MODIFIED PATTERN SEQUENCE-BASED FORECASTING FOR ELECTRIC VEHICLE CHARGING STATIONS

Contributors:

Mostafa Majidpour, Charlie Qiu, Peter Chu,and Rajit Gadh
Smart Grid Energy Research Center
University of California, Los Angeles, USA

Hemanshu R. Pota
School of Engineering & Information Technology
The University of NSW, Canberra ACT 2610 Australia

Acknowledgement

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000192.

Modified Pattern Sequence-based Forecasting for Electric Vehicle Charging Stations

Mostafa Majidpour, Charlie Qiu, Peter Chu, Rajit Gadh
Smart Grid Energy Research Center
UCLA
Los Angeles, California USA
mostafa@ee.ucla.edu

Hemanshu R. Pota
School of Engineering & Information Technology
The University of NSW
Canberra ACT 2610 Australia

Abstract—**Three algorithms for the forecasting of energy consumption at individual EV charging outlets have been applied to real world data from the UCLA campus. Out of these three algorithms, namely k-Nearest Neighbor (kNN), ARIMA, and Pattern Sequence Forecasting (PSF), kNN with k=1, was the best and PSF was the worst performing algorithm with respect to the SMAPE measure. The advantage of PSF is its increased robustness to noise by substituting the real valued time series with an integer valued one, and the advantage of NN is having the least SMAPE for our data. We propose a Modified PSF algorithm (MPSF) which is a combination of PSF and NN; it could be interpreted as NN on integer valued data or as PSF with considering only the most recent neighbor to produce the output. Some other shortcomings of PSF are also addressed in the MPSF. Results show that MPSF has improved the forecast performance.**

I. INTRODUCTION

Reducing the charging time of Electric Vehicles (EVs) (and Plug-in Hybrid Electric Vehicles (PHEVs)) is a big challenge. The minimum battery size for EVs has to be around 9 kWh according to the EV33 rule [3]. In practice, most EVs have larger batteries, e.g., 16.5 kWh for Chevrolet Volt, 24kWh for Nissan Leaf, or up to 85kWh for Tesla. Although there are fast DC chargers that can deliver up to 50kW (100A at 500VDC), the majority of charging stations are Level 1 household chargers, delivering 3.3 kW (16A at 230VAC), which makes the charging last around 8 hours for a Nissan Leaf. EV owners also can (and sometimes are obligated to) charge their vehicle in places other than home, such as public charging stations or charging stations at their work place. As of May 2014, there are 22,671 non-residential charging outlets in the US [1]. Although there has been a lot of research on charging station infrastructure [4]-[7], only 592 of the above mentioned stations are CHAdeMo fast DC chargers [2]. In this situation, an estimation of how long an EV owner should wait in order to charge the EV can be a very beneficial piece of information, especially if the expected waiting time can be accessed through the Internet or on a smartphone before leaving for the charging station. On the other hand, having access to this data will be useful for the EV charging station owners too, since it will help them to adjust their inventory in advance. Both of these requirements can be addressed by the energy consumption forecast at EV charging stations.

For the above mentioned reasons, forecasting EV loads is of recent interest to researchers. In [19], authors have proposed a method to forecast the EV charging load in China based on the Monte Carlo simulation. Reference [20] discusses three daily-load forecasting methods, namely BP and RBF Neural Networks, and GM(1,1) from the Gray model families on one charging station. Another method has been proposed in [21] based on Support Vector Regression for forecasting EV charging loads at the city level. Four forecasting methods including Decision Tables, Decision Trees, MPL Neural Networks, and Support Vector Machines (SVM) have been compared in [22] on the US aggregated residential data.

The work presented here is different from previously mentioned studies in that we have used just one type of recorded data, Charging Records, which only contains the start and end of the charging transaction and the total amount of energy received in the charging transaction (a scalar value; not time dependent). Geographical or driving habit related data was not used in our prediction. Our predictions are at the charging outlet level (not charging station, parking lot or city level) which makes it a more difficult problem as it does not have the aggregated behavior of charging stations, parking lots, or cities.

In our previous work [8], we have proposed a framework for fast prediction of the EV load at the charging outlet level for cellphone application. We found that the k-Nearest Neighbor (kNN) algorithm had a better performance. As a continuation of the previous work, in this paper we have compared kNN with two other algorithms: ARIMA as an example of the classical statistical method and Pattern Sequence-based Forecasting (PSF) as a recently proposed successful algorithm in energy price forecasting.

