Symbolic Representation of Smart Meter Data

Tri Kurniawan Wijaya, Julien Eberle, Karl Aberer
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne
Switzerland
{tri-kurniawan.wijaya, julien.eberle, karl.aberer}@epfl.ch

ABSTRACT

Currently smart meter data analytics has received enormous attention because it allows utility companies to analyze customer consumption behavior in real time. However, the amount of data generated by these sensors is very large. As a result, analytics performed on top of it become very expensive. Furthermore, smart meter data contains very detailed energy consumption measurement which can lead to customer privacy breach and all risks associated with it. In this work, we address the problem on how to reduce smart meter data numerosity and its detailed measurement while maintaining its analytics accuracy. We convert the data into symbolic representation and allow various machine learning algorithms to be performed on top of it. In addition, our symbolic representation admit an additional advantage to allow also algorithms which usually work on nominal and string to be run on top of smart meter data. We provide an experiment for classification and forecasting tasks using real-world data. And finally, we illustrate several directions to extend our work further.

Categories and Subject Descriptors

E.4 [**Coding and information theory**]: Data compaction and compression—*time series*; H.2.8 [**Database Applications**]: Data mining; G.3 [**Probability and Statistics**]: Time series analysis

Keywords

time series, symbolic representation, sensor networks, data management, smart meter, classification, forecasting

1. INTRODUCTION

Smart meters have started to be widely deployed for various reasons, such as making people aware of their consumption, understanding the usage pattern of appliances [22], or enabling fine grained energy billing. All of these sensors contribute to the generation of the so called "Big Data" and also participate in the "Internet of Things". But this trend of digitalizing every single expression of our surrounding world has also some drawbacks. First, the data collected could contain personally identifiable informations that people may not want to reveal or make accessible. This is also the case with smart meters where customer habits can be extracted from highly detailed power consumption data. Secondly, for utility companies, managing a huge size of smart meter data and performing analytics on top of it requires more and more resources (time, energy, machines).

In general, only small parts of this Big Data is really relevant and contains the few information bits that are needed for the analytics at utility companies. Therefore extracting this information before storing the data or even before centralizing it may help reducing its overall size and could also be used to hide private parts.

One approach is to develop compression that can be applied at the sensor level, well known as compressive sensing methods [1], which are designed such that the sparsity of the information could help reconstruct the real values from the compressed ones. In the case of smart meters one can think about the information as the user switching devices on and off, this is very sparse compared to the amount of real values generated every second by the smart meters. But as our goal is not to reconstruct perfectly the real values, but to perform analytics on the compressed ones, we explore another compact representation, namely symbolic representation.

In this paper we present our approach for replacing large time series of smart meter data with smaller sequences of symbols that can still support statistics and machine learning algorithms for some selected purposes, such as customer segmentation or consumption forecasting. We summarize our contributions as follows:

1) we propose symbolic representation methods for time series which admit online conversion,

2) we show how to apply our symbolic representation for two analytics tasks in the energy domain (which can be generalized), namely customer segmentation and consumption forecasting, and

3) we provide experiment results using a real-world dataset.

In Section 2, we introduce our symbolic representation and conversion methods. In Section 3, we use these schemes to encode a real-world dataset and evaluate the performance of different machine learning algorithms. Then, in Section 4, we discuss the opportunities intrinsic to the encoding of sensors data and especially using symbols. Furthermore,

EDBT/ICDT '13 , March 18 - 22 2013, Genoa, Italy

since our work span from time series processing to sensor data management, we choose to put references to related works throughout the whole paper, instead of including an additional section for related work.

2. SYMBOLIC REPRESENTATION

The data generated by sensors and in particular smart meters is represented as time series of measurements at the rate given by the specific sensor. In this section, we give the formal definition of our data (time series), and describe our approach to convert the raw data into symbols.

