Fast Prediction for Sparse Time Series: Demand Forecast of EV Charging Stations for Cell Phone Applications

Mostafa Majidpour, *Student Member, IEEE*, Charlie Qiu, Peter Chu, Rajit Gadh, *Member, IEEE*, and Hemanshu R. Pota, *Member, IEEE*

***Abstract*—This paper proposes a new cellphone application algorithm which has been implemented for the prediction of energy consumption at electric vehicle (EV) charging stations at the University of California, Los Angeles (UCLA). For this interactive user application, the total time for accessing the database, processing the data, and making the prediction needs to be within a few seconds. We first analyze three relatively fast machine learning-based time series prediction algorithms and find that the nearest neighbor (NN) algorithm (k NN with $k = 1$) shows better accuracy. Considering the sparseness of the time series of the charging records, we then discuss the new algorithm based on the new proposed time-weighted dot product (TWDP) dissimilarity measure to improve the accuracy and processing time. Two applications have been designed on top of the proposed prediction algorithm: one predicts the expected available energy at the outlet and the other one predicts the expected charging finishing time. The total time, including accessing the database, data processing, and prediction is approximately 1 s for both applications. The granularity of the prediction is 1 h and the horizon is 24 h; data have been collected from 20 EV charging outlets.**

***Index Terms*—Cellphone applications, electric vehicles (EVs), Kernel methods, nearest neighbor (NN) searches, prediction methods, sparse time series.**

I. Introduction

ELECTRIC vehicles (EVs) [or plug-in hybrid electric vehicles (PHEV)] will be an important part of the Smart Grid. The big challenge for EVs is their charging within the existing distribution system infrastructure. According to the EV33 rule (33 miles driving range for a single charge [5]), the minimum battery size in EVs varies from 8.6 to 15.2 kWh [6]. Charging batteries with the aforementioned size will take between 4 and 8 h in a Level 1 household charger (120V, 16 A). As an alternative, EV owners can charge their vehicle at their place of employment, provided that the employer has installed the chargers in the parking lot. Other than workplace parking lots, cities and private operators have invested in charging stations for public use. As of October 2013, the number of these stations in the United States has reached 19 730 which is roughly a sixth of the number of gas stations in the United States [11]. Electric power has to be consumed simultaneously upon generation. As a result, the EV charging station needs to have a good idea of how many vehicles will need charging at each moment. On the other hand, since a reasonable amount of charging will take tens of minutes, it would be useful for a customer to know when and how much energy is expected to be available at a given charging station in a given time window. Both the customer and moderator will substantially benefit from an algorithm that can predict the power consumption at charging stations.

With emerging smart phones, it would be beneficial to customers if they could receive the prediction information on their cellphone, giving them an estimate of the charging time. This paper discusses prediction algorithms that can run in less than a few seconds, so that EV users can query the system and get the results in a reasonable time on their cellphones. To our knowledge, our work is the first research that discusses fast prediction of the available energy and/or expected charging finishing time at the charging outlet level for use in a cellphone application. Data have been recorded from EV owners that charge their vehicle at the University of California, Los Angeles (UCLA) campus parking lots. Any EV can use any parking lot on the campus and each outlet might be used by different EVs during the day. The entrance and exit time of each EV to the parking lot and the total acquired energy from each outlet have been recorded as a charging record datum. Also, there is a multiplexing of the current at each station level (station is a group of outlets sharing one source of current), therefore not all the outlets can be supplied simultaneously and their consumption is dependent on each other.

Since the algorithms have been applied at the outlet level, the complexity of the algorithm is not the function of the total number of EVs or the total number of outlets in the system. For each outlet, the complexity of the algorithm is a function of the size of charging record data per day which is a bounded number (it takes a few hours to charge an EV, thus the number of EVs getting charged per day per outlet is bounded).

The work described in this paper differs from other previous works in the literature.

1) We have used just one type of recorded data, charging records, which only contains the start and end of the

Manuscript received March 1, 2014; revised July 4, 2014 and September 24, 2014; accepted October 23, 2014. Date of publication November 25, 2014; date of current version February 02, 2015. This work was supported in part by the Los Angeles Department of Water and Power (LADWP)/Department of Energy (DOE) under Grant 20699 and Grant 20686, and in part by the Smart Grid Regional Demonstration Project. Paper no. TII-14-0266.

M. Majidpour, C. Qiu, P. Chu, and R. Gadh are with the Smart Grid Energy Research Center, University of California, Los Angeles, CA 90095 USA (e-mail: mostafam@ucla.edu).

H. R. Pota is with the School of Engineering and Information Technology, University of NSW, Canberra, A.C.T. 2610, Australia (e-mail: h.pota@adfa.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2014.2374993

charging transaction and the total amount (a scalar value; not time dependent) of energy received in the charging transaction rather than geographical or any driving habit-related data.
2) Our predictions are at the charging outlet level (not parking lot or the whole building level).
3) Our method is online and fast with the whole process taking about a second.

Indeed, the reason that we do not use other sources of data is our speed requirement and the fact that adding more data will slow the process. Other state-of-the-art methods which have been applied on this same data take substantially more time. Specifically, depending on the parameter selection process, support vector regression and random forest take, respectively, 3900 and 306 s to produce the results [38].

The rest of this paper is organized as follows. Section II provides a brief review of existing literature. Section III formulates the problem. Section IV explains the methods that are evaluated in this paper in order to compare with the proposed method. Section V reports and analyzes the result of applying the algorithms on the UCLA parking structures' data. Section VI explains the proposed algorithm via modifications to discussed algorithms. Section VII explains about the implementation of the cellphone applications using the proposed algorithm. Section VIII provides the conclusion and future work.

II. Literature Review

Time series prediction (forecasting) methods predict the future of a certain variable given its past history. There is a rich literature on different methods of time series prediction that has evolved from statistics, mathematics, computer science, economics, and engineering [13]–[15]. Probably, the most famous model in time series prediction is the ARIMA model with Box-Jenkins approach [14] which has been used widely in economics and statistics [15] and is considered to be the traditional approach in time series prediction. Selecting the correct parameters for the ARIMA model is not a trivial task, and depending on the approach, it might be time consuming.

Prediction algorithms are part of most modern smart grid technologies [34] such as photovoltaic systems [36], smart buildings [32], and wind turbines [35]. Reference [39] combines probability and fuzzy systems concepts to propose a method for predicting the wind speed. A similar problem involving wind power forecasting was addressed with implementing an artificial neural network (ANN) based on the predictions of global forecast system. A new optimal type reduction for the interval type-2 fuzzy logic systems along with ANN has been proposed for load forecasting in [42].