The rest of this paper is organized as follows: Section II formulates the problem, Section III briefly explains the kNN, ARIMA, and PSF methods. Section IV reports and analyzes the result of applying these algorithms on the University of California, Los Angeles (UCLA) parking structures' EV

This work has been sponsored in part by grant from the LADWP/DOE fund 20699 & 20686, Smart Grid Regional Demonstration Project.

charging data. Section V proposes a new algorithm, Modified PSF (MPSF), and presents its application to our data, and Section VI concludes the paper with presenting the direction for the future work.

II. Problem Formulation

The objective is to predict the energy demand in the next 24 hours at each charging outlet. The total energy available for all charging stations in a parking structure is constrained by the size of the distribution transformer. Formally, we assume there is some function relating future available energy and the past consumed energy:

$$\hat{E}(t) = f\big(E(t-i), \varepsilon(t)\big) \quad i \in \{1,2,\dots\} \qquad (1),$$

where $E(t)$ is the actual energy consumption at time t, $\hat{E}(t)$ is the prediction of the energy consumption at time t, $\varepsilon(t)$ is the set of all variables (such as noise) other than past energy consumption records, $E(t-i)$, that f might depend on.

As the usual practice in forecasting, we are interested to find an estimation of $E(t)$ according to some distance criteria. For the distance (error) measurement, we have chosen Symmetric Mean Absolute Percentage Error (SMAPE). For the day i, the SMAPE is defined as:

$$SAMPE(i) = \frac{1}{H}\sum_{t\in Day(i)} \frac{|E(t)-\hat{E}(t)|}{E(t)+\hat{E}(t)} \times 100, \qquad (2),$$

where $H$is the horizon of prediction in a given day. In this paper H is 24.

The last portion of the data (last 10% in this paper) is set aside as the test set to evaluate the performance of the algorithm. Thus, our goal is to find an algorithm that minimizes the error between actual value and its prediction on the test set.

We use the notation $\underline{\hat{E}}(t)$ as the vector of prediction for the next 24 hours ending at t (Fig 1. a). Also, $t_r = \{1,2,\dots,N_{tr}\}$ and $t_s = \{N_{tr}+1,\dots,N\}$ are the set of indexes for training and test sets respectively.

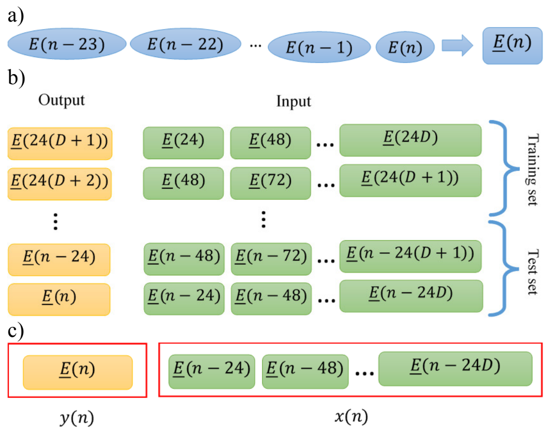

Figure 1. a) energy consumption vector (E) for 24 hours , b) input-output pairs and division of data into training and test sets, c) labeling inputs as x and outputs as y.

III. Applied Algorithms

The three prediction algorithms used in this paper have been described here briefly. A detailed description can be found in [3].

A. K-Nearest Neighbor

This algorithm is a well-known algorithm in the machine learning community [12] which previously has shown promising results on our data [8] and is used as a means of comparison in this paper.

Based on k-Nearest Neighbor (kNN) algorithm, each sample (training, test or validation) is composed of input and output pairs. In our application, the output is the energy consumption for the next 24 hours, $y(t) = \underline{E}(t)$, and the input is the concatenation of the consumption records for up to D prior days, $x(t) = \{\underline{E}(t-24), \underline{E}(t-48), \dots, \underline{E}(t-24D)\}$ (Fig. 1. c). This concatenation repeats for all days: if there are N days in the data set, there will be N-D+1 of these input-output pairs (Fig 1.b). The total number of data points is $n = 24N$. Now, in order to find an estimate for $y(t_{s^*})$ where $t_{s^*} \in t_s$ is an instance of test set indexes, first the distance between $x(t_{s^*})$ and all other $x(t_r)$ that belong to the training set is computed. Distance could be any norm of their difference; we have used the Euclidian distance here. After determining the k closest $x(t_r)$ to $x(t_{s^*})$, the average of their corresponding $y(t_r)$ is generated as $y(t_{s^*})$. In this algorithm, the parameter k needs to be determined. Fig. 2 illustrates the algorithm.

```
k-Nearest Neighbor Algorithm
Inputs: $x(t_r), y(t_r), x(t_{s^*}), k$
Output: $y(t_{s^*})$
1. for $j \in t_r$
2.     $dist[j] = \|x(t_{s^*}) - x(j)\|$
3. for $i \in \{1,\dots,k\}$
4.     $idx[i]$ = index of $i^{th}$ smallest$(dist)$
5. $y(t_{s^*}) = \frac{1}{k}\sum_{i\in\{1,\dots,k\}} y(idx[i])$
```

Figure 2. k-Nearest Neighbor Algorithm.

