Towards Accurate Electricity Load Forecasting in Smart Grids

Zeyar Aung, Mohamed Toukhy
Computing and Information Science Program
Masdar Institute of Science and Technology
Abu Dhabi, PO Box 54224, United Arab Emirates
e-mails: {zaung, mahmed}@masdar.ac.ae

John R. Williams, Abel Sanchez, Sergio Herrero
Engineering Systems Division
Massachusetts Institute of Technology (MIT)
Cambridge, MA 02139, United States of America
e-mails: {jrw, doval, sherrero}@mit.edu

***Abstract*—Smart grids, or intelligent electricity grids that utilize modern IT/communication/control technologies, become a global trend nowadays. Forecasting of future grid load (electricity usage) is an important task to provide intelligence to the smart gird. Accurate forecasting will enable a utility provider to plan the resources and also to take control actions to balance the supply and the demand of electricity. In this paper, our contribution is the proposal of a new data mining scheme to forecast the peak load of a particular consumer entity in the smart grid for a future time unit. We utilize least-squares version of support vector regression with online learning strategy in our approach. Experimental results show that our method is able to provide more accurate results than an existing forecasting method which is reported to be one of the best. Our method can provide 98.4–98.7% of average accuracy whilst the state-of-the-art method by Lv *et al.* is able to provide only 96.7% of average accuracy. Our method is also computationally efficient and can potentially be used for large scale load forecasting applications.**

Keywords*—*smart grids; data mining; load forecasting; regression analysis; support vector machines.

I. INTRODUCTION

In this section, we will briefly introduce the concept of a smart grid, discuss the problem of electricity load forecasting in the smart grid and the prior arts to solve it, and outline our proposed solution to the problem.

A. Smart Grid

A smart grid is an advanced electricity transmission and distribution network (grid) that utilizes information, communication, and control technologies to improve economy, efficiency, reliability, and security of the grid [23]. Nowadays, it is a priority of many governments worldwide to replace/upgrade their several decades old electricity grids with smart grids. For example, in 2010, the US government spent $7.02B on its smart grid initiative, while the Chinese government used $7.32B for its smart grid program [20].

The smart grid is characterized by several new trends and features: smart meters, demand response mechanisms, online customer interactions though PCs/mobile devices, dynamic electricity tariffs, online billing, incorporation of renewable energy generation (such as solar and wind energy) and electric vehicles, more reliable power transmission and distribution, dynamic load balancing, better power quality, better power security, etc. The architecture and components of the smart grid are illustrated in Figure 1.

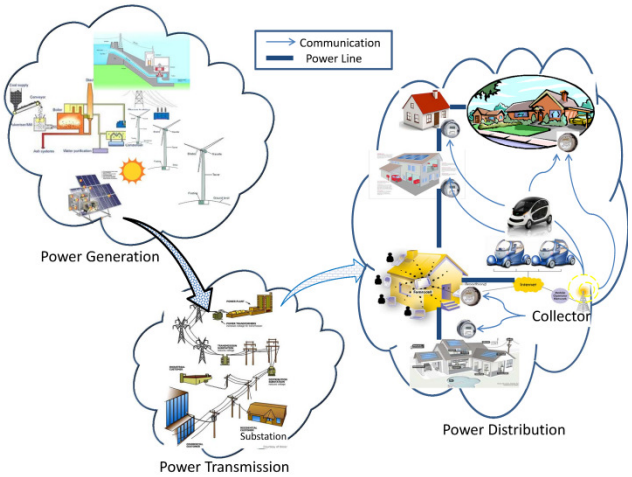

Figure 1. Architecture and components of the smart grid (reproduced with permission from [9]).

Information technology (IT) is one of the major driving forces behind a smart grid, and various IT systems and techniques such as artificial intelligence, high performance computing, simulation and modeling, data network management, database management, data warehousing, and data mining are to be used to facilitate smooth running of the smart grid [2].

B. Load Forecasting Problem

Grid load forecasting is an important task to provide intelligence to the smart grid. Accurate forecasting will enable a utility provider to plan the resources like fuel in advance and also to take control actions like switching on/off demand response appliances and revising electricity tariffs, etc.