Extensive research has focused on EV charging algorithms and charging station infrastructures [7]–[10], and as a next step, EV-related research has started to utilize forecasting algorithms [16]–[22]. Some papers apply forecasting algorithms to EV driving habits and predict the state of charge (SOC) of a particular EV and when it needs to be charged [17], [18]. The authors in [16] have applied ANN forecasting algorithms to predict the charging profile of the EV within the Building Energy Management System (BEMS) in order to improve the overall energy efficiency of the building. References [19] and [20] discuss the prediction of the EV charging profile while taking into account various sources of data, such as vehicle driving and usage data. The authors in [21] and [22] consider forecasting at the charging station level based on the EV user classification and Monte Carlo simulations method. Reference [33] discusses an energy management system for EVs that takes advantage of prediction in different levels through hierarchical model predictive control. A more comprehensive scenario of combining prediction of solar energy (by using radial basis function networks) and optimizing the cost of PHEV based on factors such as daily distance driven, electricity market price, and load values (by using genetic algorithms) has been studied in [36]. Some researchers have facilitated prediction and access to EV-related information by aggregating several sources of data [41].

As mentioned above, the aim of this work is to focus on relatively fast time series forecasting algorithms in which the whole process of prediction (including preprocessing) takes a reasonably short amount of time for a user that sends prediction-related queries from a cellphone. Machine learning (ML)-based heuristic methods have been shown to provide a good performance in forecasting [12]. Some of these methods such as *k*-nearest neighbor (kNN) and weighted kNN depending on the number of neighbors could be pretty fast. However, the selection of parameters for these ML methods still remains a challenge.

For our predictor application, we are interested in the ML-based algorithms that have low computation requirements and are relatively faster. A brief introduction of these methods will follow in Section IV.

III. Problem Formulation

The objective is to predict the available energy in the next 24 h at each charging outlet with a minimum time for processing. Formally, we assume that there is some function relating future available energy and the past consumed energy

$$\hat{E}(t) = f(E(t-i), \varepsilon(t)), \qquad i \in \{1, 2, \ldots\} \tag{1}$$

where $E(t)$ is the actual energy consumption at time t, $\hat{E}(t)$ is the prediction of the energy consumption at time t, and $\varepsilon(t)$ 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 a particular performance (or error) criteria. For the error measurement, we have chosen symmetric mean absolute percentage error (SMAPE). For the day i, the SMAPE is defined as

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

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

Since there is no access to future data in real life, the last portion of the data (last 10% in this paper) is set aside as the

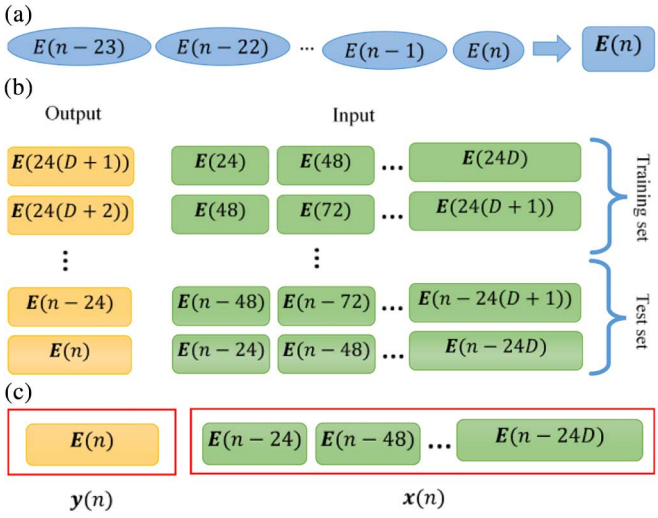

Fig. 1. (a) Energy consumption vector ($\boldsymbol{E}$) for 24 h. (b) Input–output pairs and division of data into training and test sets. (c) Labeling inputs as x and outputs as y.

test set to evaluate the performance of the algorithm. Thus, our goal is to find an algorithm that minimizes the error between the actual value and its prediction in the test set. Note that the test set is not used in either the parameter selection or training phase.

We use the notation $\widehat{\boldsymbol{E}}(t)$ as the vector of prediction for the next 24 h ending at t [Fig. 1(a)]. Also, $t_r = \{1, 2, \ldots, N_{\text{tr}}\}$ and $t_s = \{N_{\text{tr}} + 1, \ldots, N\}$ are the set of indices for the training and test sets, respectively. Later on, in the parameter selection phase, parts of the training set will be treated as the validation set. The different methods used to select the validation set are further explained in the parameter selection section.

IV. Applied Algorithms

Four prediction algorithms have been briefly described here. These algorithms were employed to compare and demonstrate the effectiveness of the proposed approach. A detailed description can be found in [25].

A. Historical Average

This algorithm is one of the simplest algorithms that is sometimes referred to as naïve approach and is used only for comparison with other methods. According to this approach, the predicted charging energy in the future is the average of the charging energy consumption in the past. Formally,

$$\hat{E}(t) = \frac{1}{D}\sum_{d=1}^{D} E(t - 24d) \tag{3}$$

where D is the depth of the averaging. For instance, the predicted energy consumption for 3 P.M. on the next day is equal to the average of the energy consumed today, yesterday, and up to the past D days at 3 P.M. D is a parameter that needs to be selected before the evaluation on the test set.

k-Nearest Neighbor Algorithm

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

Fig. 2. k-NN algorithm.

B. K-NN

This algorithm is a well-known algorithm in the ML community [26]. Based on the k-NN (kNN) algorithm, each sample (training, test, or validation) is composed of input and output pairs. In our application, as seen in Fig. 1(c), the output is the predicted energy consumption for the next 24 h

$$\boldsymbol{y}(t) = \boldsymbol{E}(t) \tag{4}$$

and the input is the concatenation of the consumption records for up to D previous days

$$\boldsymbol{x}(t) = \{\boldsymbol{E}(t - 24), \boldsymbol{E}(t - 48), \ldots, \boldsymbol{E}(t - 24D)\}. \tag{5}$$

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 $\boldsymbol{y}(t_{s^*})$ where $t_{s^*} \in t_s$ is an instance of test set indices; first, the dissimilarity between $\boldsymbol{x}(t_{s^*})$ and all other $\boldsymbol{x}(t_r)$ that belong to the training set is computed. We have used the Euclidian distance as the measure of dissimilarity here. After determining the k closest $\boldsymbol{x}(t_r)$ to $\boldsymbol{x}(t_{s^*})$, the average of their corresponding $\boldsymbol{y}(t_r)$ is generated as $\boldsymbol{y}(t_{s^*})$. In this algorithm, the parameter k needs to be determined. Fig. 2 illustrates the algorithm where $\text{dis}[j]$ refers to dissimilarity between $\boldsymbol{x}(t_{s^*})$ and $\boldsymbol{x}(j)$.