B. Auto Regressive Integrated Moving Average (ARIMA)

This model for time series behavior which is also called Box-Jenkins models [9] (because of Box and Jenkins' fundamental work in this area) models the behavior of the future variables as a linear combination of the past values and noise terms. The Auto Regressive (AR) portion models the contribution of the past values of the variable, while the Moving Average (MA) portion models the contribution of noise terms. The Integrated (I) portion models the number of differences needed in order to transform the time series to a stationary time series [13]. The ARIMA model is often specified by ARIMA(p,d,q); p, d and q are the order of the AR, I, and MA terms respectively. Mathematically ARIMA(p,d,q) for variable $X(t)$ can be written as:

$$\left(1-\sum_{i=1}^{p}\varphi_i L^i\right)(1-L)^d X(t) = \left(1-\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon(t) \qquad (3),$$

where L is the lag operator such that $LX(t) = X(t-1)$, $\varepsilon(t)$ is a representative of the noise (or shock or error) contribution, and φ, θ are the coefficients of the model that need to be determined. For our problem, the formula can be rewritten as following

$$\hat{E}(t) = (1-L)^{-d}\left(1-\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon(t) + \left(\sum_{i=1}^{p}\varphi_i L^i\right)E(t) \quad (4).$$

Notice that instead of forecasting for the next 24 hours all at once, each hour is forecast based on the previously forecast hour(s) and the past actual values. This process iterates 24 times to complete the prediction of the next 24 hours.

Estimation of φs and θs is usually less challenging and is done by some sort of fitting method like Maximum Likelihood (ML) estimation once the order of model (i.e., determining p, d, and q) is determined. Selecting the proper order for the model is usually much more difficult, and there is no unique method for it. One approach is to use the correlation analysis of the time series and error terms through Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). There are suggested tips for determining p and q based on ACF and PACF plots but it does not always give the best model [13]. After selecting the model and estimating the parameters, the fitness of the model to data is examined with criteria such as Akaike Information Criterion (AIC), or Bayesian Information Criterion (BIC). It is worth mentioning that better AIC or BIC does not mean that model has the least SMAPE. Therefore, we have used cross validation to select the best model.

In this study, we used the 'auto.arima' function of the 'forecast' toolbox in the R programming language to select the model and estimate the parameters [14]. The cross validation was used to determine the (p,d,q) triple that reduces the SMAPE the most. The φ and θ parameters were estimated on the training data for the optimum selected model to forecast on the test dataset.

C. Pattern Sequence-based Forecasting (PSF)

This method was first introduced in [15] and an improved version was published later [16]. The idea is based on assigning each 24 hour set, i.e., a day, to a cluster and then the forecast is based on the cluster labels rather than actual values in each day. By clustering, the dimension of each day reduces to one (label of the day) instead of 24 (values for 24 hours). It also adds the robustness by substituting real values (e.g., power consumption) with integer numbers (cluster labels).

The first step in applying this algorithm is to find the clustering method and the optimum number of clusters. In this study, we use the k-means clustering algorithm similar to the one in [15]. In order to determine the optimum number of clusters, k-means clustering is performed on the training dataset for a range of k and the one with the higher validity index is selected. The value of k varies from 2 to the number of unique days in the dataset. The validity index is a clustering statistic that helps to find the optimum number of clusters. The silhouette index, SI, is the validity index used in [15]. If the total number of data points is N_{tr}, for each data point $\underline{E}(i)$ (i.e., a day in the training dataset) which belongs to cluster C with n_c members, silhouette value is defined as following:

$$sil(i) = \frac{a(i) - b(i)}{\max\{a(i), b(i)\}} \quad (5),$$

where

$$a(i) = \frac{1}{n_c - 1}\sum_{j\neq i, j\in C} d(i,j)$$
$$b(i) = \frac{1}{N_{tr} - n_c}\sum_{j\notin C} d(i,j) \quad (6),$$

