Project in Energy I - 10 ECTS

Assessment of the available EV battery flexibility via a statistical characterisation of data gathered from a fast charger installed at DESL-EPFL

Riccardo Vasapollo

Report Type:	Final
Professor:	Mario Paolone
Doctoral Assistant:	Sherif Alaa Salaheldin Fahmy
Date:	July 4, 2021

1 Introduction

This report was developed in the framework of MESH4U, a demonstration project coordinated by Romande Energie and DESL-EPFL. More specifically, the project aims at validating real-time control strategies and coordinate distributed energy resources (DERs)in a real 20kV-distribution grid in Aigle, Switzerland, hosting non-controllable multi-MW PV and a small hydro power plant. The controllable units in this unique field test site area MW-class battery storage system and two 150kW-EV fast charging stations installed by GoFast (EVCS).

With the increasing proliferation of electric vehicles (EV), the need for a robust and reliable forecasting tool has become evident as uncontrolled charging might cause grid operational violations or reduce the quality of the power supply. To counter this problem, real-time control schemes are used to allocate EV charging station set-points, while coordinating DERs. In this context, forecasting EV flexibility could significantly increase the quality of the EV-aware control action. The most widely diffused forecasting techniques can be categorized as: i) parametric methods, ii) non-parametric methods. The first methods assume that the probabilistic distribution can be fully represented by a set of parameters, while in the second category the number of parameters is not fixed and it can grow with the sample size. A large number of contributions featuring non-parametric methods can be found in the literature of EV forecasting. For instance, [1] proposes a method based on Gradient Boosted Trees to forecast schedulable capacity of EV fleets. Another example can be found in [2] where Support Vector Machine is used to forecast EV charging demands.

In this report, we focus on the so-called Gaussian Mixture Model (GMM), which is a parametric method. To the best of the authors' knowledge, the application of this method for forecasting purposes of EV charging flexibility has been studied very little. Among the few instances that can be found in the literature, [3] developed a methodology to cluster charging sessions among generic connection profiles, therefore GMM was used for clustering purposes and not forecasting. Conversely, [4] uses GMM to model the charging probability of EVs, hence no the study is limited to only a restricted set of features, which are not fully representative of the EV user behaviour.

In this paper, we design and develop a GMM-based MATLAB Toolbox for EV flexibility characterization which exhibits small forecasting error, while focusing both on EV user behaviour features and EV charging sessions features. Finally, the Toolbox can be used to generate scenarios for EV-aware control frameworks based, or even simply forecast the EV load for a target day.

The discussion will unfold as follows: Section 2 summarizes the theoretical foundations of GMM, Section 3 is dedicated to investigating the global capability of GMM under a controlled experiment, Section 4 illustrates the development of our final GMM-based Forecasting MATLAB Toolbox, and finally Section 5 reports the numerical experiments carried out to benchmark our implementation on a real world data-set of EV charging session during 5 months in Switzerland provided by GoFast.

2 Gaussian Mixture Model

2.1 Theory

Gaussian Mixture Model (GMM) is a parametric probability density function identified by a weighted sum of Gaussian component density functions [5]. GMM probability density function with M components reads:

$$p(\mathbf{x} \mid \lambda) = \sum_{m=1}^{M} w_m g(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{1}$$

with $\mathbf{x}$ a K-dimensional vector, w_m, $m = 1, ..., M$ are the mixture weights and $g(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, $m = 1, ..., M$ are the Gaussian components densities. The mixture weights sum up to one unit, namely $\sum_{m=1}^{M} w_m = 1$. Each component is a K-variate Gaussian density of the form:

$$g(\mathbf{x} \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) = \frac{1}{(2\pi)^{K/2} |\Sigma_m|^{1/2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_m)' \Sigma_m^{-1} (\mathbf{x} - \boldsymbol{\mu}_m) \right\} \tag{2}$$

where $\boldsymbol{\mu}_m$ is the mean vector and Σ_m is the covariance matrix. In summary, the parameters that fully identify a GMM are mean vectors, covariance matrices and mixture weights. Henceforth, we denote λ as the collective vector of GMM-parameters:

$$\lambda = (\boldsymbol{\mu}_m, \Sigma_m, w_m), \quad m = 1, ..., M \tag{3}$$

GMM has different variants based on the type of covariance matrix Σ_m. More specifically, the latter can be full rank or constrained to be diagonal. Furthermore, parameters can be shared / tied among components (typically the covariance matrix). Noticeably, GMM does not need full rank covariance matrices to model the correlation between features as each Gaussian component contributes to the overall feature density. In other words, a linear combination of diagonal covariance matrices is fully capable of representing statistical dependence among features.

2.2 Expectation-Maximization (EM) Algorithm

EM-algorithm is an iterative procedure that attempts to identify the maximum likelihood estimator for parameters of a parametric probability distribution. In this context, the aim is to identify the vector of GMM-parameters λ such that the corresponding GMM best matches the probability distribution of the fitted vector of data $\mathbf{x}$. Given a sequence of N features $X = \{x_1, ..., x_N\}$ where x_n is a K-dimensional vector of observations, the log-likelihood of the complete GMM model is denoted by:

$$l(X \mid \lambda) = \sum_{n=1}^{N} \log p(\mathbf{x}_n \mid \lambda) \tag{4}$$

Expression (4) is nonlinear in λ, therefore nonlinear optimization techniques or iterative procedures are needed to maximize it. To this end, the main idea of EM -algorithm is to compute at each iteration a new λ^{new} with $l(X \mid \lambda^{new}) > l(X \mid \lambda)$ until a given threshold is reached. The update-expressions are constructed so that the log-likelihood function increases monotonically after every iteration.

Particularly, the Expectation step consists in the computation of the posterior probability [1] of generic Gaussian component membership:

$$\Pr(m \mid \mathbf{x}_n, \lambda) = \frac{w_m g(\mathbf{x}_n \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{k=1}^{M} w_k g(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \tag{5}$$

[1] The posterior probability distribution is the probability distribution of an unknown quantity, treated as a random variable, conditional on the evidence obtained from an experiment or survey.

A N-by-M matrix of membership weights is computed with rows summing up to unity. Thereafter, the Maximization step includes the computation of next iteration λ_{new} starting from the Expectation step result. The expressions to update weight, mean and covariance for each component are:

$$\begin{aligned} w_m &= \frac{1}{N}\sum_{n=1}^{N} \Pr(m \mid \mathbf{x}_n, \lambda) \\ \boldsymbol{\mu}_m &= \frac{\sum_{n=1}^{N} \Pr(m \mid \mathbf{x}_n, \lambda)\mathbf{x}_n}{\sum_{n=1}^{N} \Pr(m \mid \mathbf{x}_n, \lambda)} \\ \sigma_m^2 &= \frac{\sum_{n=1}^{N} \Pr(m \mid \mathbf{x}_n, \lambda)x_n^2}{\sum_{n=1}^{N} \Pr(m \mid \mathbf{x}_n, \lambda)} - \mu_n^2 \end{aligned} \tag{6}$$

To summarize the main steps of the algorithm, the following schema is presented:

Algorithm 1: EM-algorithm

Result: λ^{t+1}
t = 0, initial estimate λ^t;
while *True* **do**
  E step) Compute the components-membership posterior probability by plugging λ^t in (5) ;
  M step) Plug component-membership posterior probabilities in (6) to compute λ^{t+1} ;
  if $\left|\lambda^{t+1} - \lambda^t\right| > \epsilon$ **then**
    t = t+1;
    contrinue;
  else
    break;
  end
end

3 Controlled experiment

To investigate the capabilities of GMM models, a controlled experiment was designed and implemented on MATLAB. This experiments was developed as a playground to get insights on the inner dynamics of GMM model and to understand the importance of the different tuning parameters. Fig. 1 reports a Flowchart of the experiments' structure where the main activities can be summarized by the following list:

1. Generate random values for each feature according to a given distribution
2. Create the matrix of synthetic features where the rows are observations and the columns are features
3. PCA analysis to determine the number of components covering 99% of variance
4. Fit the simulated observation matrix with the optimal GMM model where the maximum number of component is defined by PCA analysis
5. Sample the fitted GMM-model
6. Fit the sampled data to the original known distribution used for generating the input matrix
7. Evaluate the distance between the original and final distributions