C. Weighted k-NN

Weighted kNN is based on the idea that closer points to the query should contribute more to the output and in this way improve the accuracy of the prediction compared to kNN [1].

This algorithm is very similar to the original kNN algorithm with the exception that instead of averaging the k closest training outputs [Fig. 2, step 5)] their weighted average is used, where the weights are a function of the dissimilarity between input pairs. The weights are defined based on Dudani's weights [1], which give better results compared to other similar weight assigning methods according to both literature [2] and our simulations. Thus, the following two steps will substitute for step 5) in Fig. 2:

$$w_p = \frac{\text{dis}[k+1] - \text{dis}[p]}{\text{dis}[k+1] - \text{dis}[1]}, \quad p = 1, \ldots, k \tag{6}$$

$$\boldsymbol{y}(t_{s^*}) = \frac{1}{\sum_{q\in idx} w_q} \sum_{p\in idx} w_p \boldsymbol{y}(p). \tag{7}$$

Lazy Learning Algorithm

Inputs: $\boldsymbol{x}(t_r), \boldsymbol{y}(t_r), \boldsymbol{x}(t_{s^*}), k_{max}$
Output: $\boldsymbol{y}(t_{s^*})$
1. **for** $j \in t_r$
2. $dis[j] = \|\boldsymbol{x}(t_{s^*}) - \boldsymbol{x}(j)\|$
3. **for** $i \in \{1, \ldots, k_{max}\}$
4. $idx[i] =$ index of i^{th} smallest(dis)
5. **for** $k \in \{2, \ldots, k_{max}\}$
6. $\boldsymbol{y}(k) = \frac{1}{k}\sum_{i\in\{1,\ldots,k\}} \boldsymbol{y}(idx[i])$
7. Calculate $\boldsymbol{e_{LOO}}(k)$ according to Eq. (7)
8. $k^* = \arg(\min_{k\in\{2,\ldots,k_{max}\}} \boldsymbol{e_{LOO}}(k))$
9. $\boldsymbol{y}(t_{s^*}) = \boldsymbol{y}(k^*)$

Fig. 3. LL algorithm.

D. Lazy Learning

Lazy learning (LL) is a generic term which refers to algorithms that postpone the learning until a query is submitted to the system. In fact, weighted kNN and kNN are simple forms of LL. A version of LL from [27] has been implemented in this paper. This version of the LL algorithm is similar to the kNN algorithm, in principle, with the difference that for each query, the optimum number of neighbors (k) is not fixed and is estimated separately. The idea is that for some queries it seems better to look at more neighbors and for some others, fewer neighbors would be enough. For each query, kNN is performed $k_{\max}$ times for $k = \{1, 2, \ldots, k_{\max}\}$. Then, based on leave-one-out (LOO) cross validation, the error for each k is estimated, and the output corresponding to the k with a lower LOO cross validation error is selected. We use the PRESS statistic [27] to estimate LOO for each k. For a fixed $k \in \{1, 2, \ldots, k_{\max}\}$, suppose $j^* \in \{1, 2, \ldots, k\}$ indicates the index of the jth closest neighbor to the query $\boldsymbol{x}(t_{s^*})$. For each j^*, we define an error term

$$\boldsymbol{e_k}(j^*) = k\frac{\boldsymbol{y}(j^*) - \boldsymbol{y}(i)}{k-1} \tag{8}$$

which gives the LOO cross validation error for the specific k

$$\boldsymbol{e_{LOO}}(k) = \frac{1}{k}\sum_{j^*=1}^{k} (\boldsymbol{e_k}(j^*)^T * (\boldsymbol{e_k}(j^*)). \tag{9}$$

The k with smallest $\boldsymbol{e_{LOO}}$ will be selected as the optimum k. The steps are detailed in Fig. 3.

V. Simulation Setup and Preliminary Results

A. Data

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 February 28, 2014; however, for each day, not all outlets were in use. Among all the charging outlets at UCLA, 20 outlets have charging data for more than 60 effective days (days that some nonzero charging has been reported) which have been used in our implementation. The number of effective days for each outlet is reported in Table I.

TABLE I
Number of Effective Days for Each Outlet and Percentage of Data Used for Training, Validation, and Test Sets

No	Outlet	Effective days	Training set (%)	Validation set (%)	Test set (%)
1	PS2L201LIA4	125	75	15	10
2	PS3L401LIA3	95	75	15	10
3	PS8L201LIA1	105	75	15	10
4	PS8L201LIA3	116	75	15	10
5	PS8L201LIA4	192	80	10	10
6	PS8L202LIIA1	184	75	15	10
7	PS8L202LIIA2	189	80	10	10
8	PS8L202LIIA3	179	80	10	10
9	PS8L203LIA1	154	80	10	10
10	PS8L203LIA2	153	75	15	10
11	PS8L203LIA3	159	80	10	10
12	PS8L203LIA4	158	80	10	10
13	PS9L401LIA1	196	70	20	10
14	PS9L401LIA2	147	65	25	10
15	PS9L401LIA3	328	80	10	10
16	PS9L401LIA5	210	70	20	10
17	PS9L401LIA6	290	70	20	10
18	PS9L601LIA1	227	60	30	10
19	PS9L601LIA3	199	70	20	10
20	PS9L601LIA4	145	75	15	10

Data for each outlet come in a format which is referred to as charging records. Each charging record contains the beginning and end of the charging time as well as the acquired energy.

B. Preprocessing

The charging records are converted to time series by uniformly dividing the acquired energy to the charging interval. For example, if the charging interval is 3 h and the acquired energy is 3 kWh, it is assumed that the EV received 1 kWh of energy in each hour. In preprocessing the data, if all the values in an input–output pair $(\boldsymbol{x}(i), \boldsymbol{y}(i))$ are zero or not reported, the pair was removed from the data set.