and $d(i,j)$ denotes the dissimilarity or distance between $\underline{E}(i)$ and $\underline{E}(j)$. Value of $sil(i)$ can range between -1 and +1 where a negative value $(a(i) < b(i))$ indicates a poor clustering for data point $\underline{E}(i)$ (the average distance of the $\underline{E}(i)$ from other data points in the same cluster is more than the average distance of the $\underline{E}(i)$ from all other data points belonging to other clusters) and similarly, a positive value indicates a suitable clustering for the $\underline{E}(i)$. The silhouette index is then calculated as the average of $sil(i)$ on all the data points. Evidently, the number of clusters that maximizes the silhouette index is selected as optimum.

In order to improve the finding of the true optimum number of clusters, it has been proposed in [16] to use three different validity indices for clustering, namely silhouette, Dunn, and Davies-Bouldin. The optimum number of clusters is then picked based on a voting mechanism which is very similar to majority voting. However, in our simulations, these three indices rarely agreed even on the top five number of clusters, so we decided to only use silhouette index as in [15].

After clustering, a 24 hour vector of each day is being substituted by a cluster number; so the real valued time series of $\{\underline{E}(24), \underline{E}(48), \dots, \underline{E}(n-24), \underline{E}(n)\}$ is replaced with an integer valued time series of $\{c(1), c(2), \dots, c(\frac{n-24}{24}), c\left(\frac{n}{24}\right)\}$ where $c\left(\frac{i}{24}\right)$ indicates the cluster label for data point $\underline{E}(i)$. Now, in order to find an estimate for $\underline{\hat{E}}(t_{s^*})$, where $t_{s^*} \in t_s$ is an instance of test set indexes, a template of cluster labels for the previous days $\{c(t_{s^*} - D), \dots, c(t_{s^*} - 2), c(t_{s^*} - 1)\}$ is created where similar to kNN, D is the depth of comparison (which is called window length in [16]). Then, the time series of all the preceding days of t_{s^*}, i.e. $\{c(1), c(2), \dots, c(t_{s^*} - 2), c(t_{s^*} - 1)\}$ is matched against the above mentioned template. $\underline{\hat{E}}(t_{s^*})$ is then equal to the average of the cluster centers of the days following immediately after the matched template indices. If there is no match for the template with the length of D, the template length is shortened to $D - 1$ and algorithm iterates until there is some match in the time series. Similar to kNN, parameter D is determined through cross validation.

The steps of PSF are detailed in Fig. 3.

Pattern Sequence-based Forecasting (PSF) Algorithm

```
Inputs: $\{\underline{E}(t_r)\} = \{\underline{E}(24), \underline{E}(48), \ldots, \underline{E}(n-24), \underline{E}(n)\}$, $D$
Output: $\hat{\underline{E}}(t_{s^*})$
1.  for $k \in \{2,3,\ldots,|unique\{\underline{E}(t_r)\}|\}$
2.      Perform k-means clustering with $k \to C_k$ with cluster centers $CC_1, CC_2, \ldots, CC_k$
3.      Calculate $SI[k]$
4.  $k^* = \arg(\min_k SI[k])$
5.  Replace $\{\underline{E}(t_r)\}$ with $\{c(\frac{t_r}{24})\}$ according to $C_{k^*}$
6.  while $idx = \emptyset$
7.      $temp(t_{s^*}) = \{c(t_{s^*} - D), \ldots, c(t_{s^*} - 2), c(t_{s^*} - 1)\}$
8.      $idx = \underset{i \in t_r}{find}(temp(i) == temp(t_{s^*}))$
9.      $D = D - 1$
10. $\hat{\underline{E}}(t_{s^*}) = \frac{1}{size(idx)} \sum_{i \in idx} CC_{c(idx[i])}$
```

Figure 3. PSF Algorithm according to [15].

IV. Simulation and Analysis

A. Data and Preprocessing

The algorithms described above are applied to charging stations located on the UCLA campus. The data used in this paper were recorded from December 7, 2011 to October 16, 2013; however, not all outlets were in use all days. Among charging outlets at UCLA, 15 outlets have charging data for more than 60 effective days (days that some nonzero charging has been reported); these outlets have been used in our implementation. The number of effective days for each outlet is reported in Table III.

Data for each outlet is in the format that is called Charging Records. Each charging record contains the beginning and end of the charging time as well as the acquired energy. The Charging Records are converted to time series by uniformly dividing the acquired energy to the charging interval; e.g., if charging interval is 3 hours and the acquired energy is 3kWh, it is assumed that the EV received 1kWh of energy in each hour.