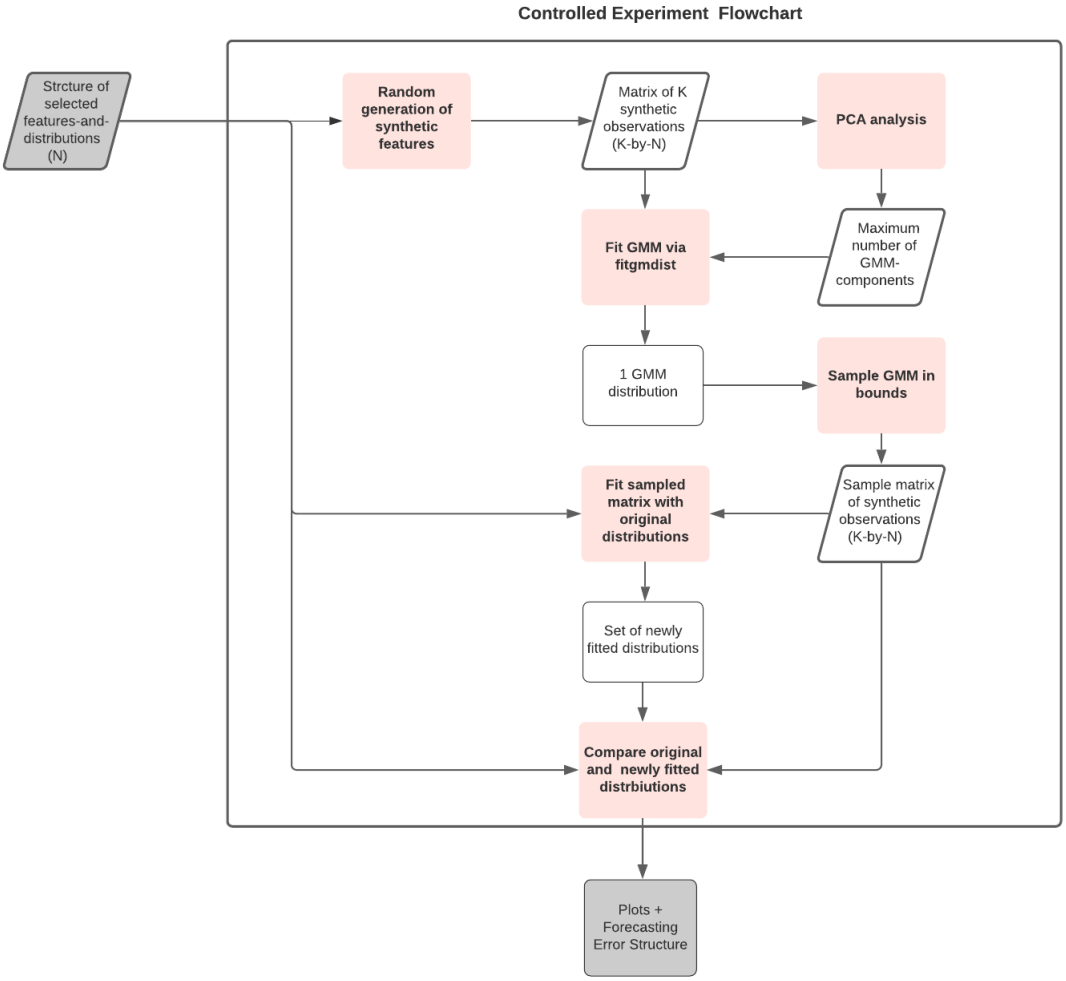


Figure 1: Controlled Experiment Flowchart

In this context, our controlled experiment has multiple goals: i) Investigate what is the global capability of GMM for EV flexibility forecasting, ii) Determine the impact of Arrival Time on forecasting and explore different approaches to handle this feature, iii) Explore the option of using PCA analysis to compute the maximum number of GMM components, iv) Develop a script based on fitgmdist that yields the optimal GMM model (parameters tuning),v) Evaluate what is the impact of truncation on the sampling process (simplistic Accept-Reject Policy), vi) Develop a first testing framework to assess the forecasting error (MAE, MAPE, R, KSIPer2).

3.1 Random generation of synthetic features

First of all, a set of features is identified and motivated by a thorough literature analysis on the current state-of-the-art methods for EV flexibility forecasting. By combining the inputs from the already existing methods and intuitions based on the desired requirements of our research, the final set of features is illustrated in Tab. 1.

A large number of peer-reviewed articles [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18] share a strong interest in investigating features that are representative either of a generic EV charging session such as the *Arrival Time* or the *Charge Duration* or of the EV users behaviour such as the *Desired SOC*. In our selection, a combination of the two is presented as the ultimate goal is to globally characterize the

Feature	Unit	Description
Arrival Time	datetime	Datetime at which the charge session starts
Initial SOC	%	State-Of-Charge of the EV battery at *Arrival Time*
Desired SOC	%	State-Of-Charge that EV user desires to achieve
Battery Capacity	kWh	Maximum capacity of the EV battery
Charge Duration	min	Duration of the charge session
Maximum Power	kW	Maximum power at which the EV battery can charge

Table 1: Most relevant features related to EV flexibility probabilistic characterization

flexibility brought by EV charging stations into the connected grid. In this context, both the behaviour of the EV users and the technical description of their charging experience is believed to be relevant.

Regarding the generation of random values, a large set of possible distribution is key to analyse how the GMM-fitting performs for more complex configurations. Consequently, we allow a range of different distribution, namely Normal, Poisson, Exponential and Log-Normal. Each features can be set to follow any of the aforementioned single distributions. For the sake of completeness, Tab. 2 reports the distributions' probability density function, parameters and conditions.

Distribution	Probability density function	Parameters	Conditions
Normal	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(\mathbf{x}-\mu)^2}{\sigma^2}}$	Mean (μ)	$-\infty < \mu < \infty$
		Standard Deviation (σ)	$\sigma \geq 0$
Lognormal	$\frac{e^{-\frac{(\log(\mathbf{x})-\mu)^2}{2\sigma^2}}}{\mathbf{x}\sqrt{2\pi}\sigma}$	Mean of logarithmic values (μ)	$-\infty < \mu < \infty$
		Standard deviation of logarithmic values (σ)	$\sigma \geq 0$
Exponential	$\lambda e^{-\lambda \mathbf{x}}$	Mean (λ)	$\lambda > 0$
Poisson distribution	$\frac{\lambda^K e^{-\lambda \mathbf{x}}}{K!}$	Mean (λ)	$\lambda \geq 0$

Table 2: Supported distributions for the synthetic features generation

Furthermore, to guarantee a more realistic representation, each distribution is truncated within a range of values. For instance, *Battery Capacity* and *Maximum Power* bounds are extracted from [19] where technical specifications of EVs currently on the market is available. *Initial SOC* is set to be between 10%-60%, while *desired SOC* between 60% and 100%. *Charge Duration* is set to be between 10 and 180 minutes. In this context, the truncation process increases significantly the complexity of the model as the truncated distribution will not correspond to the original one.

Tab. 3 reports the parameters necessary to generate the aforementioned distributions, namely Mean, Standard Deviation, Mean of logarithmic values and Standard deviation of logarithm values. In addition, their ranges are reported.

Noticeably, *Arrival Time* is handled differently as datatime formats are notoriously challenging to forecast. Several approaches are available in the literature. For instance, converting the datetime to a serial date number [2] is one of the most-widely diffused and efficient. This is motivated by the numerical complexity of handling separately the 6 separate columns representing year, month, day, hour, minute, second. In the result section, we investigate this approach by designing *Arrival Time* to be normally distributed between 01/01/2018 and 01/01/2021. Due to a lack of knowledge on the

[2] According to MATLAB documentation, a serial date number represents the whole and fractional number of days counted from a fixed date in the proleptic ISO calendar.

Feature	**Unit**	**Mean**	**Standard Deviation**	**Mean of log values**	**Standard Deviation of log values**	**Range**
Arrival Time	datetime	-	-	-	-	-
Initial SOC	%	35	35.36	3.20	1.27	[10, 60]
Desired SOC	%	80	28.28	4.35	0.36	[60, 100]
Battery Capacity	kWh	66.42	26.41	4.12	0.41	[17, 200]
Charge Duration	min	95	120.21	3.74	2.04	[10, 180]
Maximum Power	kW	92.25	53.56	4.36	0.59	[19, 230]

Table 3: Main statistics used in the generation of random features

actual frequencies of arrival, we do not look into different distributions for this feature and we leave this point open for further discussion.