There was no normalization or feature extracting from the data. The only implemented preprocessing 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 $[0,\ E_{\max}]$.

C. Parameter Selection

The following parameters need to be determined for our algorithms: Depth D for all algorithms which is the number of previous days considered in the input vector and the number of neighbors k in kNN and weighted kNN.

There are different options for performing parameter selection through validation in time series [3]. One of the popular methods is the k-fold cross validation. In k-fold cross validation, for evaluating a certain set of candidate parameters, the training data are divided into k parts ($P_i,\ i = 1, \ldots, k$) and algorithm trains on $k-1$ parts ($P_{-i} = \text{Training set} - P_i$) while the error is calculated on the remaining part (P_i), i.e., the validation set. This process iterates k times, and each time one of the parts will

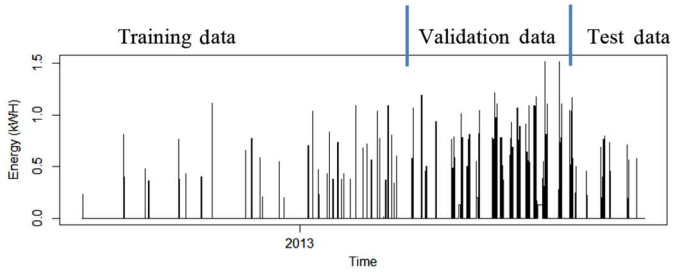

Fig. 4. Separating validation data from training data in the last block validation method.

be the validation set and the other $k-1$ parts 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.

However, [4] discusses how k-fold cross validation in its original form might not be appropriate for time series since it does not respect the order. It proposes a "time series cross validation" in which for cross validation we are only allowed to use training data prior to the validation set, i.e., if validation set is P_i, then the training data would be $P_j,\ j = 1, \ldots, i-1$ instead of P_{-i}. Another approach taken in [29] is to use the last portion of the training data for validation and parameter selection purposes. In this approach, the validation set is the last block of training data P_k and the rest of the data prior to it $P_j,\ j = 1, \ldots, k-1$ is used for training; hence, it is less computationally expensive since it evaluates parameters on just one block rather than k blocks.

We have tried all of the three-mentioned validation methods: k-fold cross validation, time series cross validation, and the last block validation. We found that they lead to similar behavior in the final results; therefore, in this paper, we are going to dedicate the last block of training data to validation data, which is relatively less computationally expensive, and therefore more appropriate for our fast prediction application. This method basically partitions the data to nonoverlapping training, validation, and test data sets as depicted in Fig. 4.

The length of the validation block is the next matter of importance. We must determine how much of the last portion of the available data is a good representative of the behavior of all the data for specific parameters. In order to answer this question, we picked different percentages of the last part of the training data for each outlet and each algorithm, and we compared it with the whole data set in terms of similarity in response to change in parameters.

The smallest percentage of data that could represent the whole training data set for all algorithms is selected as the final validation set for each outlet. Table I shows the percentage of the data that is used as validation set for each outlet. The candidate percentages were: 10%, 15%, 20%, 25%, and 30%.

For parameter selection, the depth parameter D is varied between 1 and 60 (equal to looking only at yesterday or up to the past 2 months); in order to make the process faster, a subset of 1–60, namely {1,2,. . .,9,10,15,20,. . .,60} is employed.

TABLE II
SELECTED DEPTH (D) AND NUMBER OF NEIGHBORS (K) PARAMETER FOR EACH ALGORITHM BASED ON VALIDATION RESULTS

No	Outlet	Historical average	kNN		Weighted kNN		LL
		D	*D*	*k*	*D*	*k*	*D*
1	PS2L201LIA4	1	60	1	6	2	15
2	PS3L401LIA3	1	60	1	60	2	60
3	PS8L201LIA1	1	55	1	8	2	60
4	PS8L201LIA3	1	50	1	60	2	60
5	PS8L201LIA4	2	55	1	4	2	20
6	PS8L202LIIA1	1	40	1	1	2	40
7	PS8L202LIIA2	1	60	1	1	2	35
8	PS8L202LIIA3	1	20	1	1	2	20
9	PS8L203LIA1	1	15	1	1	2	9
10	PS8L203LIA2	1	7	1	1	2	9
11	PS8L203LIA3	1	25	1	40	2	30
12	PS8L203LIA4	1	40	1	7	2	7
13	PS9L401LIA1	1	15	1	1	2	2
14	PS9L401LIA2	1	15	1	4	2	15
15	PS9L401LIA3	1	35	1	15	2	25
16	PS9L401LIA5	1	45	1	45	2	45
17	PS9L401LIA6	1	60	1	1	2	60
18	PS9L601LIA1	1	60	1	60	2	60
19	PS9L601LIA3	1	60	1	55	2	55
20	PS9L601LIA4	1	30	1	30	2	50

The number of neighbors varies between 1–for kNN and 2–5 for weighted kNN (weighted kNN with $k = 1$ is the same as kNN with $k = 1$).

D. Preliminary Results

Tables II and III show the depth (D) and number of neighbors (k) parameters, selected according to the previous section, for each site and each parameter.

After selecting the parameters, the union of the training data and validation data is treated as the new training data set and used for predicting the first day in the test data set. For predicting the second day in the test data set, the union of training data set, validation data set, and the first day in the test data is used; similarly, for predicting the ith day in the test set, all the data prior to it (training data set, validation data set, and test data set up to $(i-1)$th day) is employed.

Interestingly, as can be seen in Table III, for all outlets, the optimum number of neighbors is chosen to be equal to 1 for the kNN algorithm. This shows that, regardless of the optimum number of days to forecast, it is always better to look at the most similar event in the past and copy its future energy consumption values as the prediction. Also, the optimum number of neighbors for weighted kNN in all outlets is 2; considering that k ranges from 2 to 5 and the kNN algorithm with $k = 1$ shows better performance, one can conclude that $k = 1$ is the optimum parameter.