DEFINITION 1 (TIME SERIES). *We define time series formally as $S = \{s_1, s_2, s_3, \ldots\}$. Each $s_i \in S$ is a two tuple (t_i, v_i), where t_i is a timestamp and $v_i \in \mathbb{R}$ is a measurement on time t_i. Whenever $j \leq i$, we require t_i to be no earlier than t_j.*

Our symbolic representation aims at representing a contiguous portion of a time series as a single symbol. Symbols are taken from a predefined alphabet and in our case we will consider binary numbers, such as '0','101' or '00101'. As we can consider different symbols of different size, our alphabet will only have a partial order, '0' being equal to '01', '00' and so on. To allow reconstruction of the original time series, we also build a lookup table that will match each symbol to the average real value of it corresponding range. More specifically in the case of smart meters, the lookup table is built once at the sensor level and then sent to the aggregation server before starting to send the symbolic data. We can also consider rebuilding and resending the lookup table periodically or if the distribution of the data changes too much.

Symbolic representation needs two kinds of aggregations and therefore two kinds of segmentations.

2.1 Vertical Segmentation

Vertical segmentation is a temporal aggregation over real valued time series. It aims to reduce the numerosity of the data. In principle, we can use any aggregation method, such as average, sum, maximum/minimum value, etc. In our approach, we use average.

DEFINITION 2 (AVERAGE VERTICAL SEGMENTATION). *Given time series $S = \{s_1, s_2, s_3, \ldots\}$, then $\mathcal{V}_A(S, n) = \{\bar{s}_1, \bar{s}_2, \bar{s}_3, \ldots\}$ is a new time series resulting from vertical segmentation of S which averages n values together. Each $\bar{s}_i = (\bar{t}_i, \bar{v}_i)$, where $\bar{t}_i = t_{i*n}$, and*

$$\bar{v}_i = \frac{1}{n} \sum_{(i-1)*n+1}^{i*n} v_i.$$

2.2 Horizontal Segmentation

Horizontal segmentation changes the granularity of the values a symbol can represent. Depending on the requirement, several strategies are conceivable. To allow variability in the granularity of the symbols, we build our binary symbols of variable length by recursively dividing the sub-ranges of real values, as shown on Figure 1.

DEFINITION 3 (HORIZONTAL SEGMENTATION). *We define a* table lookup *as two tuple $L = (\mathcal{A}, B)$, where $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ is our alphabet (a set of symbols) and B as a set of separators $\beta_i \in \mathbb{R}$ where $1 \leq i \leq k-1$.*

Given time series $S = \{s_1, s_2, s_3, \ldots\}$, then $\mathcal{H}(S, L) = \{\hat{s}_1, \hat{s}_2, \hat{s}_3, \ldots\}$ is a symbolic time series resulting from horizontal segmentation of S using table lookup L. Each $\hat{s}_i = (t_i, \hat{v}_i)$, where $\hat{v}_i \in \mathcal{A}$, and:

(i) if $v_i \leq \beta_1$, then $\hat{v}_i = a_1$,

(ii) if $v_i > \beta_{k-1}$, then $\hat{v}_i = a_k$, otherwise

(iii) $\hat{v}_i = a_j$, where $\beta_{j-1} < v_i \leq \beta_j$.

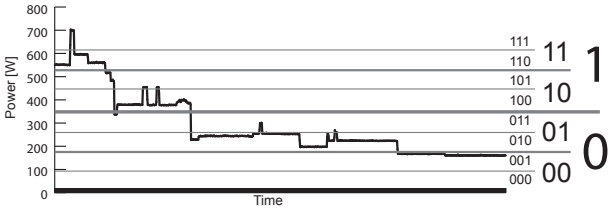

Figure 1: Construction of the variable length of symbols by recursive division of the real values range.

From Definition 3, separators and lookup tables play an important role to determine the end result of the horizontal segmentation. In this paper, we learn the *separators* using information from data distribution. We propose three different methods to generate such lookup tables, namely *uniform*, *median* and *median of distinct values*. For the division process, we propose the following methods:

a) *uniform*: each symbol is assigned ranges of values of same size. To compute them we first get the maximum value, max, and divide uniformly the range from zero to max in k subranges for each of the k symbols. Hence, $\beta_1 = \frac{max}{k}$, $max - \beta_{k-1} = \frac{max}{k}$, and $\beta_i - \beta_j = \frac{max}{k}$ for $i = j + 1$.

b) *median*: each symbol represents the same number of real values. We divide the ordered data into k equal-sized data subsets (*k-quantiles*). For the separators, we take the data values as the boundary of consecutive subsets, i.e. β_1 is the the data value in the boundary between the first and the second subset, β_2 is the boundary between the second and the third subset, and so on. This horizontal segmentation aims to maximize the entropy of the generated symbols.

c) *median of distinct values*: each symbol represents the same number of distinct real values. We take all measurement values in our time series as a set. Then, we order the elements in the set. Next, we use k-quantiles to determine the separators (similar to *median* start from this step). If the real values have enough precision to always be different this becomes equivalent to *median*, however if some value v occurs very often, this method avoid a bias toward v. We also refer to this method as *distinctmedian* for short.