The desired number of observations is set to 10000. Once a distribution is selected for each feature, a matrix of observations with the number of observations as rows and number of features as columns is formed.

3.2 PCA analysis

The Principal Component Analysis (PCA) is used as a first approach to estimate the number of maximum components of GMM. A more robust method will be developed later in the discussion as this approach would fail in case of low dimensional matrix of features as the maximum number of components would be incorrectly limited to the number of features itself. The main idea is to set the maximum number of GMM components to the number of principal components that cover the 99% variance of the matrix of features. To do so, eigenvalues and eigenvectors of the covariance matrix of normalized features are computed. In this way, the principal components are computed as new variables representing a linear combination of the initial variables. Interestingly, the deriving matrix of eigenvectors can be used to project the original matrix into a new orthogonal basis where the direction of the data explains the maximal amount of variance. In this experiment, we are not interested in fitting a reduced matrix of features but rather to keep the data in its original basis and understand what is the minimum number of representative components.

3.3 Optimal GMM model based on *fitgmdist*

In this section, a GMM-based model is developed starting from the built-in MATLAB function *fitgmdist*. This function embeds the iterative EM-algorithm illustrated previously and it yields a complete GMM-model with a given number of components as input parameter, together with the the matrix of observations. The main idea behind our implementation is to successively fit GMMs via *fitgmdist* while tuning the function's parameters to minimize the Akaike information criterion (AIC). Bayesian information criterion (BIC) have also been investigated. Typically, AIC is used for prediction as it favours "the best approximation of the true model", while BIC is used for selection, inference or interpretation as it always tries to find the "true model" in the candidate set. A combination of the two is often used to narrow down the set of candidates to only one or two models. Indeed, in our controlled experiment, we experienced the former case where AIC and BIC almost always agree. For the sake of completeness, their expression is reported here-under:

$$\begin{aligned} \mathrm{AIC} &= 2k - 2\log L \\ \mathrm{BIC} &= -2\log L + k\log n \end{aligned} \tag{7}$$

where k is the number of model parameters, L is the likelihood of the fitted GMM and n is the number of observations. Regarding the tuning parameters, Tab 4 illustrates the set *fitgmdist*'s allowed inputs, along with their range and a small description.

Table 4

Parameter	**Values**	**Description**
CovarianceType	'full' / 'diagonal'	Type of covariance matrix to fit to the data
ProbabilityTolerance	[0,1e-6]	Tolerance for posterior probabilities. In each iteration, after the estimation of posterior probabilities, fitgmdist sets any posterior probability that is not larger than the tolerance value to zero. Using a nonzero tolerance might speed up fitgmdist.
RegularizationValue	nonnegative scalar	Regularization parameter value. Set RegularizationValue to a small positive scalar to ensure that the estimated covariance matrices are positive definite.
Replicates	positive integer	Number of times to repeat the EM algorithm using a new set of initial values
SharedCovariance	true/false	Flag indicating whether all covariance matrices are identical
Start	'randSample' / 'plus'	Initial value setting method

Comments:

- The covariance matrix can take one of the following form based on *CovarianceType* and *SharedCovariance* parameters value:
  - M-by-M-by-K array: full, not shared
  - 1-by-M-by-K array: diagonal, not shared
  - M-by-M matrix: full, shared
  - 1-by-M vector: diagonal, shared
- EM-algorithm is sensible to the initial operating point. In *fitgmdist*, the component weights are always uniform. The starting covariance matrices for all components are diagonal with elements on the diagonal equal to the variance of each feature. By setting *Start* to 'randSample', M observations from the vector of observations are selected at random as initial component means. If *Start* is 'plus', the k-means++ algorithm is employed to estimate the initial component means.

Furthermore, to include the optimization over the number of GMM's components, another loop is included where M ranges from 1 to the value defined from the PCA analysis. In the next Section, this approach will be modified to tackle the problem of over-fitting, which is very often the case for models with very high number of components.

3.4 Sampling process

The sampling process is extremely important as it enables to draw random value from the fitted GMM-distributions. The main complexity relies in the fact that realistic bounds needs to be ensured during the process. In this context, a simple accept-reject policy has been followed, namely we keep the samples within bounds and reject the ones that are not in bounds. The result will be as the fitted GMM is truncated over different ranges for each feature.

3.5 Fit sampled data to original distribution

By using the original distribution with which each feature is generated, the samples are fitted via MATLAB function *fitdist*.

3.6 Comparison of initial and final distribution

The newly fitted distributions are then compared to the original data-generating distributions across all features both visually with Histograms and QQ-Plots and numerically. Regarding the numerical assessment, the Two-sample Kolmogorov-Smirnov test (KSIPer2) is used to compare each pair of distribution in terms of goodness-of-fit. To perform this test, a number of samples was randomly drawn (with truncation) from the original distribution and the final one.

Generally, a Two-sample Kolmogorov-Smirnov test determines if independent random samples, are drawn from the same underlying continuous population. H indicates the result of the hypothesis test:

- H = 0 => Do not reject the null hypothesis at the 5% significance level
- H = 1 => Reject the null hypothesis at the 5% significance level.

In addition, we use the Mean Absolute Percentage Error (MAPE) and the Mean Percentage Error (MPE) as they are classically employed in the performance evaluation of forecasting methods. This two metrics will be defined later in Section GMM-based Forecasting MATLAB Toolbox. In this scenario, they are mainly used to describe the distance between original and final distribution.

3.7 Results

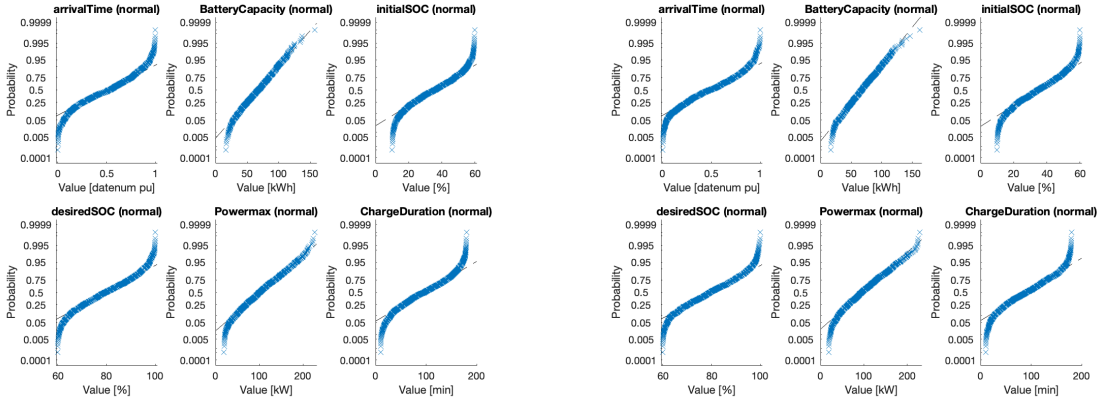
In this section we show some of the results of our numerical simulations. The goal is to draw conclusions and intuitions that will become useful for the development of a complete MATLAB Toolbox for forecasting EV flexibility.

First of all, we define two scenarios for our analysis:

1. All features are generated from *Normal* distributions
2. Mixture of distribution are used to generate features

Fig. 2 reports the QQ-Plots for the synthetic generated data with truncated distributions in the two scenarios. Clearly, the truncation process has a significant impact on the shape of the distribution. This is motivated by the fact that the scatter plot of probabilities in QQ-Plots does not follow perfectly the inter-quantile line, especially at the extreme ends. In this context, truncation comes with an additional complexity in our controlled experiment which is nevertheless fundamental to simulate features that are compatible with reality.