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

The historical average has by far the worst performance. Comparing LL results with kNN, it seems that LL was not successful in choosing the best number of neighbors (k) for each query; otherwise, it would have better results compared to kNN (which has a constant k). For better comparison, we

TABLE III
AVERAGE AND STANDARD DEVIATION (IN PARENTHESES) OF SMAPE (%) ON TEST DAYS FOR EACH ALGORITHM

No	Historical average	kNN	Weighted kNN	LL
1	72.71 (12.50)	15.93 (18.64)	23.20 (19.86)	21.42 (20.12)
2	96.45 (1.89)	3.54 (6.60)	3.64 (6.45)	4.13 (7.20)
3	97.87 (1.39)	16.14 (33.90)	34.68 (45.24)	16.61 (29.41)
4	97.50 (1.11)	48.65 (30.95)	49.71 (33.00)	49.72 (32.47)
5	94.88 (3.97)	2.59 (2.81)	14.04 (17.34)	15.10 (11.85)
6	87.52 (8.43)	28.91 (19.47)	36.04 (23.43)	35.45 (15.06)
7	83.28 (10.87)	35.85 (11.87)	26.98 (15.49)	38.79 (13.16)
8	87.07 (11.61)	28.87 (16.99)	23.17 (15.65)	38.93 (14.04)
9	89.28 (6.84)	23.80 (13.37)	37.05 (17.25)	34.47 (17.48)
10	83.67 (6.75)	32.01 (16.30)	27.93 (15.69)	32.26 (18.30)
11	79.91 (13.64)	33.17 (16.87)	34.06 (16.30)	33.86 (17.95)
12	81.59 (7.72)	29.83 (16.41)	33.68 (16.56)	35.20 (17.45)
13	81.26 (13.72)	21.36 (27.17)	22.39 (21.16)	26.12 (26.89)
14	96.06 (2.44)	13.02 (8.04)	18.92 (8.33)	16.12 (10.20)
15	93.42 (9.83)	9.44 (14.30)	12.36 (18.08)	13.31 (17.56)
16	88.09 (11.33)	7.45 (11.49)	12.72 (12.72)	13.74 (13.61)
17	76.78 (12.14)	4.16 (13.22)	19.12 (16.96)	4.16 (13.33)
18	92.03 (4.80)	11.38 (9.42)	16.41 (10.65)	16.77 (9.89)
19	91.70 (4.52)	10.06 (11.85)	14.75 (12.16)	15.52 (12.04)
20	88.44 (4.96)	8.88 (12.09)	10.27 (14.58)	12.10 (15.60)
Mean	**87.68 (7.82)**	**19.08 (15.94)**	**23.24 (18.16)**	**23.29 (17.00)**

TABLE IV
POST HOC FRIEDMAN TEST WITH HOMMEL'S METHOD OF ADJUSTING p VALUES

Post hoc Friedman test	Historical average	Weighted KNN	LL
z-value	6.919809	2.510727	3.551760
Unadjusted p-value	4.522545×10^{-12}	0.012048	0.000382
Adjusted p-value (Hommel's method)	1.356764×10^{-11}	0.012048	0.000765

kNN is considered the control method.

have applied nonparametric statistical tests to compare the algorithms according to [37]. First, the null hypothesis of these four algorithms having the same power is rejected by the Friedman test (p value $= 9.80 \times 10^{-11}$). Then the Friedman *post hoc* test was performed with Hommel's procedure for adjusting p values. Table IV shows the z values, as well as the unadjusted and adjusted p values when considering the kNN algorithm as a control method. As can be seen, adjusted p values are significantly less than $\alpha = 0.05$; thus kNN has significantly better performance than the other three algorithms.

All simulations were run with RStudio Version 0.97.551 on an Intel Core i-7 CPU at 3.40 GHz with 16 GB RAM.

VI. PROPOSED ALGORITHM BASED ON TIME-WEIGHTED DOT PRODUCT DISSIMILARITY

In Section V, we found that the NN algorithm has the best performance as the fast predictor algorithm on the examined data. However, there is still a room to improve the results since the results for some of the outlets is not yet satisfactory, e.g., outlet no. 3.

Looking at Fig. 2 for the kNN algorithm, it turns out that after selecting k (which, through parameter selection, is equal to 1), there is not much room for modifying the algorithm except step 2), which is the dissimilarity calculation step. The dissimilarity that has been used in the paper is Euclidean distance. Thus, we are focusing on modifying the dissimilarity measure in step 2) of the kNN algorithm in order to improve the performance.

A. Dissimilarity Measures

In applying the NN algorithm, we used Euclidian distance to measure the dissimilarity between two data points and determine the NN of each input query. However, the EV charging data are sparse, meaning that in significant chunks of time, e.g., during nights in the government or official parking spaces, there is no charging in progress and the consumed power is zero. When the data are sparse, it might be beneficial to employ other dissimilarity measures than Euclidian distance.

One way to define the dissimilarity measure is to use the inverse or negative of a similarity measure. A candidate for similarity measure is the dot product of two vectors since it is zero when two vectors are orthogonal to each other and is maximum when they are equal. Specifically, the dot product of two signals that have nonzero elements in different indices is equal to zero. This fits well to sparse time series prediction applications, since the similarity between two pieces of signals with different indices of nonzero elements should be as less as possible.

B. Kernelized Similarity

Upon using the dot product as similarity measure, one can use kernelized similarity measures to get more flexibility and to compute similarity in higher dimensions [31]. In particular, polynomial kernels are interesting for us here since it is the natural extension of the dot product. A polynomial kernel for similarity between x and y is often defined as follows [25]:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^t \boldsymbol{y} + c)^d \tag{10}$$

where $c \geq 0$ is a constant that is trading off the influence of higher order terms versus lower order ones and d is the degree of the polynomial kernel. Now, we can define the dissimilarity measure based on the polynomial kernel

$$\text{dis}(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)) = -K(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)). \tag{11}$$

Another alternative for defining dissimilarity based on kernels is to find the distance of two inputs in the kernel space, which can be obtained from the following equation:

$$\begin{aligned}\text{dis}(\varphi(\boldsymbol{x}(t_1)), \varphi(\boldsymbol{x}(t_2)))^2 &= K(\boldsymbol{x}(t_1), \boldsymbol{x}(t_1)) \\ &+ K(\boldsymbol{x}(t_2), \boldsymbol{x}(t_2)) - 2K(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)).\end{aligned} \tag{12}$$

It, however, needs more computation because of self-mapping terms, and it does not improve our results in practice.