In our research, we will focus on a specific problem of forecasting the "peak load" (i.e., the maximum electricity usage) of a particular consumer entity for a future time unit. The consumer entity in question can be of various

 ISBN: 978-1-61208-185-4

granularity levels. For example, it can be a smart meter (for a household), a cluster of smart meters (for a neighborhood), a power substation (for a town or city), or a power station (for an entire grid covering a large geographical area). Similarly, the time unit in question can be of different lengths. It can be 5 minutes, 15 minutes, 1 hour, 1 days, 1 week, etc.

In this paper, we will study a system to forecast the daily peak loads of individual smart meters. However, it should be noted that the same principles and techniques used in our studies are generally applicable to any load forecasting problems with any combinations of consumer entities and time granularities.

Researchers have been trying to solve the problem of electricity load forecasting since 1990's [1]. A number of methods based on different techniques such as time series analyses (like autoregressive integrated moving average (ARIMA) method [5]), fuzzy logic [14], neuro-fuzzy method [8], artificial neural network (ANN) [3], and support vector regression (SVR) [11] have been proposed.

Among these various techniques, support vector regression (SVR) is one of the latest developments. In [11], it was demonstrated that SVR could provide better results than the older methods like artificial neural network (ANN) [3] could.

C. Outline of Proposed Solution

As in the previous work [3] and [11], we try to approach the problem of smart grid's load forecasting from the data mining perspective. In particular, we propose a peak load forecasting model based on the data mining technique of support vector regression (SVR) using least squares [15]. More specifically, we use the online greedy least-squares SVR proposed by Engel *et al.* [7].

In order to predict the peak load P_d of a particular day d, we can roughly consider P_d as a non-linear combination of a number of attributes from different sources: peak loads of previous N days, average temperatures of previous N days, holiday records of previous N days, forecasted temperature of day d, and whether day d is a holiday (weekend or public holiday).

So, for each "target" peak load value in the historical record, we construct a "feature vector" covering the abovementioned attributes associated with the target. Then, we train our least-square SVR system using a set of <*feature vector, target*> pairs for a large enough number of days. The result of this training process is a least-squares regressor model.

We can use the resultant regressor model to forecast the peak load value P_d of a given day d. For that, we have to construct a feature vector for the day d in the same manner as in the training step. In constructing the feature vector, we need to know the forecasted temperature of the day d (if it is in the future) and whether it is a holiday (which can be easily known in advance). Then, the feature vector of day d is supplied to the regressor model to generate the forecasted peak load value of that day.

After the day d is already passed and its actual peak load value (the target) already known, the regressor model is updated with the <*feature vector, actual target*> pair for the day d, thus resulting in a fresh model which best reflects the latest trend of events.

A schematic representation of our proposed load forecasting scheme is given in Figure 2.

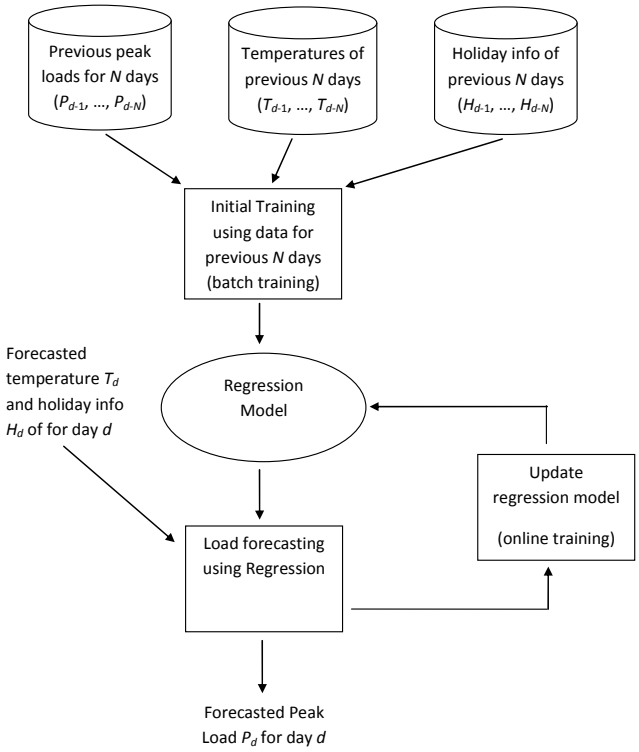

Figure 2. Overview of the proposed load forecasting scheme.