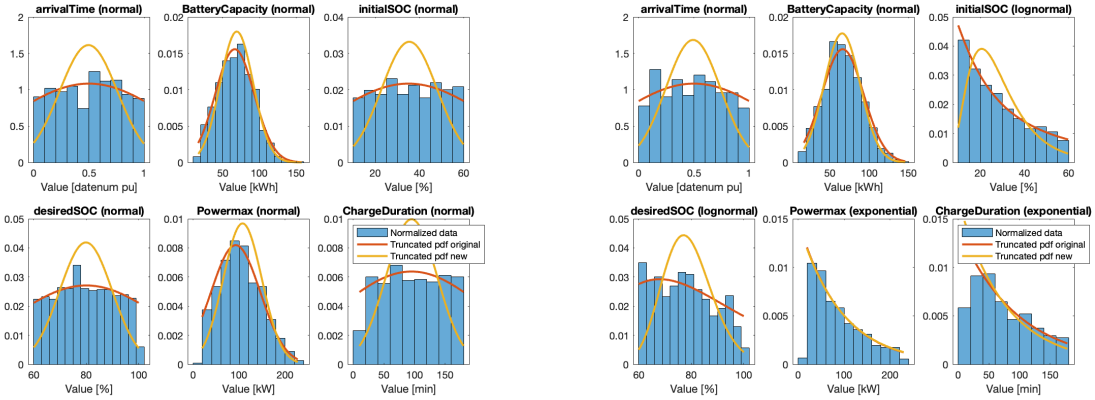
After creating the new distributions by fitting the matrix of samples obtained from the GMM, Fig. 3 is created. In particular, this figure reports histogram of the normalized synthetic generated data for the two scenarios (Normal distributions and different Distributions). The truncated probability density functions are represented with the red line. In addition, the orange line represents the truncated probability density function of the new distributions, namely obtained from fitting the samples from the resulting GMM. In this way, we can graphically compare the two distribution and the fitting process



(a) Normal Distributions

(b) Different Distributions

Figure 2: QQ-Plots for synthetic generated features



(a) Normal Distributions

(b) Different Distributions

Figure 3: Histograms - Fitting process on the original data with the newly generated distributions

against the original set of data. The closer the red and orange line are, the better the GMM-fitting process will be. Clearly, the most "challenging" features are the ones with the smallest range of truncation. For instance, *Arrival Time* seems to be fitted quite poorly as the values were normalised in a range from 0 to 1. The normalization is needed as the datenum format is typically characterized by very large numbers, thus rendering the fitting even more complex.

All things considered, the results are encouraging for some of the randomly generate features such as *Battery Capacity* and *Maximum Power* for both scenarios and *Charge Duration* for the mixed distribution scenario. Conversely, the largest fitting error is for *Initial SOC* and *Desired SOC* in both scenarios and *Charge Duration* when considering only normal distributions.

To numerically test the fitting process, the metrics in Tab. 5 and Tab 6 are reported. As expected, the numerical results agree with our previous visual inspection. Interestingly, the highest error is registered for *Arrival Time* with 37.39% and 28.12% for respectively for MAPE and MPE in the second scenario. In the first scenario, *Arrival Time* is also the worst, followed by *Charge Duration* with a MAPE of 17.91% and MPE of 10.59%. Regarding the KSIPer2 test, in the simplest the null hypothesis is rejected at the 5% significance level for all features, while for the second scenario it is rejected only for *Arrival Time*, *Initial SOC* and *Desired SOC*. On a more general note, both MAPE and MPE seem to get worse for cases where the range of truncation is particularly strict, while pretty small percentage errors are identified for features defined on a larger domain.

In conclusion, we can see how the developed model needs improvement on several aspects among which the *Arrival Time* handling, the testing framework as well as the overall capability of the GMM

	KSIPer2	**MAPE [%]**	**MPE [%]**
Arrival Time	1	29.74	20.08
Battery Capacity	1	6.80	6.49
Initial SOC	1	11.79	4.90
Desired Capacity	1	3.79	1.47
Maximum Power	1	10.51	8.63
Charge Duration	1	17.91	10.59

Table 5: Performance metrics - Normal distributions for all features

	KSIPer2	**MAPE [%]**	**MPE [%]**
Arrival Time	1	37.39	28.12
Battery Capacity	0	3.71	2.47
Initial SOC	1	13.62	6.22
Desired Capacity	1	3.08	0.27
Maximum Power	0	2.08	-0.74
Charge Duration	0	7.14	-7.14

Table 6: Performance metrics - Different Distributions for all features

to describe a large set of features with only one fitting process. Moreover, an important point to investigate is how to estimate the maximum number of GMM components. All these points will be tackled and taken into account in the the section where the complete MATLAB Toolbox for EV flexiblity is developed.

4 GMM-based Forecasting MATLAB Toolbox

A MATLAB Toolbox was developed starting from the information learned during our controlled experiment. The ultimate goal of this tool is to to statistically characterize historical data relative to EV charging sessions via GMM. The expected matrix of features has observations as rows and features as columns. Particularly, the first six columns are required to be the arrival time (datetime format) of the EV user. Apart from this, the function do not have constraints on the number or type of features included as successive columns. As an output, the function will simply provide the best GMM-distribution / set of GMM-distributions representing the original data.

More specifically, the main body of the code is characterized by the following steps:

1. Pre-select observations based on the *Arrival Time*
2. Apply Multivariate-GMM approach
3. Apply Univariate-GMM approach
4. Apply Mixed-GMM approach
5. Select the best approach based on the fitting performance

In Fig. 4, a Flowchart summarises the steps and their interconnection. In this Flowchart, the three different approaches for the fitting process are are highlighted in red as a pre-defined process. The first one (Multivariate-GMM approach) uses the entire matrix of features. Conversely, the second approach (Univariate-GMM approach) fits individually each feature and yields a sequence of GMM distributions. Finally, the third approach (Mixed-GMM approach) combines the two previous methods. Clearly, the

first approach is indicated in cases in which all features are correlated while the second approach tackles more efficiently cases in which there is a weak correlation. Ideally, we would like to combine the two approaches so that the best possible output is achieved, namely we fit correlated features with Multivariate-GMM approach and uncorrelated features by Univariate-GMM approach.

The Toolbox is implemented in the function *fit_gmm_best_model.m*. The inputs are:

- The matrix *X* of features with observations in rows and features in columns. The first 6 columns are required to be a datetime format.
- The structure *config* with configuration parameters for the fitting process. To be set in the function *set_config.m*. In the configuration structure, few parameters need to be set before launching the script:
  - *select_observations*: configuration for pre-selection of the observations
  - *MaxGMMComponents*: maximum number of GMM components allowed (default 10)
  - *KFold*: Cross-validation parameter (default 10)
  - *weights_FE*: weights for accuracy, bias and correlation in the selection of the best GMM model

Finally, the main output is the structure *best_model* featuring the following fields:

- A cell structure *gmm_best* containing a GMM distribution / set of GMM distributions
- The structure *FE* containing accuracy, bias, correlation and p-Value test for each feature
- A vector *idx_corr* reporting the indexes of correlated features in case of Mixed-GMM approach (optional)

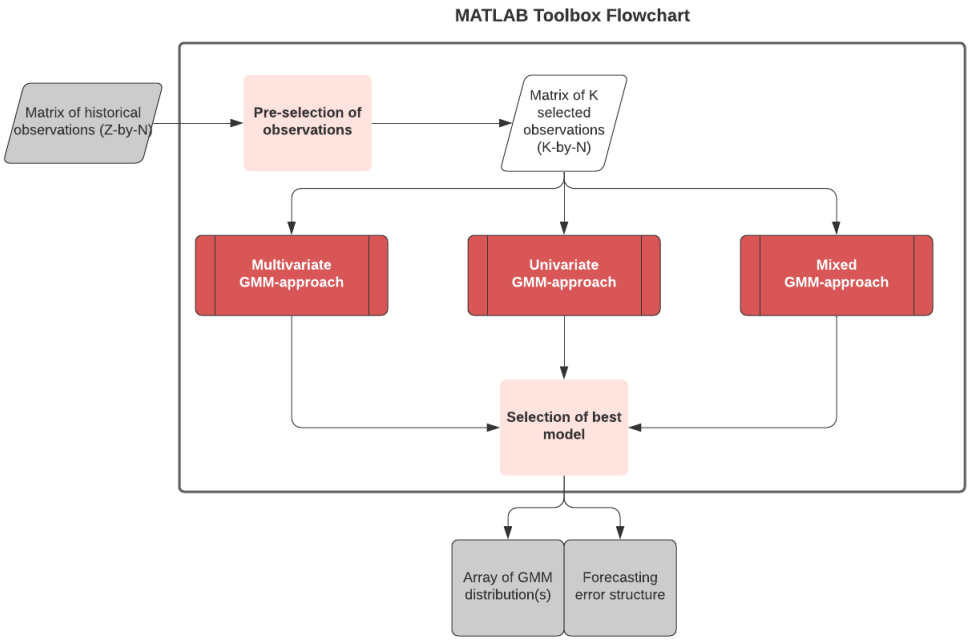


Figure 4: MATLAB Toolbox Flowchart

4.1 Pre-selection of observations

Given the poor results in the controlled experiment for *Arrival Time* forecasting, an alternative approach for handling this feature was evaluated. The importance to tie our newly generated data with the time of arrival motivates a pre-selection of the observations. To this end, the input matrix of features is automatically filtered based on a set of control parameters specified by the user, namely the year, the month, the type of day (weekday / weekday), the day and the hour. For instance, the user can decide to create scenarios on EV flexibility for a specific type or day and the tool will automatically provide the best model that is statistically representing of a subset of the historical data where only that given type of day is included.