C. Time-Weighted Dissimilarity

Another intuitive modification of the dissimilarity measures could be time weighting; for instance, outputs $\boldsymbol{y}(t_1)$ and $\boldsymbol{y}(t_2)$

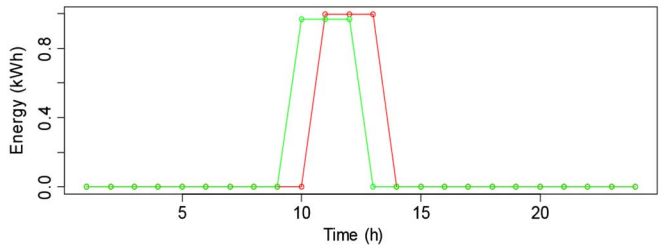

Fig. 5. Actual energy consumption (green) and its prediction with TWDP-based NN algorithm (red) for a sample test day in outlet 1. The SMAPE for this day is 16.93%.

are more similar if the recent values of their corresponding inputs $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$ are more similar. In order to weight, the recent values for an input that has been defined in (5), we have used linear time weighting

$$\boldsymbol{TW} = [1 + D, \ldots, 1 + 2\Delta, 1 + \Delta, 1]$$

where $\Delta = 1/(24D - 1)$ and D is the depth of input. Also, we have tried other weighting methods such as $\exp(\boldsymbol{TW})$, but we did not see improvement in the final results.

Combining all the modifications together, the dissimilarity measure used in kNN algorithm will be substituted with

$$\text{dis}(\boldsymbol{x}(t_1), \boldsymbol{x}(t_2)) = -(\boldsymbol{x}(t_1)^t \text{diag}(\boldsymbol{TW})\boldsymbol{x}(t_2) + c)^d. \quad (13)$$

D. Results

We used the same settings as discussed in Section V and the only difference in this section is modifying the dissimilarity measure. This modification, however, adds two more parameters c and d according to (10) which, like other parameters, need to be determined with validation. However, in different simulations, we did not see much of a difference in the final results of the NN algorithm with the change in c or d. Therefore, we set $c = 0$ and $d = 1$ and use dot product instead of higher order kernels. The conclusion is that similarity in higher order terms are not necessary for finding the prediction-related similarity in our application, and only the linear terms' contribution is enough for prediction. This is good news computational wise since there is no need to try higher order terms. Combining dot product and time weighting, our dissimilarity measure will be time-weighted dot product (TWDP).

Fig. 5 shows a sample test day from Outlet 1 and its prediction. The SMAPE for this specific day is 16.93%. The results for all outlets are presented in Table V. According to the Table V, the average SMAPE has been improved in all methods compared with the Euclidean distance case. It is notable that the maximum SMAPE for all methods has been decreased to less than 35% SMAPE.

In order to test for statistical significance, we use the same method as in previous section: Friedman test to reject the null hypothesis that all these algorithms have similar power, and if the null hypothesis is rejected, then we apply the *post hoc* procedures to compare the statistical significance of the power of these algorithms. The p value is 0.000118 for the Friedman test; therefore, there is a statistical significance between the

TABLE V
AVERAGE AND STANDARD DEVIATION (IN PARENTHESES) OF SMAPE (%) ON TEST DAYS FOR EACH ALGORITHM WITH TWDP DISSIMILARITY MEASURE

No	Outlet	kNN (k = 1)	Weighted kNN (k = 2)	LL
1	PS2L201LIA4	13.60 (16.55)	18.15 (16.58)	19.41 (14.63)
2	PS3L401LIA3	3.37 (6.95)	3.28 (6.04)	4.12 (6.13)
3	PS8L201LIA1	0.62 (2.27)	0.62 (2.39)	0.90 (2.20)
4	PS8L201LIA3	1.04 (3.90)	1.85 (4.43)	1.85 (5.06)
5	PS8L201LIA4	9.24 (9.87)	19.07 (13.06)	19.07 (12.48)
6	PS8L202LIIA1	20.23 (18.92)	23.43 (20.84)	22.00 (17.59)
7	PS8L202LIIA2	30.07 (16.41)	35.00 (9.26)	33.79 (10.05)
8	PS8L202LIIA3	23.83 (18.97)	26.04 (18.83)	26.59 (20.23)
9	PS8L203LIA1	22.03 (13.37)	24.78 (22.20)	25.24 (15.03)
10	PS8L203LIA2	20.84 (14.90)	24.32 (16.81)	23.92 (16.86)
11	PS8L203LIA3	31.20 (11.54)	36.57 (17.52)	35.54 (16.90)
12	PS8L203LIA4	26.53 (15.27)	24.69 (13.35)	26.56 (12.34)
13	PS9L401LIA1	22.85 (21.03)	32.94 (28.15)	31.91 (27.97)
14	PS9L401LIA2	11.93 (7.62)	13.56 (8.61)	13.88 (10.04)
15	PS9L401LIA3	6.40 (13.14)	6.40 (13.35)	6.40 (13.89)
16	PS9L401LIA5	14.49 (12.88)	16.17 (13.84)	15.72 (13.62)
17	PS9L401LIA6	19.09 (17.69)	14.23 (21.82)	14.23 (21.96)
18	PS9L601LIA1	13.69 (12.87)	16.62 (14.67)	16.42 (14.03)
19	PS9L601LIA3	7.58 (11.72)	8.15 (10.89)	8.84 (10.71)
20	PS9L601LIA4	6.44 (11.43)	6.61 (11.16)	6.84 (11.43)
Mean		**15.27 (13.24)**	**17.52 (14.56)**	**17.59 (14.01)**

TABLE VI
POST HOC FRIEDMAN TEST WITH HOMMEL'S METHOD OF ADJUSTING p VALUES. MODIFIED NN IS CONSIDERED THE CONTROL METHOD

Post hoc Friedman test	Weighted kNN (k = 2)	LL
z-value	3.083221	3.794733
Unadjusted p-value	0.002047	0.000147
Adjusted p-value (Hommel's method)	0.002047	0.000295

methods, and we can apply the *post hoc* Friedman test and consequently adjust the p values. The statistics have been displayed in Table VI. Accordingly, the adjusted p values are less than $\alpha = 0.05$ which shows the statistical significance of the proposed method over weighted kNN and LL.

The interesting difference in the pattern of SMAPE errors between results from two different dissimilarity measures has been depicted for NN case in Fig. 6. As this figure shows, for outlets that the Euclidean dissimilarity has relatively high errors such as outlet no. 4, the TWDP dissimilarity has relatively low errors and vice versa. The fact that SMAPE error in outlet no. 4 has decreased from 48.65% (NN with Euclidean distance) to 1.04% (NN with dot product dissimilarity) illustrates that TWDP was extremely successful in finding similar points in the time vectors and making the prediction based on that. This phenomenon shows that, depending on the characteristic of the time series in hand, we might need to change our point of view (from measuring Euclidean dissimilarity to TWDP one) to be able to see the similar points in the training data to our test query.