We tested our proposed method using the smart metering data from the two regions in Germany for the year 2009. Experimental results demonstrate that our approach is both accurate and computationally efficient.

The remaining of the sections is organized as follows. Section II describes our proposed load forecasting method using online least-squares support vector regressions in details. Section III presents the experimental results of the proposed method in comparison with a state-of-the-art method. Section IV discusses the future work and concludes the paper.

II. METHOD DETAILS

Support vector machine (SVM) [16] is a kind of maximum margin classifier which was originally proposed to solve the problem of binary classification. Among a large number of training data vectors, only a few are selected as "support vectors" that define the maximum margin. Only

the support vectors are utilized in predicting the classes of the testing data vectors, thus leading to a good generalization.

Later, it was realized that SVM can be adapted to solve the problem of regression [13]. Suykens and Vandewalle proposed a least-squares version of support vector regression (SVR) [15] which is particularly suitable to solve regression problems in time series data. The least-squares SVR tries to find the solution by solving a set of linear equations instead of a convex quadratic programming for classical SVMs.

A brief description of the least-squares SVR is given below in Subsections A, B, and C. This description is adapted and modified from the original ones given in [10], [15], and [19].

A. General SVM Formulation

Suppose we have a training set of n samples $\{x_i, y_i\}$ $(i = 1, \cdots, n)$ with input data vector $x_i \in \mathbb{R}^m$ (where m is the dimensionality of x_i) and corresponding binary class labels $y_i \in \{-1, +1\}$. In Vapnik's original formulation [16], the SVM classifier is defined by the conditions:

$$\left.\begin{array}{ll} w.\phi(x_i) + b \geq 1, & \text{if } y_i = +1 \\ w.\phi(x_i) + b \leq -1, & \text{if } y_i = -1 \end{array}\right\} \quad (1)$$

which can be rewritten as a single condition:

$$y_i(w.\phi(x_i) + b) \geq 1, \qquad i = 1, \cdots, n \quad (2)$$

where $\phi(x)$ is a nonlinear mapping function of a vector from original space to the high (possibly infinite) dimensional space, w is a weight vector which defines the separation hyperplane, and b is an offset of the separation hyperplane from the origin $(0, 0)$.

If the given data set is inseparable (i.e., separating hyperplane does not exist), a slack variable ξ_i is introduced in such a way that:

$$\left.\begin{array}{ll} y_i(w.\phi(x_i) + b) \geq 1 - \xi_i & i = 1, \cdots, n \\ \xi_i \geq 0, & i = 1, \cdots, n \end{array}\right\} \quad (3)$$

By applying the structural risk minimization principle, the risk bound (i.e., learning error) of the classifier can be minimized by solving the following minimizing problem:

$$\min J_1(w, \xi) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i \quad (4)$$

subject to the constraints:

$$\begin{array}{ll} y_i[w.\phi(x_i) + b] \geq 1 - \xi_i, & i = 1, \cdots, n \\ \xi_i \geq 0, & i = 1, \cdots, n \end{array}$$

where C is the slack penalty parameter to control the net effect of the slack variables.

In order to remove the complex constraints of the above minimization problem in Equation (4), we introduce Lagrangian multipliers $\alpha_i \geq 0$ $(i = 1, \cdots, n)$ [4]. Thus, the minimization problem becomes:

$$\min L_1(w, \xi, \alpha) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \alpha_i (y_i(w.\phi(x) + b) - 1 + \xi_i) \quad (5)$$

subject to the constraints:

$$\xi_i \geq 0, \quad i = 1, \cdots, n$$

The optimal point will in the saddle point of the Lagrangian function. Thus, we have:

$$\left.\begin{array}{l} \frac{\partial L_1}{\partial w} = 0 \implies w = \sum_{i=1}^{n} \alpha_i \phi(x_i) \\ \frac{\partial L_1}{\partial b} = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \frac{\partial L_1}{\partial \xi_i} = 0 \implies 0 \leq \alpha_i \leq C, \qquad i = 1, \cdots, n \end{array}\right\} \quad (6)$$

By substituting w by its expression, we get the following quadratic programming problem:

$$\max Q_1(\alpha) = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (7)$$

subject to the constraints:

$$0 \leq \alpha_i \leq C, \qquad i = 1, \cdots, n$$

Here, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ is called the kernel function (which will be elaborated below in Section II.C). By solving this quadratic programming problem subject to the constraints, we will get the separating hyperplane in the high dimensional space, that is, the classifier in the original space.