In terms of software implementation, the main functions to refer to are *set_config_select_observations.m* and *select_observations.m*. Their description is omitted from the discussion.

4.2 Multivariate-GMM approach

The Optimal GMM model based on *fitgmdsit* developed in Section 3.3 was improved and extended to include K-fold Cross-Validation. The latter technique is typically employed to evaluate machine learning models on a restricted data-set where over-fitting might occur. However, we believe that it can be also efficiently employed to avoid the GMM over-fitting in case of a very high number of GMM components. In our application, Cross-Validation consists in randomly dividing the dataset in K groups or fold of the same size. Then, the training and test set are identified for the hold-out cross-validation of each partition K. At this point, the fitting process is launched on the training data-set, along with the optimization of *fitgmdist* parameters. Finally, a sample of data is drawn from the resulting temporary GMM distribution and the mean absolute error between the newly generated data and the test set is computed. Every fitted GMM-distribution is saved during the iterations. At the loop exit, the model with the lowest mean absolute error is chosen. The implementation is illustrated in the Flowchart of Fig. 9 in the Appendix.

The Multivariate GMM-approach uses the aforementioned function to fit the entire matrix of observations. In the same way as the controlled experiment, sampling-in-bounds is launched and the Forecast Error structure is computed. A graphical representation of the entire approach is reported in Fig. 10 in the Appendix.

Finally, the MATLAB function relative to this Section is *gmm_multivariate.m*. Furthermore, inputs and outputs are the same as *fit_gmm_best_model.m*. Particularly, *gmm_best* is the cell structure containing only one cell with the fitted GMM distribution.

4.3 Univariate-GMM approach

In opposition to Multivariate-GMM approach, Univariate-GMM approach is optimal when the features are not correlated. In the results part, we will validate this intuitions with a number of numerical experiments. Interestingly, this approach is equivalent to repeating the Multivariate-GMM approach when the input matrix is only composed by one feature at a time. The function to refer to is *gmm_univariate.m* where inputs and outputs are the same as *gmm_best*. However, the cell structure *gmm_best* will contain a number of GMM distributions equal to the number of features. The full Flowchart of the Univariate GMM-approach is reported in the Appendix (Fig. 11).

4.4 Mixed GMM-approach

The Mixed GMM-approach was developed to combine the two previous methods as it might be beneficial to fit correlated data with a Multivariate GMM-approach, while uncorrelated features with a Univariate GMM-approach. This is motivated by the intuition that Multivariate GMM is able to capture the interconnections between correlated features and eventually increase the accuracy of the forecasting. To determine if the features are correlated or not, a correlation analysis based Pearson's Linear Correlation Coefficient is carried out. In our application, the coefficient of correlation $R(i,j)$ between two generic features i and j of the matrix of features X is expressed as:

$$R(i,j) = \frac{\sum_{k=1}^{K}(X_{i,k}-\bar{X_i})(X_{j,k}-\bar{X_j})}{\left\{\sum_{k=1}^{K}(X_{i,k}-\bar{X_i})^2 \sum_{k=1}^{K}(X_{j,k}-\bar{X_j})^2\right\}^{1/2}} \tag{8}$$

where $\bar{X_i}$ and $\bar{Y_j}$ are the mean value of respectively column i and j in X. The matrix R is obtained by applying (8) to each pair of columns in the matrix of features matrix X. Thereafter, X is split in two new matrices, namely one for correlated features and the second one for uncorrelated. The accepted tolerance for two features to be correlated is configurable in *config*. Its default value is 0.5 where -1 is negative correlation and +1 is positively correlated. Particularly, the absolute value of R is taken when checking the correlation condition so that negative correlations are not excluded. Once we have split the matrix of features in two, we can separately apply the Multivariate GMM-approach to the matrix of correlated features and the Univariate GMM-approach to the uncorrelated matrix of features. In Appendix (Fig. 12) a Flowchart on the approach structure can be found.

4.5 Select best GMM approach

A comprehensive set of metrics is developed to compare the three different approaches and ultimately decide which one is the best for our forecasting purposes. The framework of metrics ultimately aims at testing each method in terms of:

- Accuracy: It represents the average discrepancy between individual pairs of observation and forecast
- Bias: It measures the mean deviation from average observation and average forecast. A negative sign indicates under-estimation and viceversa.
- Correlation: It measures the correlation between the vector of observation and forecast via Pearson's Linear Correlation Coefficient
- Goodness-of-Fit: P-value test on the null hypothesis of having the same underlying distribution for the vector of observations and forecast. Two-sample Kolmogorov-Smirnov test (KSIPer2) is used.

Several different metrics are chosen for Accuracy and Bias, while for Correlation and GOF we simply use the ones described previously. Tab 7 reports the whole set and relative formula.

Metrics such as MSE, RMSE, MAE, MAPE, ME, MPE are typically used for estimating the performance of a forecasting tool. However, they are sometimes misleading and difficult to interpret. For instance, MAPE is not defined when the true value is zero, it does not handle asymmetry in the forecast (over-forecasting and under-forecasting) and it is not resistant to outliers. To counter these drawbacks, we propose to include more robust, symmetric and resistant metrics such as MSA and SSPB [20].

Table 7: Set of metrics considered for testing the forecasting Accuracy and Bias

	Metric	**Unit**	**Formula**
Accuracy	Mean square error (MSE)	[different scale]	$\frac{1}{K}\sum_{k=1}^{K}\epsilon_k^2$
	Root mean square error (RMSE)	[real unit]	$\sqrt{MSE}$
	Mean absolute error (MAE)	[real unit]	$\frac{1}{K}\sum_{k=1}^{K}\lvert\epsilon_k\rvert$
	Median absolute error (MdAE)	[real unit]	$M(\lvert\epsilon_k\rvert)$
	Mean absolute percentage error (MAPE)	[%]	$\frac{100}{K}\sum_{k=1}^{K}\left\lvert\frac{\epsilon_k}{x_k}\right\rvert$
	Symmetric mean absolute percentage error (sMAPE)	[%]	$\frac{100}{K}\sum_{k=1}^{K}\left\lvert\frac{\epsilon_k}{(x_k+y_k)/2}\right\rvert$
	Median symmetric accuracy (MSA)	[%]	$100(\exp\{M(\lvert\log_e Q_i\rvert)\}-1)$
Bias	Mean error (ME)	[real unit]	$\frac{1}{K}\sum_{k=1}^{K}\epsilon_k$
	Mean percentage error (MPE)	[%]	$\frac{100}{K}\sum_{k=1}^{K}\frac{\epsilon_k}{x_k}$
	Median log accuracy ratio (MdLQ)	[different scale]	$M(\log_e Q_k)$
	Symmetric signed percentage bias (SSPB)	[%]	$100\text{sign}(MdLQ)(\exp\{\lvert MdLQ\rvert\}-1)$

Legend:

- x denotes the K-dimensional observation vector and y the K-dimensional forecast vector
- $\epsilon = y - x$ is the forecast error and $Q = y/x$ is the ratio between forecast and observation

MSA is an improved version of MAPE that takes into account of eventual asymmetry in the error distribution. In other words, MSA is the typical unsigned percentage error between forecast and observation. sMAPE can also be used to assess the forecasting accuracy as it is considered robust and resistant. However, it is defined on a 200% scale, therefore it is not as easy as to interpret as MSA.