In fact, the effective characteristic here seems to be sparseness of the time series. Fig. 7 shows the percentage of the

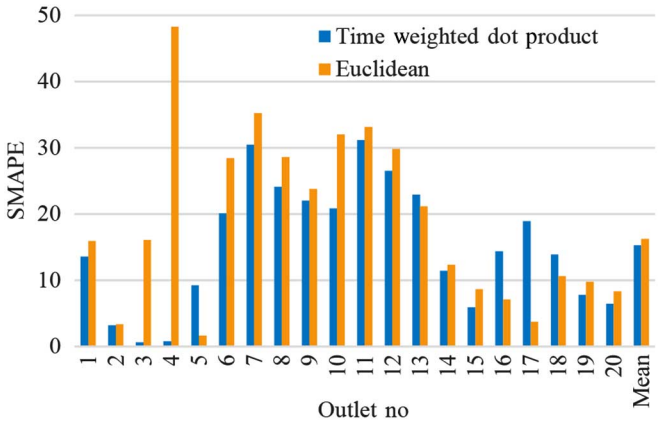

Fig. 6. Comparing the accuracy of TWDP and Euclidean-based dissimilarities in NN.

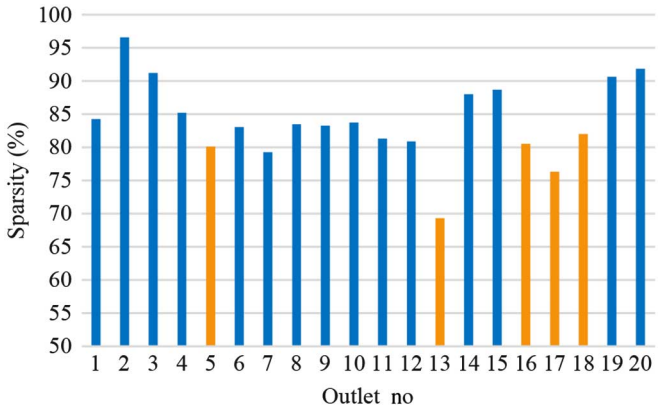

Fig. 7. Sparseness of the time series of each outlet calculated at the optimum depth. Orange and blue bars are for outlets that Euclidean and TWDP dissimilarity has better accuracy, respectively.

sparseness of the time series (number of zero entries divided by the total number of entries in the times series) calculated with the optimum depth for each outlet. Comparing the orange and blue bars shows an interesting relationship: The time series, which TWDP dissimilarity does better are mostly sparser ones.

In order to take advantage of both dissimilarity measures, we implement the best method for each outlet with the best dissimilarity measure. For example, in Fig. 6, for outlet no. 4, we use the NN with the TWDP dissimilarity, whereas for outlet no. 17, we use the NN with Euclidean dissimilarity.

An accurate prediction will enable station owners to provide a realistic time to charge which is essential to customer satisfaction. The more accurate algorithm will help the EV charging outlet owner to predict the energy consumption of his/her facility in the future, thus the station owner can utilize all the capacity of charging stations and, therefore, obtain more profit. Depending on which factor is more important for a certain EV station owner, she/he can penalize the over prediction of consumption (which translates to empty stations in sometimes of the day) or under prediction of the consumption (which translates to a disappointed EV owner whose car was not charged in the predicted time frame) in an appropriate way. Here, the SMAPE accuracy measure introduced in (2) is a symmetric one and does not penalize either over prediction or under prediction. However, algorithms are readily usable for asymmetric error measurement criteria.

VII. Cellphone Applications Based on Proposed Algorithm

Based on the results, NN with the TWDP dissimilarity measure has higher accuracy on most of the outlets with sparser time series and NN with the original Euclidean distance dissimilarity is performing better on outlets with less sparse time series were selected as the algorithm for the cellphone application. We implemented two algorithms on top of the prediction algorithm.

1) One application takes the outlet name, required energy (kWh) needed to charge the vehicle, and the start of charging time as input. The output is the predicted end time of charging.
2) Another application takes the outlet name, starting time, and ending time for the charging as input. The output is the predicted amount of available energy in kWh.

The total time for the algorithm to run is about a second (less than a second for less crowded outlets), which is composed of the time to 1) run in C# under Microsoft Visual Studio 2012; 2) access the database through Microsoft SQL Server 2012; and 3) generate the output. This is well within the acceptable time for a mobile application response time. The algorithm is now running on the mobile application and is available to UCLA EV owners.

VIII. Conclusion

In this paper, we have developed an approach for fast demand prediction of the sparse time series in general and specifically EV charging outlets. We found that, in general, NN-based predictions generate better predictions than kNN and weighted kNN. We modified the dissimilarity measure in NN from standard Euclidean distance to TWDP in two ways: 1) changing Euclidean distance to (negative) dot product; and 2) adding time weightings to the dissimilarity measure so that recent similar indices in time series get more weight than older ones. Each of these modifications improves the accuracy by itself and their combination improves the results more.

We have implemented this method in the cellphone application system that is used by UCLA EV owners.

Acknowledgment

The authors would like to thank C.-Y. Chung for his input.

References

[1] S. A. Dudani, "The distance-weighted k-nearest neighbor rule," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-6, no. 4, pp. 325–327, Apr. 1976.
[2] J. Zavrel, "An empirical re-examination of weighted voting for K-NN," in *Proc. 7th Belg.-Dutch Conf. Mach. Learn.*, Tilburg, Netherlands, 1997, pp. 139–148.
[3] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statist. Surv.*, vol. 4, pp. 40–79, 2010.