There was no normalization or feature extracting from the data. The only implemented pre-processing was to force energy records that were mistakenly recorded as more than the physical maximum of the charging device (E_{max}) and less than zero to the interval of $[0, E_{max}]$.

B. Parameter Selection

The following combinatorial parameters need to be determined for our algorithms via cross validation: Depth (D) for kNN and PSF, number of neighbors (k) for kNN, and (p,d,q) for ARIMA.

There are some challenges with cross validation when applying machine learning methods to time series forecasting problems [17]. On one hand, in the machine learning community, methods such as k-fold cross validation are popular. In k-fold cross validation, for evaluating a certain set of candidate parameters, the training data is randomly divided into k blocks. Then the algorithm trains on k-1 blocks while the error is calculated on the remaining part, i.e., the validation set. This process iterates for total of k times, where each time one of the blocks will be the validation set and the other k-1 blocks will make up the training set. The average error of the algorithm on these k iterations will determine the final error for the current selection of parameters. The whole process is repeated for different combinations of parameters and the combination that yields the least final error is selected as algorithm parameters. For k-fold cross validation, 5 or 10 is a typical choice for k [10].

On the other hand, in time series forecasting literature, in order to recognize the temporal order of the data, methods similar to last block validation are more popular. In this method, only the last block of the training data is considered as validation data and the performance of different parameters are reported on it. It is similar to k-fold cross validation, except that the data is not selected randomly and only the last folder is being treated as validation set.

Advantage of k-fold cross-validation is to use all the training data for validation whereas advantage of the last block validation is respecting the temporal order. However, it has been shown that last block validation has a poor estimation of the error on the test set [17]. In order to take advantage of these two methods, we are using a modified form of blocked cross validation method. Blocked cross validation [17] is similar to k-fold cross validation, except that the order of the samples in each block has been preserved. Now, since the first block does not have any preceding values, the modification (similar to [18]) is to select cross validation blocks after a minimum training data (which is used for training the first block). Fig. 4 illustrates the modified blocked cross validation with five validation blocks. First, the algorithm is trained on {T1, T2} blocks and validated on V1 block, then it is trained on {T1,T2,V1} blocks and validated on V2 block, and so on. This cross validation method uses the maximum possible data (in comparison with last block validation) while respecting the temporal order of time series data.

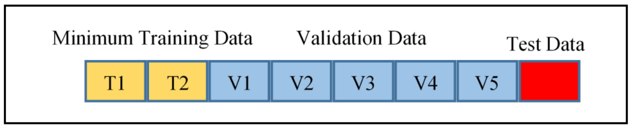

Figure 4. Modified blocked cross vailadtion. Training data is divided to minimum training data {T1,T2} and validation data {V1,…V5}. Model is first trained on minimum training data {T1,T2} and evaluated on V1, then it is trained on {T1,T2,V1} and evaluated on V2, up until training on {T1,T2,V1,…,V4} and evaluating on V5.

In cross validation, the depth parameter (D) is varied between 1 to 60 (equal to looking only at yesterday or up to past two months) and the number of neighbors (k) varied between 1 to 10 for kNN. Also, in the auto.arima function, maximum of p and q was set to 5 and 8 respectively. Parameter d was picked by the auto.arima function based on the KPSS test [14].

C. Results and analysis

Training set in our simulation was the first 90% of the data which makes the test set the last 10% of the data. Minimum triaging data was 30% of the training data (30% of 90%=27% of the whole data) and validation data consist 70% of the training data. The results were not that sensitive to less or more minimum training data. We used five blocks in cross validation.

Table I and II show different parameters selected through cross validation, for each site and each algorithm.

Table IV shows the average and standard deviation of SMAPE on test days for each algorithm and each outlet.

TABLE I. SELECTED PARAMETERS FOR EACH ALGORITHM BASED ON CROSS VALIDATION RESULTS

No	Outlet	kNN (Depth)	ARIMA (p,d,q)	PSF (Depth)	MPSF (Depth)
1	PS3L401LIA3	50	5,1,5	20	1
2	PS8L201LIA1	40	1,1,8	53	1
3	PS8L201LIA3	60	0,1,8	60	1
4	PS8L201LIA4	25	3,1,8	49	1
5	PS8L202LIIA1	10	2,1,6	59	1
6	PS8L202LIIA2	7	5,1,7	7	1
7	PS8L202LIIA3	15	5,0,8	59	1
8	PS9L401LIA1	8	4,1,6	5	1
9	PS9L401LIA2	60	5,1,5	12	1
10	PS9L401LIA3	20	1,1,2	60	1
11	PS9L401LIA5	6	4,1,5	33	1
12	PS9L401LIA6	60	5,1,6	11	2
13	PS9L601LIA1	45	4,1,8	14	1
14	PS9L601LIA3	40	4,1,5	17	1
15	PS9L601LIA4	50	5,1,8	25	1