SSPB has a similar meaning as MPE but it is not affected by the asymmetry in the MPE distribution. Indeed, this new metric penalizes over and under predictions in the same way. Moreover, the sign function will give the direction of the Bias.

The final selection of the model is based on the computation of a the global forecasting error (FE). The value of FE is designed to be a weighted sum of percentage errors in the class accuracy (A), namely MAPE, sMAPE, MSA, bias (B), namely MPE, SSPB, and correlation (R). In particular, FE is used as an indicator of the global capability of the different approaches.

$$FE = w_1 * A + w_2 * B + w_3 * R \tag{9}$$

A, B and R are computed by taking the mean of absolute values across all features and percentage

metrics in that class.

$$
\begin{aligned}
A &= \frac{1}{N}\sum_{n=1}^{N}\frac{|MAPE_n| + |sMAPE_n| + |MSA_n|}{3} \\
B &= \frac{1}{N}\sum_{n=1}^{N}\frac{|MPE_n| + |SSPB_n|}{2} \\
R &= \frac{1}{N}\sum_{n=1}^{N}|100 * (1 - R_n)|
\end{aligned}
\tag{10}
$$

The weights need to be assigned by the user based on the application requirement. A default value of 0.33 for each is set. Note that the metrics employed are percentages. The absolute value is taken as we are not interested in evaluating the direction of bias for under or over-estimation. Likewise, we are not interested in evaluating if the correlation is positive or negative.

Finally, the best model is selected by taking the one characterized by smallest Forecasting Error FE, which jointly minimize the the average mean error on accuracy, bias and correlation.

5 Results on GoFast Dataset

GoFast has provided us with a real-world data set of 1841 observations relative to EV charging sessions in Switzerland between 01/12/2020 and 30/04/2021. For each observation, we have information on:

1. *Arrival Time*: datetime format
2. *Location Type*: string where the values can be TOP (most frequently used EV charging station), BOTTOM (least frequently used EV charging stations), or EPFL (EV charging station at EPFL)
3. *Plug Type*: string where the values can be: AC Plug, AC Socket, CHADEMO CCS, DC Typ2
4. *Energy*: average energy in kWh that is delivered by the plug to the car during the charging session
5. *Charge Duration*: interval of time between arrival time and the moment at which the car is unplugged. Unit is minutes.

We forecast *Charge Duration* and *Energy* as they are the closest features to the one we are actually interested in forecasting. For the first numerical experiment, we analyse the entire data-set without filtrating by *Location Type*. We also select only weekdays as a day type so that the forecasting will be relative of a generic weekday in the operations of an EVCS in Switzerland. We do not focus on the probabilistic characterization of the frequency of arrival as that is left for further studies. Instead we test the performance of our MATLAB Toolbox by creating histogram with the superimposed probability density function of the best fitted GMM model on the original set of historical data. Thereafter, we present the numerical results under our standardized testing framework.

Fig. 5 shows an overall very good fitting for the two features under analysis. The best model chosen is the Univariate-approach where the number of components are 10 and 7 for respectively first and second GMM distributions. Numerical results for the forecasting error are reported in Tab. 8. Regarding the charge duration, the highest error is registered for MSE (6.28%) and MPE (2.53%) respectively in terms of accuracy and bias. As for energy, the largest error is given by 9.05% MAPE and 7.80% MPE respectively for accuracy and bias. The correlation between observation vector and forecast is extremely high (almost 1) for both features. In addition, the KSIPer2 test confirms that we cannot reject the null hypothesis of the two vectors being drawn from the same population at the 5% significance level.

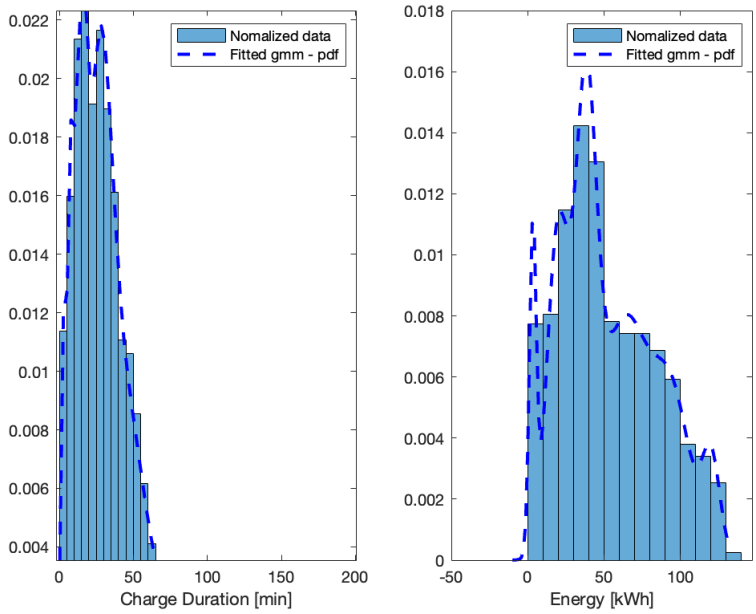


Figure 5: Best GMM-model - Weekdays EV sessions in Switzerland - GoFast dataset

Metric	***Charge Duration [min]***	***Energy [kWh]***
MSE [ds]	6.28	0.75
RMSE [ru]	2.51	0.87
MAE [ru]	0.74	0.66
MdAE [ru]	0.40	0.53
MAPE [%]	4.22	9.05
sMAPE [%]	0.94	0.87
MSA [%]	1.66	1.23
ME [ru]	0.07	0.10
MPE [%]	2.53	7.80
MdLQ [ds]	0.00	0.00
SSPB [%]	-0.02	0.04
R	0.9938	0.9996
KSIPer2	0	0

Table 8: Forecast error evaluation - Weekdays EV sessions in Switzerland - GoFast dataset

In the second numerical experiment, we test our algorithm for the EVCS at EPFL for weekdays EV charging sessions. The filtered data-set only includes 282 observations. Better performance are expected with a larger number of observations on EV charging sessions. Clearly, Fig. 6 and Tab. 9 show a worse fitting compared to first experiment probably given by the smaller number of observations as an input. In this case, our Toolbox yields a Multivariate model with 8 components as the best one. Although MAPE and MPE seem to be relatively high (up to 76.68% for MAPE and 69.28% for MPE on *Energy*), these metrics are not accurate as they are sensible to very small forecast values (denominator close to 0). Hence, we also need to look at more resistant and robust percentage metrics such as MSA and SSPB. Overall, MSA and SSPB are never higher than 10-15% for both features, which is extremely good. This promising result is further confirmed by the high correlation between forecast and observations. Finally, KSIPer2 is also equal to 0, namely we cannot reject the null hypothesis at the 5% significance level.

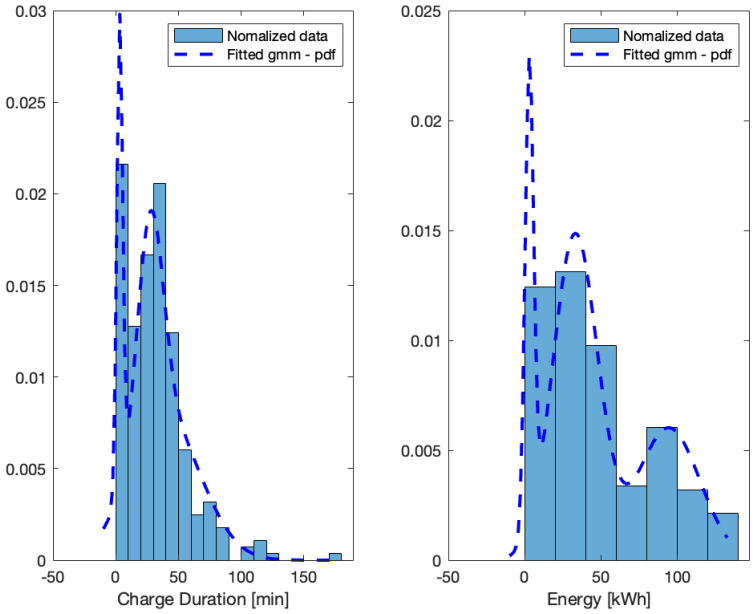


Figure 6: Best GMM-model - Weekdays EV sessions at EPFL - GoFast dataset

Metric	***Charge Duration [min]***	***Energy [kWh]***
MSE [ds]	17.07	6.46
RMSE [ru]	4.13	2.54
MAE [ru]	3.22	1.99
MdAE [ru]	2.84	1.41
MAPE [%]	22.97	76.68
sMAPE [%]	4.51	3.49
MSA [%]	12.05	4.53
ME [ru]	2.58	-1.41
MPE [%]	22.46	69.28
MdLQ [ds]	0.11	-0.03
SSPB [%]	11.93	-3.36
R	0.9921	0.9989
KSIPer2	0	0