[4] R. Hyndman. (2010, Oct. 4). [Online]. Available: http://robjhyndman.com/hyndsight/crossvalidation/
[5] M. Kintner-Meyer, K. Schneider, and R. Pratt, "Impacts assesment of plugin hybrid vehicles on electric utilities and regional U.S. power grids Part 1: Technical analysis," Pacific Northwest National Laboratory, 2007 [Online]. Available: http://energytech.pnnl.gov/publications/pdf/phev_feasibility_analysis_part1.pdf
[6] M. Majidpour and W. P. Chen, "Grid and schedule constrained electric vehicle charging algorithm using node sensitivity approach," in *Proc. Int. Conf. Connect. Veh. Expo (ICCVE)*, 2012, pp. 304–310.
[7] C. Chung, A. Shepelev, C. Qiu, C. Chu, and R. Gadh, "Design of RFID mesh network for electric vehicle smart charging infrastructure," in *Proc. IEEE Int. Conf. RFID-Technol. Appl. (RFID-TA)*, Johor Bahru, Malaysia, Sep. 4–5, 2013, pp. 1–6.
[8] S. Mal, A. Chattopadhyay, A. Yang, and R. Gadh, "Electric vehicle smart charging and vehicle-to-grid operation," *Int. J. Parallel Emergent Distrib. Syst.*, vol. 27, no. 3, pp. 249–265, Mar. 2012.
[9] C. Chung *et al.*, "Safety design for smart electric vehicle charging with current and multiplexing control," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Vancouver, Canada, Oct. 21–24, 2013, pp. 540–545.
[10] W. Su, H. Eichi, W. Zeng, and M. Y. Chow "A survey on the electrification of transportation in a smart grid environment," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 1–10, Feb. 2012.
[11] Alternative Fuels Data Center, U.S. Department of Energy. (2013, Oct.) [Online]. Available http://www.afdc.energy.gov/fuels/stations_counts.html
[12] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Rev.*, vol. 29, pp. 594–621, 2010.
[13] T. G. Dietterich, "Machine learning for sequential data: A review," *Springer*, vol. 2396, pp. 15–30, 2002.
[14] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. Hoboken, NJ, USA: Wiley, 2013.
[15] J. D. Hamilton, *Time Series Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1994.
[16] K. N. Kumar, P. H. Cheah, B. Sivaneasan, P. L. So, and D. Z. W. Wang, "Electric vehicle charging profile prediction for efficient energy management in buildings," in *Proc. IEEE Conf. Power Energy*, 2012, pp. 480–485.
[17] A. Aabrandt *et al.*, "Prediction and optimization methods for electric vehicle charging schedules in the EDISON project," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Tech. Conf.*, 2012, pp. 1–7.
[18] R. Shankar and J. Marco, "Method for estimating the energy consumption of electric vehicles and plug-in hybrid electric vehicles under real-world driving conditions," *IET Intell. Transp. Syst.*, vol. 7, no. 1, pp. 138–150, Mar. 2013.
[19] A. Ashtari, E. Bibeau, S. Shahidinejad, and T. Molinski "PEV charging profile prediction and analysis based on vehicle usage data," *IEEE Trans. Smart Grid*, vol. 3, no. 1, pp. 341–350, Mar. 2012.
[20] M. Alizadeh, A. Scaglione, J. Davies, and K. S. Kurani, "A scalable stochastic model for the electricity demand of electric and plug-in hybrid vehicles," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 848–860, Mar. 2014.
[21] Y. Q. Li, Z. H. Jia, F. L. Wang, and Y. Zhao, "Demand forecast of electric vehicle charging stations based on user classification," *Appl. Mech. Mater.*, vol. 291–294, pp. 855–860, 2013.
[22] J. Wang *et al.*, "Electric vehicle charging station load forecasting and impact of the load curve," *Appl. Mech. Mater.*, vol. 229–231, p. 853, 2012.
[23] G. Seymour, *Predictive Inference*. London, U.K.: Chapman and Hall, 1993.
[24] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, vol. 2, pp. 1137–1143.
[25] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006, vol. 1.
[26] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Stat.*, vol. 46, no. 3, pp. 175–185, 1992.
[27] G. Bontempi, S. Ben Taieb, and Y. Le Borgne, "Machine learning strategies for time series forecasting," in *Business Intelligence*. Berlin, Germany: Springer-Verlag, 2013, pp. 62–77.
[28] D. M. Allen, "The relationship between variable selection and data augmentation and a method for prediction," *Technometrics*, vol. 16, no. 1, pp. 125–127, 1974.
[29] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7067–7083, 2012.
[30] M. Majidpour *et al.*, "Fast demand forecast of electric vehicle charging stations for cell phone application," in *Proc. IEEE/PES Gen. Meeting*, Washington, DC, USA, Jul. 27–31, 2014, to be published.
[31] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," *Autom. Remote Control*, vol. 25, pp. 821–837, 1964.
[32] M. Sechilariu, B. Wang, and F. Locment "Building integrated photovoltaic system with energy storage and smart grid communication," *IEEE Trans. Ind. Electron.*, vol. 60, no. 4, pp. 1607–1618, Apr. 2013.
[33] F. Kennel, D. Gorges, and S. Liu "Energy management for smart grids with electric vehicles based on hierarchical MPC," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1528–1537, Aug. 2013.
[34] C. E. Borges, Y. K. Penya, and I. Fernandez "Evaluating combined load forecasting in large power systems and smart grids," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1570–1577, Aug. 2013.
[35] E. Terciyanlı *et al.*, "Enhanced nationwide wind-electric power monitoring and forecast system," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1171–1184, May 2014.
[36] C. Chen and S. Duan, "Optimal integration of plug-in hybrid electric vehicles in microgrids," *IEEE Trans. Ind. Informat.*, vol. 10, no. 3, pp. 1917–1926, Aug. 2014.
[37] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms." *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.
[38] M. Majidpour, C. Qiu, P. Chu, R. Gadh, and H. Pota, "A novel forecasting algorithm for electric vehicle charging stations," in *Proc. Int. Conf. Connected Veh. Expo (ICCVE'14)*, Vienna, Austria, 2014, pp. 1–6.
[39] G. Zhang, H.-X. Li, and M. Gan "Design a wind speed prediction model using probabilistic fuzzy system," *IEEE Trans. Ind. Informat.*, vol. 8, no. 4, pp. 819–827, Nov. 2012.
[40] C. S. Ioakimidis, L. J. Oliveira, and K. N. Genikomsakis "Wind power forecasting in a residential location as part of the energy box management decision tool," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2103–2111, Nov. 2014.
[41] J. C. Ferreira, V. Monteiro, and J. L. Afonso, "Vehicle-to-anything application (V2Anything App) for electric vehicles," *IEEE Trans. Ind. Informat.*, vol. 10, no. 3, pp. 1927–1937, Aug. 2014.
[42] A. Khosravi and S. Nahavandi, "Load forecasting using interval type-2 fuzzy logic systems: Optimal type reduction," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1055–1063, May 2014.

Author's photographs and biographies not available at the time of publication.