As it was reported in our earlier work [8], and can be seen in Table II, the optimum k (number of neighbors) is chosen to be equal to 1 for the kNN algorithm in all outlets. This shows that, regardless of the optimum number of previous days to forecast, when dealing with algorithms based on nearest neighbors, it is always better to look at the most similar event in the past and copy its future energy consumption values as the prediction.

TABLE II. SELECTED NUMBER OF NEIGHBOURS FOR kNN AND CLUSTERS FOR PSF BASED ON CROSS VALIDATION RESULTS

No	kNN (Neighbors)	PSF (Clusters)	MPSF (Clusters)
1	1	29	28
2	1	20	33
3	1	38	47
4	1	29	24
5	1	98	89
6	1	101	101
7	1	2	90
8	1	44	65
9	1	71	88
10	1	2	130
11	1	86	86
12	1	104	127
13	1	80	74
14	1	65	54
15	1	53	28

TABLE III. NUMBER OF EFFECTIVE DAYS FOR EACH CHARGING OUTLET

No	Effective Days
1	95
2	84
3	97
4	171
5	163
6	168
7	151
8	178
9	126
10	307
11	189
12	269
13	206
14	178
15	124

Another interesting observation is the poor average performance of the ARIMA model compared with NN (Table IV, last row). We speculate the reason is lots of irregularities in the data along with sparseness (some outlet plugs are not used for a few days and these days do not occur periodically). Consequently, ARIMA that looks for periodic behaviors will fail in this situation compared to NN that looks for local similarities and is able to work around this situation.

All simulations were run with RStudio Version 0.98.507 on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM.

TABLE IV. AVERAGE AND STANDARD DEVIATION OF SMAPE (%) ON TEST DAYS FOR EACH ALGORITHM

No	kNN	ARIMA	PSF	MPSF
1	3.95 ± 6.64	9.71 ± 29.28	42.50 ± 39.41	6.30 ± 8.79
2	6.02 ± 13.12	5.30 ± 22.33	8.35 ± 19.65	0.90 ± 3.64
3	0.60 ± 2.18	1.13 ± 10.58	70.01 ± 7.26	0.25 ± 1.56
4	13.31 ± 12.94	26.17 ± 43.78	15.85 ± 25.52	10.72 ± 11.80
5	20.46 ± 18.64	39.42 ± 48.23	38.01 ± 23.89	31.99 ± 22.00
6	36.25 ± 15.99	40.39 ± 48.58	37.04 ± 27.63	26.67 ± 16.99
7	25.72 ± 19.26	84.18 ± 34.60	81.81 ± 20.89	23.33 ± 18.41
8	29.48 ± 32.33	40.63 ± 48.61	27.82 ± 16.02	18.38 ± 21.77
9	17.56 ± 12.01	46.65 ± 49.33	22.00 ± 12.55	13.98 ± 12.72
10	7.80 ± 15.81	11.81 ± 31.63	96.87 ± 9.01	7.76 ± 17.51
11	14.23 ± 14.97	8.05 ± 26.83	49.07 ± 39.38	8.40 ± 13.68
12	18.05 ± 23.27	9.34 ± 28.47	49.63 ± 31.46	23.25 ± 20.31
13	16.94 ± 14.23	27.84 ± 44.20	35.85 ± 33.73	15.51 ± 10.64
14	9.04 ± 9.55	15.18 ± 34.95	39.50 ± 34.38	10.72 ± 10.70
15	11.88 ± 14.34	8.57 ± 27.46	35.08 ± 39.08	8.63 ± 12.14
Mean	**15.42 ± 15.02**	**24.96 ± 35.26**	**43.29 ± 25.32**	**13.78 ± 13.51**

V. PROPOSED ALGORITHM: MODIFIED PSF

Based on the results analyzed in the previous section, the Nearest Neighbor (NN) algorithm has higher accuracy on most of the charging outlets; and outperforms PSF. However, noticing the way PSF works, it is very similar to kNN, with the difference that the number of nearest neighbors (k) is not specified beforehand; rather it is equal to the number of template matching incidents. On the other hand, from the results in Table II, it is evident that kNN has best performance when only one nearest neighbor is considered. Therefore, we propose a modified PSF, so that the PSF considers only the most recent match in the previous data and returns the corresponding cluster center as output.