Table 9: Forecast error evaluation - Weekdays EV sessions at EPFL - GoFast dataset

6 Limitations and Future Studies

The limitations of our Toolbox are mostly related to the difficulty of forecasting Arrival Time. On one hand, the alternative approach of pre-selecting the observations solves the problem of tying the forecast features to a given targeted day. On the other hand, do not explicitly forecasting Arrival Time means that we are not able to predict how many cars will arrive in the targeted day and, obviously, at what time they will arrive. Although the Toolbox is capable of forecasting Arrival Times with the serial normalized number approach, the forecast is typically low-quality and the sampled values do not respect the real Frequency of Arrival. An experiment was carried out where the entire GoFast data-set is used and Frequency of Arrivals are extracted from both original and sampled matrices (number of samples is the same as number of observations). The result is shown in Fig. 7 where a dramatic discrepancy between the two parts can be seen.

To counter this problem, the pre-selection approach can be combined with the following proposed two-step procedure, namely predicting the number of EV charging sessions per day and then estimating

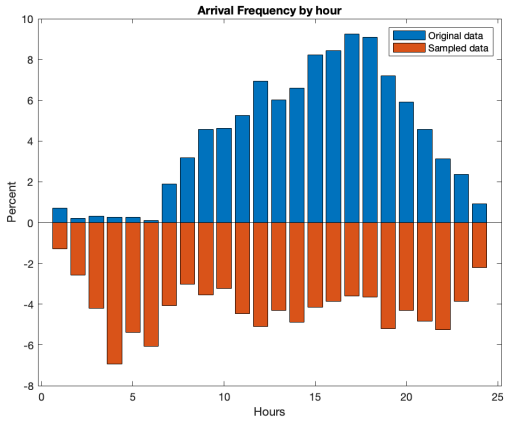


Figure 7: Comparison of original and forecast frequency of arrival - All set of EV sessions in Switzerland - GoFast dataset

the Arrival Time for each EV. With this procedure, we do not have to assess explicitly the Frequency of Arrival. Instead, we focus on the average number of EVs charging for the targeted day, then we compute the Arrival Time from its empirical cumulative distribution function ($ecdf$).

The first step is the estimation of the mean number of EV charging sessions per day, which is done by:

1. Pre-selecting observations based on targeted day
2. Group count the number of EV charging sessions per day
3. Creating truncated normal distribution around mean and std of this vector
4. Sampling one value from this distribution which will correspond to the the number of session in one day

Clearly, this number of EV sessions will be relative to the targeted days as the matrix of observations is filtered at the beginning of the procedure.

As mentioned, the second step is the Arrival Time prediction. Our proposed procedure reads:

1. Random draw a vector (between 0 and 1) of dimension equal to the number of EV charging session
2. Compute $F(x) = ecdf(x)$ where x is the normalized-to-unity serial number of Arrival times
3. Get $x = iecdf(Q)$ where iecdf is the inverse of $ecdf$

In Fig. 8, it can be seen how the Arrival Times are extracted from $iecdf$. Noticeably, the values of x need to be transformed back to real units (they were normalized), then converted in datetime format.

In conclusion, the forecast on features is based on the pre-selection of the historical data relative to the desired period of time. In this way, the generated features will be statistically representative without increasing the complexity of the model. At the same time, the number of features in the input matrix is reduced, thus also improving the computational efficiency. To determine the number of EV sessions per day of one particular EVCS, we propose the aforementioned two-step methodology. A combination of the MATLAB Toolbox and the two-step procedure is proposed to fully capture the probabilistic nature of EV-flexblity.

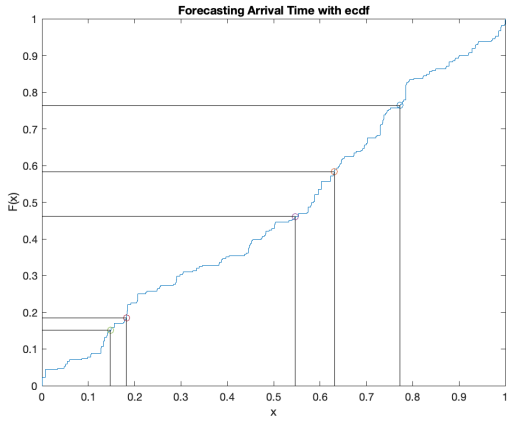


Figure 8: Arrival Time Prediction with ecdf-based approach

As a last limiting point, we acknowledge that the sampling process does not take fully into account of the technical operational constraints of the EVCS, hence the forecast observations will be sometimes unfeasible. For instance, the number of plugs available and the maximum cumulative power capability of the EV charging station might be unrealistic are two extremely important constraints to be respected. Most likely, the forecaster tool will capture the probabilistic distribution of each feature but it will not be able to provide feasible forecast of EV charging sessions.

7 Conclusion

In conclusion, a controlled experiment was designed to investigate the global capability of GMM in the framework of EV flexibility forecasting. After insight was gathered on the parameters and applicability of this forecasting model, a complete MATLAB Toolbox was designed and implemented. The model has been extensively described both in terms of requirements of the input matrix of observations and in terms of assumptions for every single design choice. Finally, the MATLAB Toolbox was evaluated for a small data-set of EV charging sessions provided by GoFast.