Of course if the overall distribution of the real values is perfectly uniform and limited to a fixed range, these three methods are equivalent.

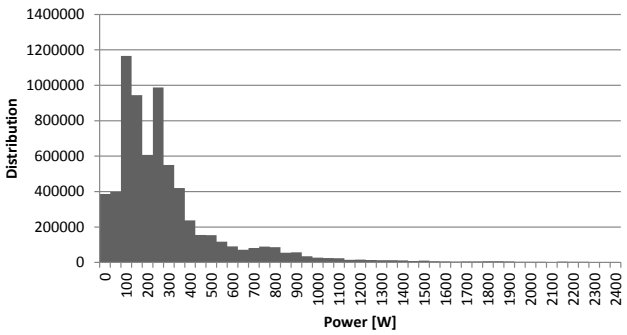

Figure 2: Distribution of Energy levels with 1 second sampling rate in the REDD dataset follows a log-normal distribution.

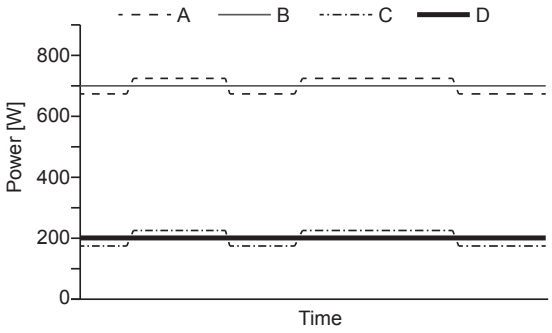

Figure 3: Without normalization A and B (C and D) are more similar, but with normalization A and C (resp. B and D) would be put together.

The most similar approaches in the literature so far are SAX [14] and iSAX [20]. But they were designed to run offline with a fixed alphabet size, $\mathcal{A}$ with a fixed k, and assuming that the distribution of v_i is Gaussian. Using this assumption, the v_i are first normalized and then separators β_i are taken at pre-defined values from a table such that they divide equally the samples. In our case the distribution is log-normal as shown on the Figure 2 and our *median* approach is a generalization of the method used for SAX. Moreover, individual normalization per house would not allow us to differentiate big consumers from the small ones as we can see in Figure 3, where consumer A and B would be normalized to the same level as C and D respectively. Therefore we need to develop other methods for generating the symbols.

On real-time systems, data is continuously produced and processing can be done either online each time a new value is measured, or by batches on past values. But, of course the algorithm cannot use future data. In our conversion process to symbols, the first horizontal segmentation has to be performed before the system can start to process any data. Indeed, the range of values assigned to a symbol cannot change too often, otherwise it would need to propagate also the new lookup table and make difficult to run any algorithm on top of the symbols as they may have different representations. Therefore, we propose to use historical data for determining the distribution of the real values and compute the separators according to the three methods described above. The time period used should be representative for the typical behavior of the measured phenomenon. In our case of power consumption measurement, it should include day and night or weekday and weekends values.

For horizontal segmentation, there exist online algorithms which are able to select dynamically when a new segment has to be started [8, 9, 15]. But in our case, variation of the time dimension of a symbol would give it a different semantics and thus preventing us to run algorithms on top of symbolic data. For this reason we chose to predefine the length of the temporal aggregation and evaluate different options.

2.3 Compression ratio

The compression ratio depends on both the vertical and the horizontal segmentation granularity. For example, if the original data is stored as *double* (64bit) and sampled at 1 Hz, we have around 680 kB of data per day. Now if we use 16 symbols and an aggregation of 15 minutes, it would leave us with only 384 bit, three order of magnitude lower. Of course the lookup table should be taken into account, but it can be amortized over time. Communication and storage overhead, induced for example by protocols and indexes should also be taken into account for a real system.

