Pattern Sequence Based Forecasting

In 2008, Álvarez et al. presented a new algorithm [4] called Label-based Forecasting (LBF), which combines clustering and string matching to generate predictions for a given time series $\vec{s} \in \mathbb{R}^m$. It was later modified and renamed Pattern Sequence Based Forecasting (PSF) [1]. PSF first divides the time series into equal-sized segments $X = \{\vec{x_1}, \vec{x_2}, ..., \vec{x_n}\}, \vec{x_i} \in \mathbb{R}^c$, called cycles, whose length c should be equal to an existing periodic component of the series. In the original paper, which applied the PSF algorithm to a time series consisting of hourly observations, c was chosen to be 24, corresponding to one day. Afterwards, k-means clustering is applied to the sequence of vectors X to obtain a scalar sequence $L = \{l_1, l_2, ..., l_n\}$ of cluster labels. The hyperparameter k, determining the number of clusters, is chosen via a simple search within predefined upper and and lower bounds, where for each k, the k-means clustering algorithm is run and one or more cluster metrics are used to evaluate the resulting clustering. Afterwards, the k that resulted in the best cluster metric score is kept for all further clustering. While the first LBF algorithm used only the Silhouette index, the modified PSF computes a majority score based on three cluster evaluation metrics (Silhouette-, Dunn-, and Davies-Bouldin index). However, the paper on PSF [1] fails to showcase an improvement in prediction accuracy as measured by the mean relative error, as both the original LBF and PSF achieve the same error scores on the same datasets. Therefore, and to speed up computation, the *Python* implementation used in this thesis relies solely on the Silhouette index to find k.

To predict the next cycle $\vec{x}_{n+1}$ of the time series, the pattern that is made up by the previous W labels in L are considered, where W is called the window size. L is searched for all matching subsequences that exactly match this pattern, and the index of each single label following a match is retained. These indices are then used to retrieve the corresponding cycles in X, whereupon their average is computed to obtain $\vec{x}_{n+1}$. If more predictions are needed, the latest predicted cycle is appended to X and the algorithm is repeated. The following equations show how a prediction is computed:

$$P = L_{n-W}^{n} \tag{1}$$

$$E_P = \{i \mid L_{i-(W+1)}^{i-1} = P,\, i < n\} \tag{2}$$

$$\vec{x}_{n+1} = \frac{1}{|E_P|} \sum_{i \in E_P} X_i \tag{3}$$

Similar to finding k, W is also determined by searching a predefined interval of values. To this end, X is split in time into a training and a validation set, where the training set makes up 70 % of X. Thereafter, for each candidate W, a number of cycles equal to the size of the validation set are predicted and the mean absolute error is used to evaluate the prediction performance. The W resulting in the lowest prediction error is then kept for all further forecasts. In [1], 12-fold cross validation was used to determined W, with each fold corresponding to one month of data, whereas [4] used a form of leave-one-out cross validation, where the total absolute error between the 1 step ahead prediction of each cycle in X and its observed value was minimized. If no matching pattern of size W can be found, W is decremented and the search repeated. Figure 1 shows a diagram illustrating the PSF algorithm and highlights the prediction loop.

As evident in Figure 1, the data is normalized before applying PSF. In [1], each hour in a cycle is divided by the cycle mean, but reverting this operation would require knowing the mean of every future predicted cycle, which is not feasible in a real world prediction scenario. Instead, simple min-max normalization is applied:

$$\vec{s}_{norm} = \frac{\vec{s} - min(\vec{s})}{max(\vec{s}) - min(\vec{s})} \tag{4}$$

Where $\vec{s}_{norm}$ is the normalized series. If the minimum of $\vec{s}$ is negative, its absolute value is added to all values of $\vec{s}$ before normalizing. The min-max normalization approach was also taken by Bokde et al. [2], who wrote a software library, written in the R programming language, that implements a variant of PSF. The *Python* implementation used in this project is based on this library.