Appendix

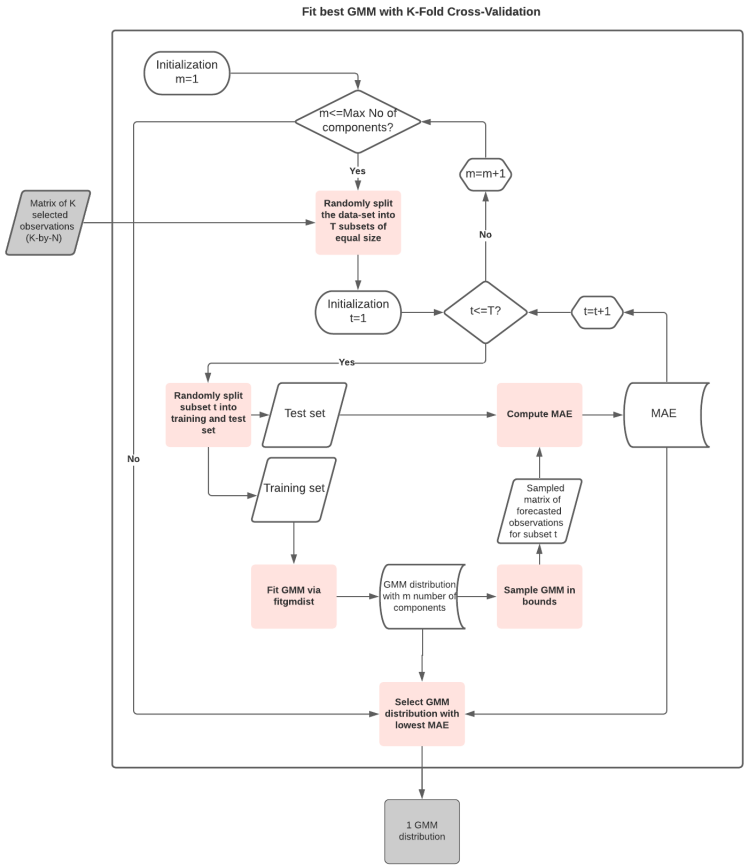


Figure 9: Fit best GMM with K-Fold Cross-Validation Flowchart

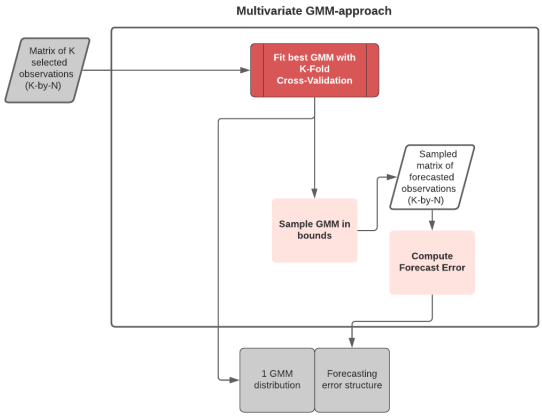


Figure 10: Multivariate GMM-approach Flowchart

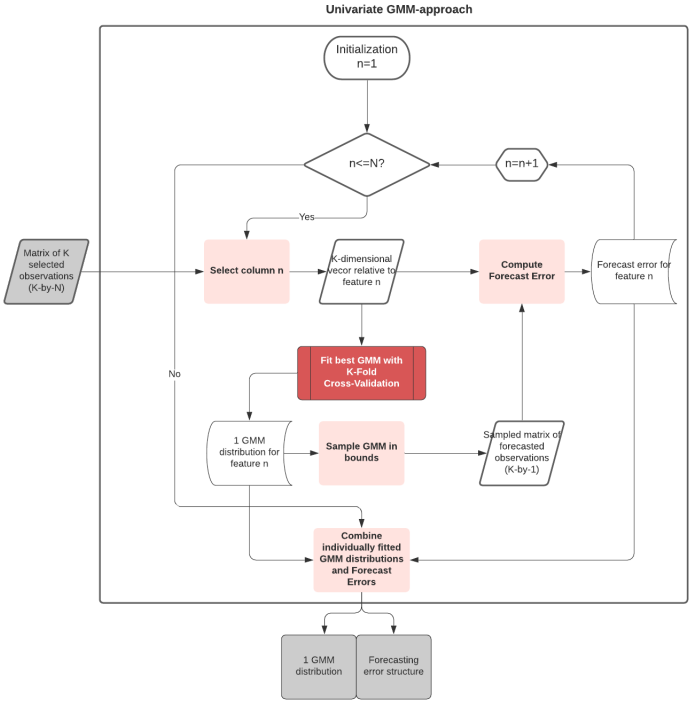


Figure 11: Univariate GMM-approach Flowchart

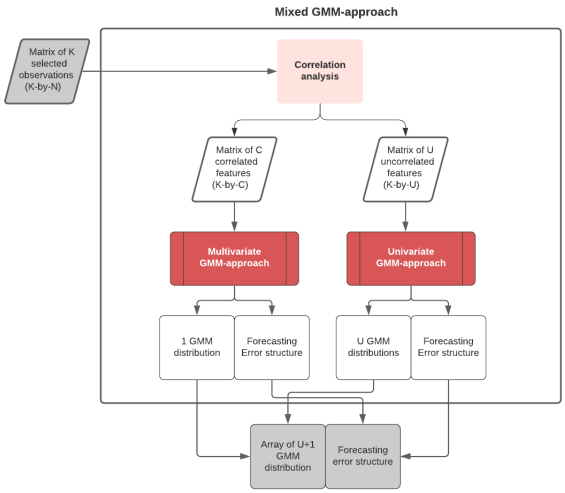


Figure 12: Mixed GMM-approach Flowchart

References

[1] Meiqin Mao, You Yue, and Liuchen Chang. "Multi-time scale forecast for schedulable capacity of Electric Vehicle fleets using big data analysis". In: *2016 IEEE 7th International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*. 2016, pp. 1–7. DOI: 10.1109/PEDG.2016.7527051.

[2] E. S. Xydas et al. "Forecasting Electric Vehicle charging demand using Support Vector Machines". In: *2013 48th International Universities' Power Engineering Conference (UPEC)*. 2013, pp. 1–6. DOI: 10.1109/UPEC.2013.6714942.

[3] Marc Cañigueral and Joaquim Meléndez. "Flexibility management of electric vehicles based on user profiles: The Arnhem case study". In: *International Journal of Electrical Power Energy Systems* 133 (2021), p. 107195. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2021.107195. URL: https://www.sciencedirect.com/science/article/pii/S0142061521004348.

[4] Markus Gödde et al. "Modelling the charging probability of electric vehicles as a gaussian mixture model for a convolution based power flow analysis". In: *2015 IEEE Eindhoven PowerTech*. 2015, pp. 1–6. DOI: 10.1109/PTC.2015.7232376.

[5] Chuong. "The multivariate Gaussian distribution". In: 2004.

[6] A. Nait-Sidi-Moh et al. "A Prediction Model of Electric Vehicle Charging Requests". In: *Procedia Computer Science* 141 (2018). The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops, pp. 127–134. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.10.158. URL: https://www.sciencedirect.com/science/article/pii/S1877050918318088.

[7] Kejun Qian et al. "Modeling of Load Demand Due to EV Battery Charging in Distribution Systems". In: *IEEE Transactions on Power Systems* 26.2 (2011), pp. 802–810. DOI: 10.1109/TPWRS.2010.2057456.

[8] F. Koyanagi et al. "Monte Carlo simulation on the demand impact by quick chargers for electric vehicles". In: *1999 IEEE Power Engineering Society Summer Meeting. Conference Proceedings (Cat. No.99CH36364)*. Vol. 2. 1999, 1031–1036 vol.2. DOI: 10.1109/PESS.1999.787457.

[9] F. J. Soares, J. A. Peças Lopes, and P. M. Rocha Almeida. "A Monte Carlo method to evaluate electric vehicles impacts in distribution networks". In: *2010 IEEE Conference on Innovative Technologies for an Efficient and Reliable Electricity Supply*. 2010, pp. 365–372. DOI: 10.1109/CITRES.2010.5619777.

[10] Eric Sortomme et al. "Coordinated Charging of Plug-In Hybrid Electric Vehicles to Minimize Distribution System Losses". In: *IEEE Transactions on Smart Grid* 2.1 (2011), pp. 198–205. DOI: 10.1109/TSG.2010.2090913.

[11] D. Steen et al. "Assessment of Electric Vehicle Charging Scenarios Based on Demographical Data". In: *IEEE Transactions on Smart Grid* 3 (2012), pp. 1457–1468.

[12] Chris Develder et al. "Quantifying flexibility in EV charging as DR potential: Analysis of two real-world data sets". In: *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 2016, pp. 600–605. DOI: 10.1109/SmartGridComm.2016.7778827.

[13] Maigha and Mariesa L. Crow. "Electric Vehicle Scheduling Considering Co-optimized Customer and System Objectives". In: *IEEE Transactions on Sustainable Energy* 9.1 (2018), pp. 410–419. DOI: 10.1109/TSTE.2017.2737146.

[14] Yuancheng Zhao et al. “An Optimal Domestic Electric Vehicle Charging Strategy for Reducing Network Transmission Loss While Taking Seasonal Factors into Consideration”. In: *Applied Sciences* 8 (Jan. 2018), p. 191. DOI: 10.3390/app8020191.

[15] Gaizka Saldaña et al. “Analysis of the Current Electric Battery Models for Electric Vehicle Simulation”. In: *Energies* 12.14 (2019). ISSN: 1996-1073. DOI: 10.3390/en12142750. URL: https://www.mdpi.com/1996-1073/12/14/2750.

[16] Michel Zade et al. “Quantifying the Flexibility of Electric Vehicles in Germany and California—A Case Study”. In: *Energies* 13.21 (2020). ISSN: 1996-1073. DOI: 10.3390/en13215617. URL: https://www.mdpi.com/1996-1073/13/21/5617.

[17] Meiqin Mao et al. “Schedulable capacity forecasting for electric vehicles based on big data analysis”. In: *Journal of Modern Power Systems and Clean Energy* 7.6 (2019), pp. 1651–1662. DOI: 10.1007/s40565-019-00573-3.

[18] Dongchen Qin et al. “Modeling and Simulating a Battery for an Electric Vehicle Based on Modelica”. In: *Automotive Innovation* 2 (Aug. 2019). DOI: 10.1007/s42154-019-00066-0.

[19] Fully Charged - Official Data Partner. *Electric Vehicle Database*.

[20] Chao Chen, Jamie Twycross, and Jonathan Garibaldi. “A new accuracy measure based on bounded relative error for time series forecasting”. In: *PLOS ONE* 12 (Mar. 2017), pp. 1–23. DOI: 10.1371/journal.pone.0174202.