3. EXPERIMENTS

There are three smart meter datasets available which are based on real measurements:

- REDD dataset [13]: contains data of 6 houses for appliance and house level consumption. The measurement is done every second for the duration of 1 to 2 months.
- Smart* dataset [2]: contains data of 443 houses for house level consumption for the duration of 24 hours. In addition, it has three houses with more fine grained house level measurements (every second) for around 3 months, where one of them also has circuit level measurement.
- Irish CER dataset [4]: contains data of around 5000 houses for house-level consumption. The measurement is done every 30 minutes for the duration of one and a half years.

For our experiments, we decided to use REDD dataset as a tradeoff between duration and granularity. It has more fine grained measurements compared to Irish CER dataset, and it has a longer measurement duration compared to Smart* dataset.

For each of the separators generation methods presented in the previous section, we used the first two days of data for each house to compute the necessary statistics. This allows us to get an estimation of the distribution of the data, without using the complete dataset. As an illustration, Figure 4 shows the accumulative mean, median, and median of distinct values of house 1 in REDD dataset for three consecutive days. The statistics start to converge after day one.

Our symbolic encoding procedure needs two parameters, namely the temporal aggregation length and the size of the symbols alphabet.

Aggregation length The aggregation length determines the smoothing of the data and can be seen as a low-pass

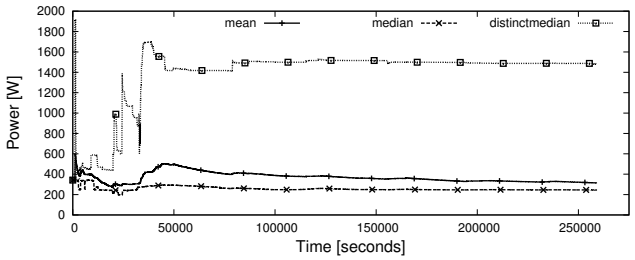

Figure 4: Statistics on energy consumption of house 1 in REDD dataset (one day = 86400 seconds).

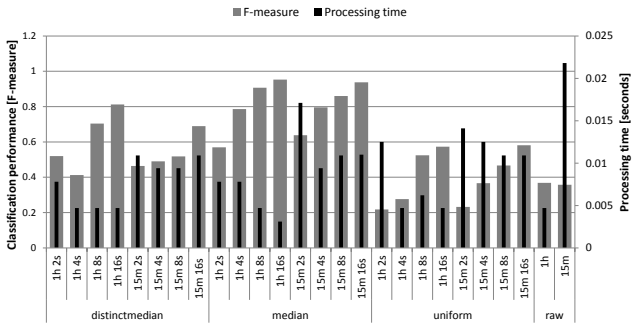

Figure 5: Evaluation of a Naive Bayes classifier over symbolic and raw data.

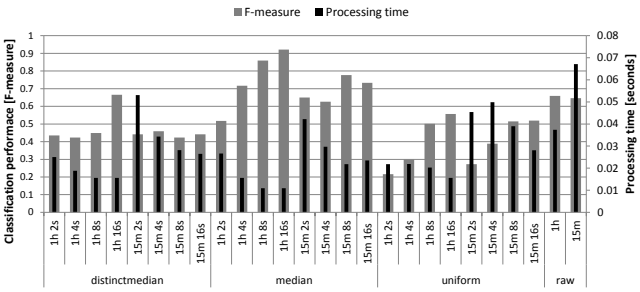

Figure 6: Evaluation of a Random Forest classifier over symbolic and raw data.

filter. It can also be applied directly on the raw values to highlight the effects of the other parameter only. We choose 15 minutes and 1 hour as they reflect the typical segmentation in smart energy algorithms (see for example [16, 23]).

Alphabet size The size of the alphabet determines the granularity of the data and the accuracy of the reconstruction. However, too many symbol variations would make us loose the advantages of efficiency in data processing and machine learning. Furthermore, as our symbols are stored as binary numbers, we used only the power of 2. In particular, we look at different values between 2 and 16.

We used the total power consumption of the house, by summing the two main power time series for each house. In the following, we look at two main scenarios of energy data management: customer segmentation and consumption forecasting.