B. Least Squares SVM Formulation

Suykens and Vandewalle derived the least squares version of the SVM classifier by reformulating the minimization problem as below [15]:

$$\min J_2(w, b, e) = \mu \left(\frac{1}{2} \| w \|^2 \right) + \zeta \left(\frac{1}{2} \sum_{i=1}^{n} e_i^2 \right) \quad (8)$$

subject to the equality constraints:

$$y_i(w.\phi(x_i) + b) = 1 - e_i, \qquad i = 1, \cdots, n$$

The least-squares SVM classifier formulation above implicitly corresponds to a regression interpretation with binary targets $y_i = \pm 1$.

Both μ and ζ are parameters to tune the amount of regularization versus the sum squared error. The solution does only depend on the ratio $\gamma = \mu / \zeta$, therefore the original formulation uses only γ as tuning parameter. Therefore, we have:

$$\min J_2(w, b, e) = \frac{1}{2} \| w \|^2 + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 \tag{9}$$

The solution of the least-squares regressor is obtained after the Lagrangian function is constructed as follows:

$$\begin{aligned} & L_2(w, b, e, \alpha) \\ & = J_2(w, b, e) - \sum_{i=1}^{n} \alpha_i \{ y_i(w.\phi(x_i) + b) - 1 + e_i \} \\ & = \frac{1}{2} \| w \|^2 \\ & \quad + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n} \alpha_i \{ y_i(w.\phi(x_i) + b) - 1 + e_i \} \end{aligned} \tag{10}$$

where $\alpha_i \in \mathbb{R}$ $(i = 1, \cdots, n)$ are the Lagrange multipliers. Again, the conditions for optimality are:

$$\begin{aligned} & \frac{\partial L_2}{\partial w} = 0 \implies w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \\ & \frac{\partial L_2}{\partial b} = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \frac{\partial L_2}{\partial e_i} = 0 \implies \alpha_i = \gamma e_i, \qquad i = 1, \cdots, n \\ & \frac{\partial L_2}{\partial \alpha_i} = 0 \implies y_i(w.\phi(x_i) + b) - 1 + e_i = 0, \\ & \qquad\qquad\qquad\qquad\qquad\qquad i = 1, \cdots, n \end{aligned} \tag{11}$$

By the elimination of w and e, we will have a linear programming problem instead of a quadratic programming one:

$$\begin{bmatrix} 0 & y^T \\ y & \Omega + \gamma^{-1} I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_n \end{bmatrix} \tag{12}$$

where $y = [y_1, \cdots, y_n]$, $1_n = [1, \cdots, 1]$, $\alpha = [\alpha_1, \cdots, \alpha_n]$, and I_n is an $n \times n$ identity matrix.

Here $\Omega \in \mathbb{R}^{n \times n}$ is the kernel matrix whose individual element $\Omega_{i,j}$ $(i, j = 1, \cdots, n)$ is defined as follows [6], [10].

$$\Omega_{i,j} = \phi(x_i.x_j) = K(x_i, x_j) \tag{13}$$

C. Kernel Function

For the kernel function $K(\bullet,\bullet)$ one typically has the following choices [10], [19]:

Linear kernel:

$$K(x_i, x_j) = x_i.x_j \tag{14}$$

Polynomial kernel of degree p:

$$K(x_i, x_j) = \left(\frac{x_i.x_j}{c} \right)^p \tag{15}$$

Radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp\left(- \frac{\| x_i - x_j \|^2}{\sigma^2} \right) \tag{16}$$

Multi-layer Perceptron (MLP) kernel:

$$K(x_i, x_j) = \tanh(k\, x_i.x_j + \theta) \tag{17}$$

where p, c, σ, k and θ are constants. Here, Mercer's condition holds for all $c, \sigma \in \mathbb{R}^+$ and $p \in \mathbb{N}$ values in the polynomial and RBF case, but not for all possible choices of k and θ in the MLP case. The scale parameters c, σ and k determine the scaling of the inputs in the polynomial, RBF and MLP kernel functions. This scaling affects the kernel's bandwidth, which is an important factor in generalization of a kernel method [10], [19].