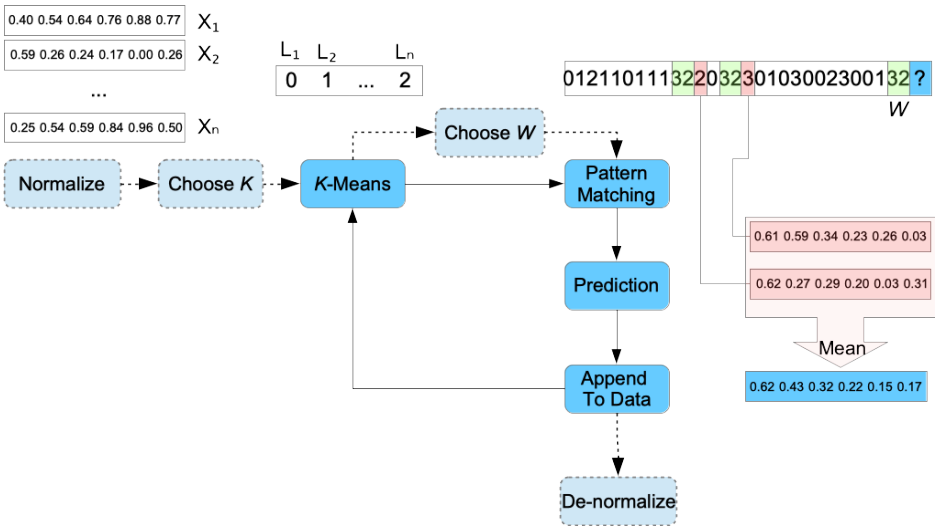

Figure 1: Diagram illustrating the PSF algorithm. Lighter shaded rectangles and dotted arrows represent operations that are only executed once, while the darker shaded rectangles and solid arrows represent the prediction loop, which is executed for each new prediction.

When predicting a cycle, it is possible that W reaches 0, e.g., when the previous cycle is an outlier and thus constitutes a cluster of size 1. This issue is discussed in neither [4] nor [1], but [2] introduces a fallback option that aims to solve this problem by simply using the last (outlying) cycle as the prediction. However, this method can have the undesirable effect that the same cycle is used for all predictions. Another approach could be to use the average of all preceding cycles as the prediction. While this would solve the aforementioned repetition issue, this prediction could also be sub-optimal, e.g., if the time series contains a few extraordinary cycles whose values are significantly larger or smaller than the average cycle. As these cycles are unusual, it might be desirable to discard them before computing a mean to use as the prediction. This can be achieved by using the centroid of the largest cluster as the predicted next cycle, an approach presented in [3], where Majidpour et al. present a modified PSF algorithm that combines PSF and elements from the k-nearest neighbors algorithm. A comparison of three separate PSF algorithms that only differed in the choice of fallback method, conducted as part of this thesis, showed that for the Danish weather parameter and electricity consumption time series, the algorithm employing the centroid based fallback method achieved the lowest mean absolute prediction error for electricity demand and in the large majority of weather parameter cases. Hence, it was picked as the fallback method for the *Python* implementation of PSF.

While both the original articles on LBF and PSF and multiple others mention a cycle length $c = 24$ for hourly electricity demand time series, trying to follow their example for the times series at hand yielded unusable predictions, which did not even follow the basic annual seasonal pattern and achieved a vastly lower forecast MAE than naive models calculating the average or predicting the previous year's data. This result was shared between the R implementation of PSF and the custom *Python* implementation written for this project. Inspecting the label series L revealed that the k in k-means was chosen to be only 2, leading to a binary labeling of the cycles. The lack of discrimination between cycles resulted in a feedback loop, where after a short amount of time, all matching patterns for future predictions were found in the section of L corresponding to previously predicted cycles. This manifested itself as step-like patterns in the prediction plots that would eventually converge to a straight horizontal line. In [3], Majidpour et al. encountered a similar problem and attempted to mitigate it by setting the lower bound of k to 10 % of distinct days in the data and the maximum k equal to the number of distinct days. However, it is unclear as to how this lower bound was chosen. In addition, for the data used in this thesis, this approach is practically infeasible when run on a laptop computer, as 2547 possible values of k reaching from 282 to 2829 would need to be considered, resulting in a running time of multiple hours for a single time series. Instead, a modification that finally led to sensible predictions was to set the cycle length c equal to 1 year, or 8760 hours when working with an hourly temporal resolution. Although, logically, this solution should reduce the algorithm's ability to capture finer details in the data, e.g., by clustering holidays and other high-consumption days together, it still

yielded competitive prediction accuracy when compared to the other approaches investigated in this project. Later, it was discovered that the most likely reason for the PSF algorithm failing initially to produce usable predictions when working on hourly data with $c = 24$ was the annual seasonality of the data. Simply computing the first order seasonal difference of the hourly electricity consumption data before applying PSF alleviated the issue.

References

[1] Francisco Martinez Alvarez, Alicia Troncoso, José C Riquelme, and Jesus S Aguilar Ruiz. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243, 2010.

[2] Neeraj Bokde, Gualberto Asencio-Cortés, Francisco Martínez-Álvarez, and Kishore Kulat. Psf: Introduction to r package for pattern sequence based forecasting algorithm. *arXiv preprint arXiv:1606.05492*, 2016.

[3] Mostafa Majidpour, Charlie Qiu, Peter Chu, Rajit Gadh, and Hemanshu R Pota. Modified pattern sequence-based forecasting for electric vehicle charging stations. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 710–715. IEEE, 2014.

[4] Francisco Martínez-Álvarez, Alicia Troncoso, José C Riquelme, and Jesús S Aguilar-Ruiz. Lbf: A labeled-based forecasting algorithm and its application to electricity price time series. In *2008 Eighth IEEE International Conference on Data Mining*, pages 453–461. IEEE, 2008.