3.1 Classification

Identifying customers having a similar consumption profile (customer segmentation) is very relevant for the future of smart grid, especially for demand response system and intelligent distribution channel. However, as we only have 6 houses in our dataset, we consider each house having its own cluster and split it by days. Then, we take some of these days for training and the rest for evaluating the classification. This follows the same assumption as for the previously presented distribution estimation, that users behave similarly over time. Interestingly this could also be seen as an attack against changing ID's privacy protection mechanisms such as the one used in [3] that could be also applied for anonymizing smart meters data.

As the dataset contains gaps (missing values), we select days where the house has "enough" data, putting the threshold at 20h per day of data and build one vector per day, the class label being the house number. The so generated files were used as input for Weka's [10] implementation of various classifiers such as Random Forest, Logistic, J48, and Naive Bayes. Another advantages of our approach is that it is not linked to any specific classifier. Hence, all algorithms supporting nominal values can be applied.

Evaluation using 10-fold cross-validation gives the results shown on Figures 5 and 6. The timing value was computed as the average over 10 runs and the classification performances are evaluated with the F-measure (the weighted harmonic mean of Precision and Recall). To have vectors of same size, raw values were also aggregated, by taking the average over 15 minutes, respectively 1 hour.

Time-wise, 15 minutes aggregation is in average slower than the corresponding 1 hour ones and the raw dataset always took slightly longer to process, mostly because it was composed of numerical values instead of symbols. The running time over the full raw vectors (actual measurements, without aggregation) was much slower by two orders of magnitude. Hence, we choose not to include it in the figures.

As expected, the classification performances vary depending on the algorithm used, but it is consistent across the different parameters. Therefore, we show only the figures for Naive Bayes and Random Forest. Accuracy improves with the size of the alphabet as more detailed information, with a better granularity, are kept during the conversion process. On the contrary, one hour aggregate performs sometimes better than the 15 minutes ones, especially with 16 symbols. This can be explained as the noise filtering effect of our low-pass filter.

The classification using raw values, however, depends much more on the algorithm and Random Forest is the one performing better. But even with this algorithm, it is not able to outperform *median* encoding performance. To check if this is due to a loss of information by aggregating values, we run also the Random Forest classifier over the full dataset by using the same day separation. However, there is only an accuracy increase of 4%. This is not enough to catch the performance of the *median* encoding.

On average, *median* encoding performs better than *distinctmedian*, which is better than *uniform*. Indeed, by choosing to maximize the representativity of a symbol in the *median* scheme, it allows classifiers to better differentiate the houses, as their encoding is based on their own statistics, adding information to the encoded data. The choice of the delimiters containing information specific to the house

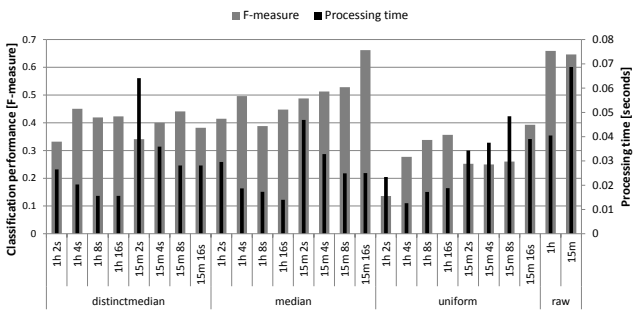

Figure 7: Evaluation of a Random Forest classifier over symbolic (using a single lookup table) and raw data.

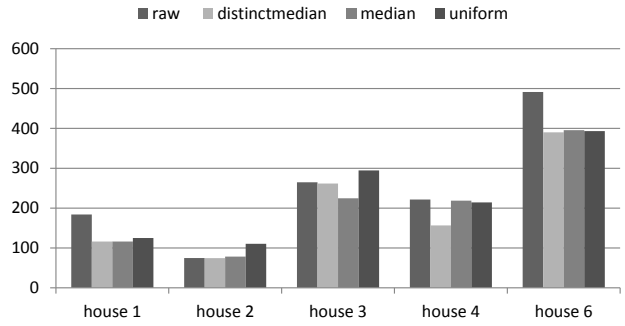

Figure 8: MAE of symbolic forecasting using Naive Bayes. Forecasting performance using raw value is shown for comparison. House 5 is skipped because there is not enough data.