D. Online Learning

In our proposed method, we employ a version of least-squares SVR, namely the online greedy SVR proposed by Engel *et al.* [7]. In this online learning setup, we first train our least-squares SVR system with a large enough set of data in a batch mode. Then, we deploy the system for regressing (forecasting) an unknown future data.

When the actual value of the forecasted data is came to know, the SVR system is updated using this actual data. In this way, the SVR system is always up-to-date and can truthfully represent the latest trend of the data. Therefore, online learning enables us to reduce the effect of the "concept drift" phenomenon [18] which usually occurs in time-series data, and thus improving the accuracy of forecasting.

 ISBN: 978-1-61208-185-4

E. Feature Vector Construction

In order to forecast the peak load P_d of a given day d, we construct (encode) a feature vector with 32 attributes as listed in Table I. These 32 attributes are empirically chosen.

TABLE I. FEATURE VECTOR USED IN FORECASTING THE PEAK LOAD P_d OF A GIVEN DAY d.

Attrib -ute ID	Feature Description	Formula
1 to 28	Peak load of previous 28 days	P_{d-1} to P_{d-28}
29	Average peak load of previous 7 days	$1/7\ (P_{d-1} + P_{d-2} + \ldots + P_{d-7})$
30	Average temperature of previous 7 days	$1/7\ (T_{d-1} + T_{d-2} + \ldots + T_{d-7})$
31	Forecasted average temperature of the day d	T_d
32	Whether the day d is a holiday (weekend or public holiday)	H_d

The individual and the average peak load information can be obtained from the given peak load data set itself. The historical temperature information for different regions of the world can be extracted from the Weather Underground website [24]. Holiday information of countries all over the world is available from the Holidays-Info.com website [22].

We use the scaling facility of the LibSVM software [21] to map the values of each attribute into the range of –1 to +1. This scaling exercise helps us improve the forecasting accuracy by a considerable extent. The experimental results of scaling vs. without scaling are discussed in Section III.D.

III. EXPERIMENTAL RESULTS

In this section, we will discuss about the datasets that we use in our experiment, how the datasets are of training and testing subsets, the results that our proposed method achieved in comparison with an existing state-of-the-art method, the effects of scaling vs. non-scaling, and finally the computational efficiency of the method.

A. Datasets

We use two datasets in our experiment. The first one is the electricity usage data for the year 2009 logged by a smart meter deployed in a household in Lower Saxony (LS) region of Germany. We will call this dataset as **LS Dataset**. The second one is the data for 2009 logged by a smart meter installed in a household in North Rhine-Westphalia (NRW) region of Germany. We will name this dataset as **NRW Dataset**.

Each original datasets contains the electricity usage readings of the smart meter at every 15 minutes. From these readings, we extract the peak load (i.e., the maximum reading) for each day. The daily peak load profile of LS Dataset for the whole year of 2009 is illustrated in Figure 3.

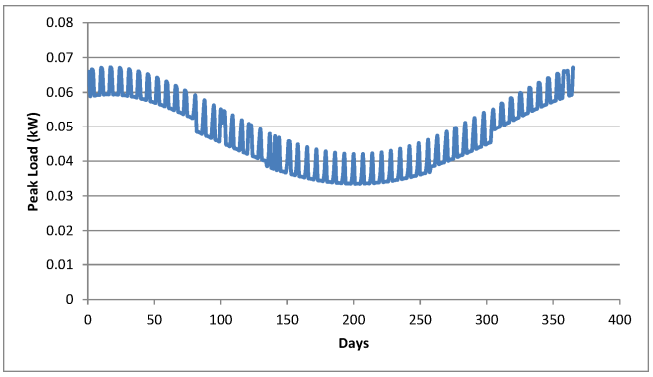

Figure 3. Daily peak load profile of an individual household in Lower Saxony (LS), Germany in 2009. The spikes indicate the increases of peak loads on holidays. Electricity usage is higher in winter and lower in summer.

B. Training and Testing

For each dataset, we use each daily peak load value and its associated feature vector (as discussed in Section III) from February 01, 2009 to June 30, 2009 (150 days) as the training data for our forecasting system. (Note: we simply cannot start from January 01, 2009 because we need the data for the previous 4 weeks, i.e., 28 days, to construct a feature vector.)

The remaining days of the year from July 01, 2009 to December 31, 2009 (184 days) are used for testing (as well as for model updating in our online learning setup).