Also, the optimum cluster size seems to be ill conditioned for some of outlets, e.g., outlet 7, and 10. As an example silhouette index as a function of number of clusters is depicted in Fig 5 for outlet 10. This is expected since the data is sparse; therefore, the number of clusters that maximizes the silhouette index might be only two (one cluster for near zero days and one cluster for non-zero days). Two clusters essentially map the time series to a binary one and it is not distinguishing enough for forecasting purposes since it cannot code different possible scenarios. We decided to start k (number of clusters in k-means) equal to 10% of distinct days to avoid degenerate clustering.

An issue in the original PSF algorithm is that sometimes there is no matched sequence in the past, even when the depth of the template is 1. This means that the last day in the test template $c(t_{s^*}-1)$, is a member of a cluster with only one member. The algorithm fails in this case. In our modification, we set the output in such a case to be the center of the most common cluster. The modified algorithm is depicted in Fig 6.

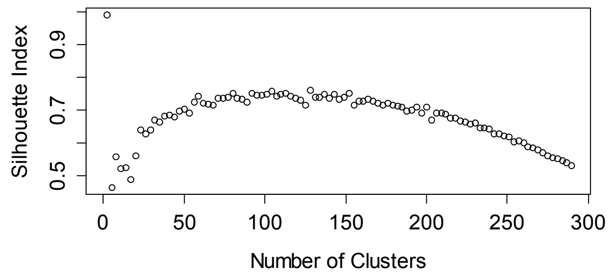

Figure 5. Silhouette index for clustering outlet 10 data. At 2 (selected by PSF), there is a cosmetic maximum for the index while 130 (selcted by MPSF) seems to be a better maximum; clustering and prediction error wise.

The result of applying the MPSF is shown in Table IV. Comparing PSF and MPSF results, it is interesting how increase in the optimum number of clusters (Table II), has decreased the depth of template matching (Table I) such that for most outlets, by looking only at today's data a low error prediction can be made for tomorrow's consumption.

Modified Pattern Sequence-based Forecasting (MPSF) Algorithm

```
Inputs: ${\underline{E}(t_r)} = {\underline{E}(24), \underline{E}(48), \dots, \underline{E}(n-24), \underline{E}(n)}, D$
Output: $\hat{\underline{E}}(t_{s^*})$
1.  for $k \in \{0.1 \times |unique\ \{\underline{E}(t_r)\}|, \dots, |unique\ \{\underline{E}(t_r)\}|\}$
2.          Perform k-means clustering with $k \rightarrow C_k$ with cluster
            centers $CC_1, CC_2, \dots, CC_k$
3.          Calculate $SI[k]$
4.  $k^* = \arg(\min_k SI[k])$
5.  Replace $\{\underline{E}(t_r)\}$ with $\{c(\frac{t_r}{24})\}$ according to $C_{k^*}$
6.  while $(idx = \emptyset \,\&\, D > 0)$
7.          $temp(t_{s^*}) = \{c(t_{s^*} - D), \dots, c(t_{s^*} - 2), c(t_{s^*} - 1)\}$
8.          $idx = \max_{i \in t_r}(find(temp(i) == temp(t_{s^*})))$
9.          $D = D - 1$
10. if $D = 0$
11.         $idx = \arg(\max\left(\mathrm{count}_{i \in t_r}\, c(i)\right))$
12. $\hat{\underline{E}}(t_{s^*}) = CC_{c(idx)}$
```

Figure 6. Modifed PSF (MPSF) Algorithm.

VI. Conclusion and Future Work

In this paper, we analyze three algorithms for the prediction of energy consumption in EV charging stations at the outlet level and propose an improved one. Out of these three algorithms, kNN, ARIMA, and PSF, Nearest Neighbor (kNN with k=1) was the most successful algorithm with the SMAPE measure. Considering the advantage of PSF which is increased robustness to noise in comparison with NN, and the advantage of NN which has the least SMAPE for our data, we proposed Modified PSF (MPSF) where only the most recent match in the previous data is used to generate the prediction. Results show that the modification has improved the performance.

The NN algorithm is currently running on the cellphone application for EV owners at UCLA due to its speed [8]. In the future work, we plan to reduce the computational load of the MSPF algorithm further, to make it suitable for the real time cellphone application.