(quantiles for example), the encoding will generate sequences of symbols that are typical to the considered house, thus making them more differentiable and classifiable.

To verify the hypothesis above, we ran again the encoding, but by using statistics over all houses instead. The results are show in Figure 7. Overall, the performance of classification on symbolic data is decreased. However, using Random Forest classifier, *median* encoding still manage to reach the same level as the raw values. And, using Naive Bayes classifier it even outperformed it (see Table 1). This shows that this encoding preserves all needed information for performing classification over symbolic representation of sensor data. Table 1 summarizes all our classification experiments.

3.2 Forecasting

Consumption forecasting is an important task in the smart grid for determining what action utility companies should take given the energy availability. In this experiment, we aim to do short-term hourly residential load forecasting. For residential consumption, even in the same activity, a small change in order of which appliance turned on result in totally different pattern. While consumption pattern for a community is more plausible and has been studied a lot [5, 6, 11, 18, 19, 21], this is not the case with consumption pattern for an individual house.

We show a case where we predict the next day electricity consumption of a house given 1 week historical data. When we do forecasting using symbols, it means that we forecast the next symbols. In our symbolic representation, each symbol represents a range of values. However, in the consumption forecasting task, what we need are the real consumption values (instead of symbols). For this matter, in our experiment, we define semantics of a symbol as the center of its range. At first, it seems that this exposes the disadvantage of using symbols in forecasting. But, as we can see later, performance of our symbolic forecasting is comparable with (in some cases better than) forecasting using the real values.

For each house, we take 1 week hourly consumption data as training and the next day hourly consumption data for testing. We perform symbolic forecasting using *median*, *distictmedian*, and *uniform* using alphabet of length 16. We choose this length as a trade off between symbol range's granularity and encoding length. We reduce the forecasting task into classification task using lag attributes of length 12 comprises of 12 previous symbols. The target attribute is the next symbols. For the algorithms, in principle we can use any machine learning algorithm for classification. However, in this experiment we show the result using Random Forest and Naive Bayes.

As a comparison to our symbolic forecasting performance, we also perform consumption forecasting on real values. Several techniques have been suggested in the literature, e.g., regression [5, 18], ARMA [12, 21], neural network [11], fuzzy logic [17], and support vector machine [6, 7, 19]. However, all are applied to load on distribution level. In this experiment, we use support vector machine for regression to forecast (real value) residential level consumption.

Experiment result can be seen in Figure 8 and 9. The performances are measured by *mean average error* (MAE). Although our symbolic forecasting suffers from its inability to express the real values (and use the center of their range instead), its performance is comparable to real value forecasting. In some cases, our symbolic forecasting outperform real value forecasting (as in forecasting for house 1, 4, and 6 using Naive Bayes, and house 1, 3, 6 using Random Forest). For *median*, and *distinctmedian*, their segmentation is based on values' frequency (more segment in range with more values). This helps them to give more detail on dense value ranges and be more coarse (leave out unnecessary detail) one less dense ranges. For *uniform*, although its segmentation is not based on value range frequencies, its segmentation plays a role in filtering unnecessary fluctuation which often happens in residential consumption.

Looking at the promising performance of our symbolic representation, it opens a possibility of customized segmentation with background knowledge. Consider a simple example of an expert who is interested on two segmentation: *low* and *high* consumption, where *low* means consumption below a certain threshold, and *high* means consumption above a certain threshold. This can be easily represented as a symbolic representation using alphabet of size 2.

4. DISCUSSION AND FURTHER WORK

We designed the horizontal segmentation with the goal of being flexible. Indeed, higher resolution symbols can easily be converted to lower resolution and lower resolution symbols can be compared to higher resolution ones. This allows to run most of the machine learning algorithms even if the symbolic time series have been encoded with different resolutions, or if the resolution changed in time.

We believe that the flexibility on the way we construct

Table 1: F-measure for each method with 1 hour and 15 minutes aggregation, with 2 to 16 symbols. + means that the encoding use a single lookup table for all houses.