We use dlib C++ library [17] for the implementation of online greedy least-squares SVR algorithm by Engel *et al.* [7]. We use a radial basis function (RBF) kernel, which is described in Equation (16), with the kernel scaling parameter $\sigma = 15$, which is empirically determined.

C. Results

We compare the accuracy performance of our proposed method with another least-square SVR-based method by Lv *et al.* [11], which uses a different feature vector encoding. A RBF kernel is also used for it with the parameter $\sigma = 18$, which is the optimum for that method. To enable a fair comparison, their regression model is also re-trained after every test instance in order to ensure an up-to-date model.

The forecasted peak load values for 184 test days from July 01, 2009 to December 31, 2009 are computed using both methods and are compared against the actual peak load values.

An example of the forecasted values by the two methods and the actual values for the month of December 2009 for LS Dataset are demonstrated in Figure 4. It can be observed

 ISBN: 978-1-61208-185-4

from the figure that our method can predict daily peak loads more accurately than the method by Lv *et al.*

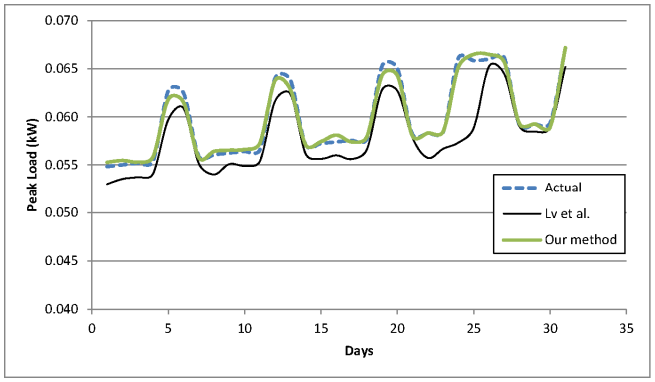

Figure 4. Example of forecasted results. The actual peak loads and the forecasted values by Lv et al. [11] and our method for the period of December 1 to 31, 2009 on LS Dataset.

In order to systematically analyze the performance of the two methods, we use two criteria: *relative error* and *accuracy* in our experiment. For each testing day, the relative error and the accuracy of the forecasted peak load are calculated as follows.

$$relative\ error = \frac{|actual\ peak\ load - forecasted\ peak\ load|}{actual\ peak\ load} \times 100\% \quad (18)$$

$$accuracy = 100 - relative\ error \quad (19)$$

For the testing period of 184 days, the comparisons of relative error values of the two methods are given in Figure 5 for LS Dataset and Figure 6 for NRW Dataset respectively. We can visually observe from the figures that our proposed method provides lower relative errors than the method by Lv *et al.* in a majority of cases.

For LS Dataset, the average relative error of our method is 1.3% (i.e., 98.7% average accuracy) whilst that of Lv *et al.* is 3.3% (i.e., 96.7% average accuracy).

For NRW Dataset, the average relative error of our method is 1.6% (i.e., 98.4% average accuracy) whilst that of Lv *et al.* is still 3.3% (i.e., 96.7% average accuracy).

Load forecasting is quite a mature technology in which many methods are able to provide an accuracy level of ~95% in general [11]. For example, the method by Lv *et al.* provides 96.7% of accuracy in our experiment. Here in our research, we are able to further improve the forecasting accuracy to the level of 98.4–98.7%. Although the 1.7–2.0% increase in accuracy may be small in absolute terms, this achievement is non-trivial. It is a common phenomenon that improving the performance of a particular technology is quite difficult when it becomes mature (i.e., when the higher plateau in its technology S-curve is reached) [12].

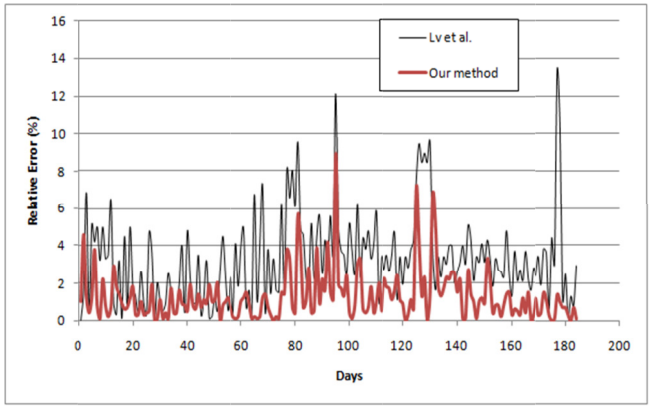

Figure 5. Relative error performances of Lv et al. [11] and our method for the period of June 1, 2009 to December 31, 2009 on LS Dataset.

