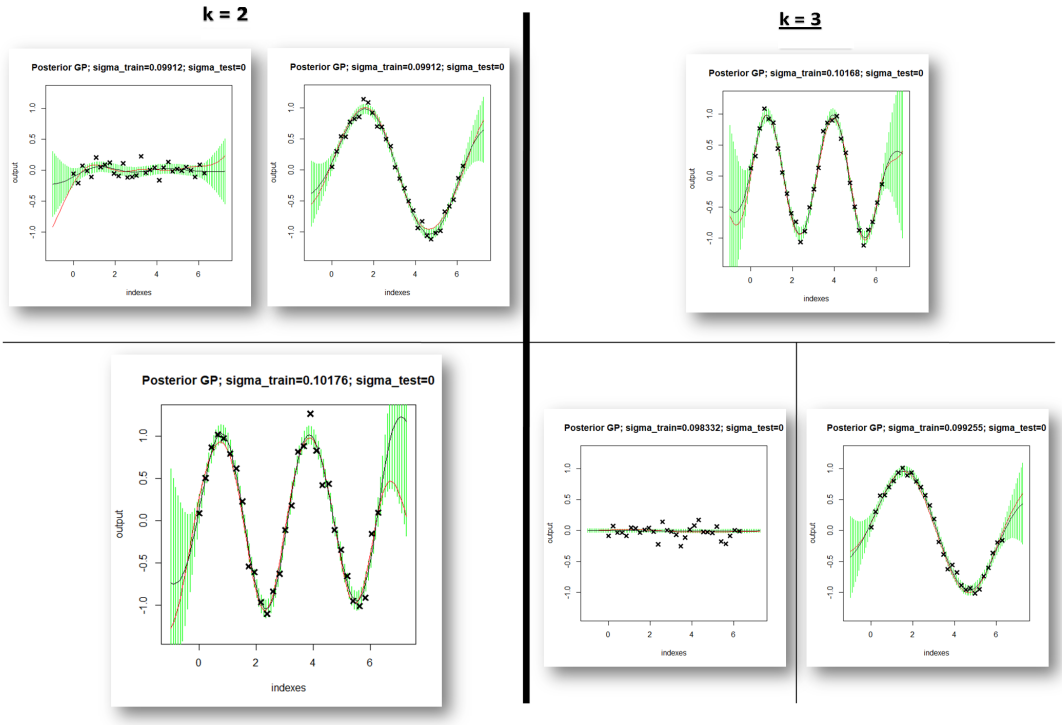

Figure 4: Results: first version, $k \in \{2,3\}$

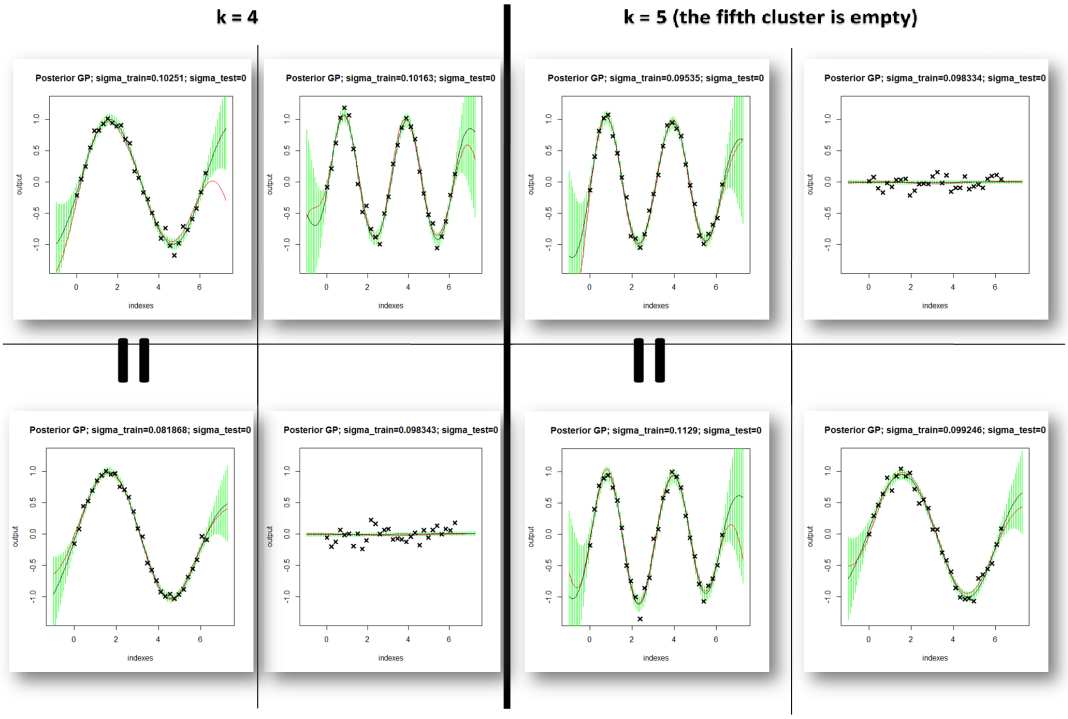

Figure 5: Results: first version, $k \in \{4,5\}$