References

[1] Alternative Fuels Data Center, U.S. Department of Energy, [Online]. Available http://www.afdc.energy.gov/fuels/stations_counts.html

[2] http://www.chademo.com/

[3] M. Majidpour, W.P. Chen, "Grid and Schedule Constrained Electric Vehicle Charging Algorithm Using Node Sensitivity Approach", *Proc. 2012 Intl. Conf. Connected Vehicles and Expo (ICCVE),* pp. 304-310.

[4] C. Chung, A. Shepelev, C. Qiu, C. Chu, R. Gadh, "Design of RFID Mesh Network for Electric Vehicle Smart Charging Infrastructure", *IEEE RFID TA 2013*, Johor Bahru, Malaysia, 4-5 September, 2013.

[5] S. Mal, A. Chattopadhyay, A. Yang, R. Gadh, "Electric vehicle smart charging and vehicle-to-grid operation", *Intl Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 3. March 2012.

[6] C.Chung, E. Youn, J. Chynoweth, C. Qiu, C. Chu, R. Gadh, "Safety Design for Smart Electric Vehicle Charging with Current and Multiplexing Control", *2013 IEEE International Conference on Smart Grid Communications,* Vancouver, Canada, 21-24 October, 2013.

[7] C. Chung, J. Chynoweth, C. Qiu, C. Chu, R. Gadh, "Design of Fast Response Smart Electric Vehicle Charging Infrastructure", *IEEE Green Energy and Systems Conf.*, Long Beach, CA, Nov 25, 2013.

[8] M. Majidpour, C. Qiu, C-Y. Chung, P. Chu, R. Gadh, H. Pota, "Fast Demand Forecast of Electric Vehicle Charging Stations for Cell Phone Application", in *Proc. IEEE/PES General Meeting, 27-31 July 2014, Washington, D.C., USA,* in press.

[9] GEP Box, GM Jenkins, GC Reinsel, *Time series analysis: forecasting and control*, Wiley Series in Probability and Statistics, 2013 (4th ed.)

[10] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection". *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence* 2 (12): pp.1137–1143, 1995.

[11] C. M. Bishop, N. M. Nasrabadi, *Pattern recognition and machine learning*. Vol. 1. New York: springer, 2006.

[12] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression". *The American Statistician* 46 (3): pp. 175–185, 1992.

[13] Weisang, G., & Awazu, Y. (2008). Vagaries of the Euro: an Introduction to ARIMA Modeling. Case Studies in Business, Industry, and Government Statistics, 2, 45-55.

[14] Hyndman, R.J. and Khandakar, Y. (2008) "Automatic time series forecasting: The forecast package for R", *Journal of Statistical Software*, **26**(3).

[15] Martínez-Álvarez, F., Troncoso, A., Riquelme, J. C., & Aguilar-Ruiz, J. S. (2008, December). LBF: A labeled-based forecasting algorithm and its application to electricity price time series. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 453-461). IEEE.

[16] Martinez Alvarez, F., Troncoso, A., Riquelme, J. C., & Aguilar Ruiz, J. S. (2011). Energy time series forecasting based on pattern sequence similarity. Knowledge and Data Engineering, IEEE Transactions on, 23(8), 1230-1243.

[17] Bergmeir, Christoph, and José M. Benítez. "On the use of cross-validation for time series predictor evaluation." Information Sciences 191 (2012): 192-213.

[18] http://robjhyndman.com/hyndsight/crossvalidation/

[19] Luo, Zhuowei, Yonghua Song, Zechun Hu, Zhiwei Xu, Xia Yang, and Kaiqiao Zhan. "Forecasting charging load of plug-in electric vehicles in China." In Power and Energy Society General Meeting, 2011 IEEE, pp. 1-8. IEEE, 2011.

[20] Feixiang, Xie, Huang Mei, Zhang Weige, and Li Juan. "Research on electric vehicle charging station load forecasting." In Advanced Power System Automation and Protection (APAP), 2011 International Conference on, vol. 3, pp. 2055-2060. IEEE, 2011.

[21] Chen, Wenying, Xingying Chen, Yingchen Liao, Gang Wang, Jianguo Yao, and Kai Chen. "Short-term load forecasting based on time series reconstruction and support vector regression." In *TENCON 2013-2013 IEEE Region 10 Conference (31194)*, pp. 1-4. IEEE, 2013.

[22] Xydas, E. S., C. E. Marmaras, L. M. Cipcigan, A. S. Hassan, and N. Jenkins. "Electric Vehicle Load Forecasting using Data Mining Methods."