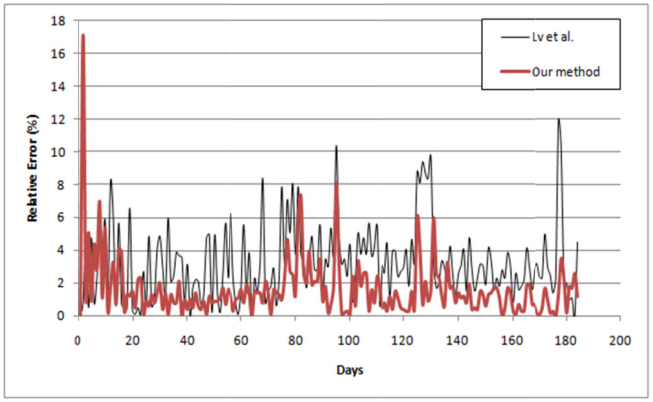

Figure 6. Relative error performances of Lv et al. [11] and our method for the period of June 1, 2009 to December 31, 2009 on NRW Dataset.

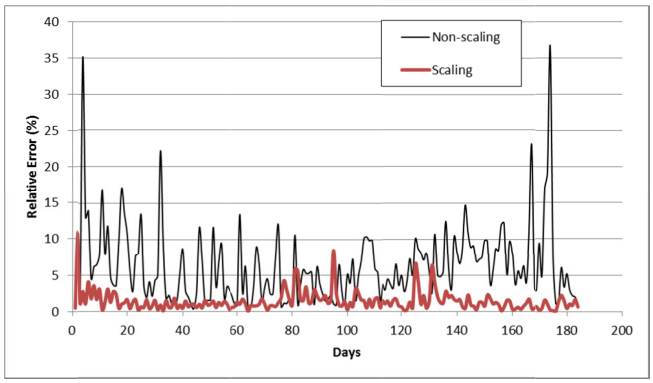

Figure 7. Relative error performances of scaling vs. non-scaling approaches in our method. The results are the averages of those for LS and NRW Datasets for the period of June 1, 2009 to December 31, 2009.

D. Scaling vs. Non-scaling

As mentioned in Section II.E, we use the scaling facility of LibSVM [21] to map the values of each of our 32 attributes into the range of –1 to +1. Scaling helps

significantly reduce the relative error (i.e., improve the accuracy) of our proposed method over directly using the data without any scaling. We can clearly observe this trend in Figure 7. The overall average of the relative errors for both LS and NRW Datasets with scaling is 1.4% while that without scaling is 6.2%. The superiority of scaling is because it prevents the dominance of attributes with larger value ranges over those with smaller ranges in calculating the Lagrange multipliers α in Equation 12.

E. Computational Efficiency

The proposed method is developed in C++ and tested on a modest laptop PC with Intel Core Duo 1.83 GHz processor and 2GB of main memory running Windows Vista 32-bit Edition. The program is compiled with Microsoft Visual C++ 2008 using O2 optimization.

The method is found to be quite efficient and scalable. The overall running time of the training for 150 days and the testing (and re-training) for 184 days is only 210 milliseconds for LS Dataset and 220 milliseconds for NRW Dataset respectively. Thus, our proposed method can be potentially deployed in a larger scale to forecast the loads of tens of thousands of consumer entities like smart meters on a distributed computing platform.

IV. Conclusion and Future Work

In this paper, we have presented an accurate load forecasting method which can potentially provide greater intelligence (smartness) to the upcoming smart grids. In our approach, we adopt the least-squares support vector regression technique incorporated with online learning. Experimental results show that our method is able to provide more accurate results than an existing forecasting method by Lv *et al.* [11], which is reported to be one of the best methods, and is also computationally efficient. As the future work, we intend to explore the idea of automatic feature selection for our regression model in order to further improve its accuracy. In addition, we plan to rigorously test our method with multiple smart grid load datasets from different countries/industries and fine tune the method so as to ensure its general usability. Finally, we hope our proposed method with these future improvements can be potentially useful to utility companies in their large-scale load forecasting applications for consumer entities at any granularity levels (such as individual households, neighborhoods, towns, cities, and large geographical regions) by providing results with better precisions.