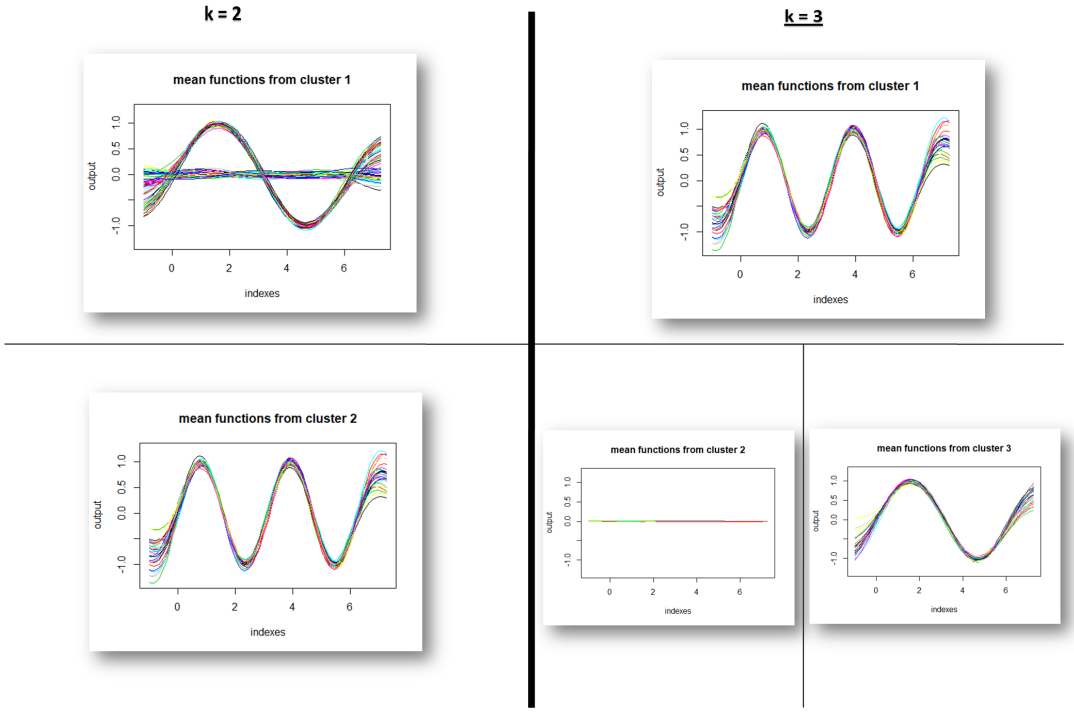

Figure 6: Results: second version, $k \in \{2,3\}$

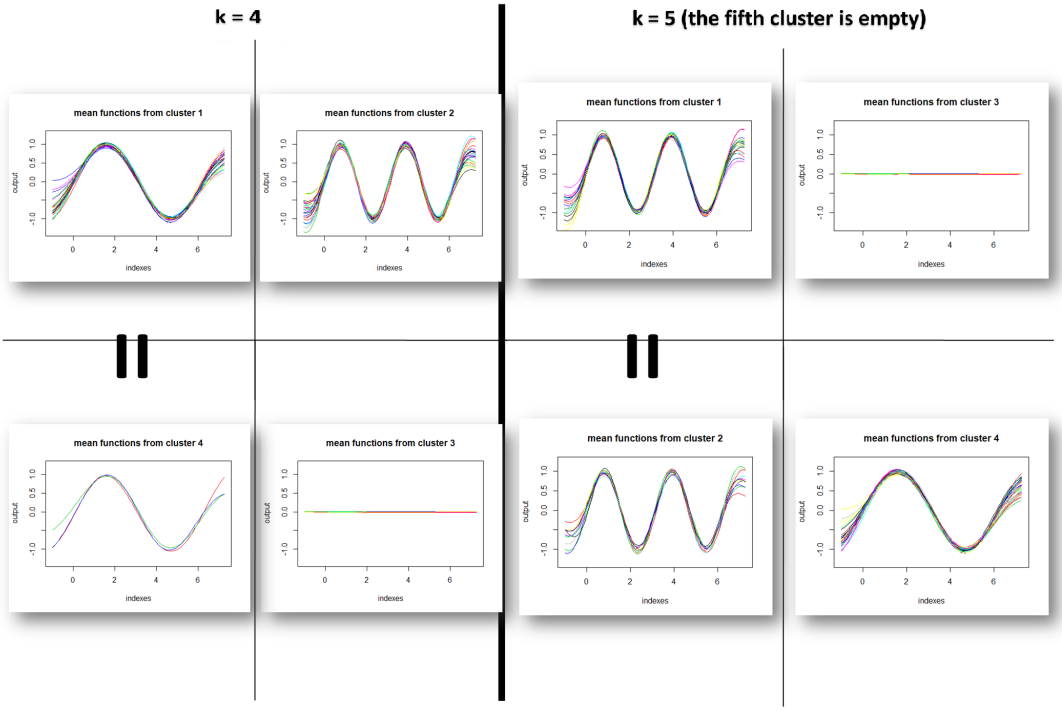

Figure 7: Results: second version, $k \in \{4,5\}$