Second Group Assignment

This is your second of three homework assignments. The value of this assignment is 8%. Each group will receive an extended version of the electricity consumption dataset to be analyzed using the R language and environment for statistical computing and graphics.

This assignment builds directly on the data exploration phase in the previous assignment and addresses the training of Hidden Markov models for the purpose of **unsupervised intrusion detection** in supervisory control systems. For household electricity consumption data, we can generally assume that normal instances occur far more frequent than anomalies; that is, except for the inevitable noise, the datasets considered are largely comprised of "normal'' data instances. Thus, the resulting models learned during training should be robust to the relatively few anomalies a dataset may contain (e.g., caused by technical component malfunctions).

Please complete the tasks described below and submit an electronic copy of your solution, only one per group, through CourSys by **October 27, 2020**.

For the dataset assigned to your group, complete the following tasks:

1. For one of the three observed response variables, `Global_active_power`, `Global_reactive_power` or `Global_intensity`, determine a time window on weekdays that shows a clearly recognizable electricity consumption pattern over a time period of several hours. Provide a rational for your choice by explaining the pattern exhibited by the chosen time window, linking the changes in power consumption to assumed routine activities in the real world. You may do this by writing a short script (narrative) of what typically happens at which time during the time window and how this leads to the observable changes in power consumption.

2. Consider the dataset provided here for training a number of univariate HMMs that each have a different number of states across a range from not less than 4 states to not more than 20 states. For each HMM, compute the log-likelihood measure on the training dataset. In addition, compute the *Bayesian information criterion*[1], or BIC, as a measure of the complexity of your model. The goal is to find the intercept of the two plots for log-likelihood and BIC values respectively so as to determine the best model (avoiding overfitting). You may not need to train HMMs for each and every number of states within the range by making smart choices.

   How likelihood and BIC measures are used:

[1] The Bayesian information criterion (BIC) is a criterion for model selection among a finite set of models; the model with the lowest BIC is preferred. It is based, in part, on the likelihood function and it is closely related to the Akaike information criterion (AIC). When fitting models, it is possible to increase the likelihood by adding parameters but doing so may result in overfitting.

Likelihood of a sequence of observations for a given model indicates how likely the model can produce this sequence of observations. In other words, a high likelihood means that the model is a good representation of the dataset and can produce the observation sequence with a relatively high chance. It is important to note that we are actually calculating **log-likelihood** (instead of likelihood), which is always a negative number.
On the other hand, BIC is a measurement of the complexity of our model. Increasing the number of states of an HMM, may increase log-likelihood but, at the same time, also increases the complexity of the model. This trade-off is inescapable. We aim to strike a sensible balance between log-likelihood and BIC value for a good model.

In this course we train HMMs using the **depmixS4** package in R. Training an HMM in depmixS4 has 2 steps: first, you specify the parameters of the HMM, and second, you fit the model to the data. One important parameter you should specify in the first step is the number of HMM states as explained earlier.

Below are the commands you need to train an HMM and get log-likelihood and BIC values as well as the links to the documentations of the depmixS4 package.

```
model <- depmix(response =, data =, nstates = )
fitModel <- fit(model)

summary(fitModel)
```

https://cran.r-project.org/web/packages/depmixS4/vignettes/depmixS4.pdf
https://cran.r-project.org/web/packages/depmixS4/depmixS4.pdf

3. As a basic point anomaly detection method, consider the *Moving Average,* as explained below. Compute the moving average over any of the weeks (7 days) represented in the dataset using a fixed size **moving time window** of consecutive datapoint points. Decide on a threshold value to detect point anomalies, i.e. data points that relative to the moving average value differ considerably (due to noise or other factors). The basic assumption in forecasting of time series values considered here is that the moving average of the most recent observations provides a reasonable expectation (estimate) for the next observation. This is a “simplification” which is convenient but not always true in general.

   Step 1: Consider a fixed size window of observations (for instance, a window of 7-10 observations).

   Step 2: For the chosen response variable, calculate the average of the window and then slide the window by **one** observation at a time. (This will eventually smoothen the curve of that response.)

Step 3: At any point of Step 2, if the difference of the observation and the calculated average is either above or below a certain threshold, that observation can be considered a point anomaly of the response variable in question.

Please submit a short report for your solution and the R code through CourSes by OCT 27.

Thank you!