Acknowledgement

This research was sponsored by the Government of Abu Dhabi, United Arab Emirates through its funding of the MIT-Masdar Institute Collaborative Research Project on "Data Mining for Smart Grids" (award number 10CAMA1).

References

[1] G. Adams, P. G. Allen, and B. J. Morzuch, "Probability distributions of short-term electricity peak load forecasts," *International Journal of Forecasting*, vol. 7, pp. 283-297, 1991.

[2] M. Arenas-Martínez, S. Herrero-Lopez, A. Sanchez, J. Williams, P. Roth, P. Hofmann, and A. Zeier, "A comparative study of data storage and processing architectures for the smart grid," in *Proceedings of the First IEEE International Conference on Smart Grid Communications*, pp. 285-290, 2010.

[3] M. Beccali, M. Cellura, V. L. Brano, and A. Marvuglia, "Forecasting daily urban electric load profiles using artificial neural networks," *Energy Conversion and Management*, vol. 45, pp. 2879-2900, 2004.

[4] D. P. Bertsekas, *Nonlinear Programming (Second Edition)*, Athena Scientific, 1999.

[5] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting (Second Edition)*, Springer, 2002.

[6] Z. S. H. Chan, H. W. Ngan, A. B. Rad, A. K. David, and N. Kasabov, "Short-term ANN load forecasting from limited data using generalization learning strategies," *Neurocomputing*, vol. 70, pp. 409-419, 2006.

[7] Y. Engel, S. Mannor, and R. Meir, "Sparse online greedy support vector regression," in *Proceedings of the 2002 European Conference on Machine Learning*, pp. 84-96, 2002.

[8] A. Khotanzad, E. Zhou, and H. Elragal, "A neuro-fuzzy approach to short-term load forecasting in a price-sensitive environment," *IEEE Transactions on Power Systems*, vol. 17, pp. 1273-1282, 2002.

[9] D. Li, Z. Aung, J. Williams, and A. Sanchez, "Efficient authentication scheme for data aggregation in smart grid with fault tolerance and fault diagnosis," accepted for publication at *The 2012 IEEE Power and Energy Society Conference on Innovative Smart Grid Technologies*, Washington D.C., USA, January 2012.

[10] H. Liu, W. Ma, H. Liu, and Y. Hu, "Predicting price of target in acquisition by support vector machine," in *Proceedings of the 2011 International Conference on E-Business and E-Government*, pp. 1-4, 2011.

[11] G. Lv, X. Wang, and Y. Jin, "Short-Term Load Forecasting in Power System Using Least Squares Support Vector Machine," in *Proceedings of the 2006 International Conference on Computational Intelligence, Theory and Applications, 9th Fuzzy Days*, pp. 117-126, 2006.

[12] M. A. Schilling and M. Esmundo, "Technology S-curves in renewable energy alternatives: Analysis and implications for industry and government," *Energy Policy*, vol. 37, pp. 1767-1781, 2009.

[13] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199-222, 2004.

[14] K. B. Song, Y. S. Baek, D. H. Hong, and G. Jang, "Short-term load forecasting for the holidays using fuzzy linear regression method," *IEEE Transactions on Power Systems*, vol. 20, pp. 96-101, 2005.

[15] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, pp. 293-300, 1999.

[16] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

[17] http://dlib.net/ <retrieved December, 2011>

[18] http://en.wikipedia.org/wiki/Concept_drift <retrieved December, 2011>

[19] http://en.wikipedia.org/wiki/Least_squares_support_vector_machine <retrieved December, 2011>

[20] http://greenenergyreporter.com/renewables/cleantech/china-to-invest-7-3-bln-in-smart-grid-projects-in-2010/ <retrieved December, 2011>

[21] http://www.csie.ntu.edu.tw/~cjlin/libsvm/ <retrieved December, 2011>

[22] http://www.holidays-info.com/ <retrieved December, 2011>

[23] http://www.nist.gov/smartgrid/ <retrieved December, 2011>

[24] http://www.wunderground.com/ <retrieved December, 2011>

 ISBN: 978-1-61208-185-4