	Random Forest	J48	Naive Bayes	Logistic	Logistic+	Random Forest+	J48+	Naive Bayes+
distinctmedian 1h 2s	0.44	0.42	0.52	0.41	0.33	0.33	0.31	0.32
distinctmedian 1h 4s	0.42	0.32	0.41	0.51	0.33	0.45	0.29	0.41
distinctmedian 1h 8s	0.45	0.56	0.70	0.43	0.35	0.42	0.47	0.51
distinctmedian 1h 16s	0.67	0.49	0.81	0.74	0.46	0.42	0.41	0.65
distinctmedian 15m 2s	0.44	0.48	0.46	0.44	0.36	0.34	0.33	0.30
distinctmedian 15m 4s	0.46	0.37	0.49	0.45	0.36	0.40	0.40	0.46
distinctmedian 15m 8s	0.42	0.33	0.52	0.46	0.49	0.44	0.29	0.54
distinctmedian 15m 16s	0.44	0.52	0.69	0.68	0.50	0.38	0.40	0.66
median 1h 2s	0.52	0.61	0.57	0.51	0.36	0.41	0.45	0.40
median 1h 4s	0.72	0.58	0.79	0.72	0.33	0.50	0.52	0.50
median 1h 8s	0.86	0.65	0.91	0.69	0.27	0.39	0.38	0.52
median 1h 16s	0.92	0.72	**0.95**	**0.95**	0.50	0.45	0.41	0.61
median 15m 2s	0.65	0.58	0.64	0.43	0.22	0.49	0.37	0.30
median 15m 4s	0.63	0.64	0.80	0.70	0.50	0.51	0.49	0.66
median 15m 8s	0.78	0.44	0.86	0.86	0.58	0.53	0.47	0.75
median 15m 16s	0.73	0.57	0.94	**0.95**	0.70	0.66	0.46	**0.82**
uniform 1h 2s	0.22	0.22	0.22	0.24	0.14	0.14	0.14	0.14
uniform 1h 4s	0.30	0.28	0.28	0.31	0.30	0.28	0.22	0.28
uniform 1h 8s	0.50	0.38	0.52	0.36	0.22	0.34	0.25	0.15
uniform 1h 16s	0.56	0.44	0.57	0.43	0.33	0.36	0.39	0.48
uniform 15m 2s	0.27	0.16	0.23	0.23	0.25	0.25	0.19	0.20
uniform 15m 4s	0.39	0.38	0.37	0.34	0.27	0.25	0.17	0.26
uniform 15m 8s	0.51	0.51	0.47	0.30	0.31	0.26	0.24	0.26
uniform 15m 16s	0.52	0.52	0.58	0.49	0.21	0.39	0.30	0.42
raw 1h	0.66	0.53	0.37	0.29	0.29	0.66	0.53	0.37
raw 15m	0.65	0.57	0.36	0.36	0.36	0.65	0.57	0.36
raw 1sec	**0.71**	0.52	0.24	-*	-*	**0.71**	0.52	0.24

*) this values is not computed due to Java heap space issues with the Logistic algorithm.

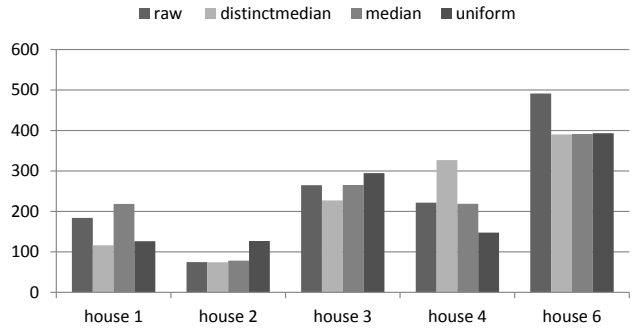

Figure 9: MAE of symbolic forecasting using Random Forest. Forecasting performance using raw value is shown for comparison. House 5 is skipped because there is not enough data.

the horizontal segmentation opens the door to much more advanced strategies for selecting on the fly the best symbol table, such as to minimize the amount of data to communicate and maximize the utility of the resulting symbols. In addition, when the consumer consumption pattern changes drastically, e.g., due to seasonal change, or having an additional family member, on the fly symbol table modification could be useful. To study the effect of seasonal change, one can consider to use Irish CER dataset which has more than one year measurement.

The notion of optimal horizontal segmentation is relative to the application, different segmentations can be optimal for different tasks. For example we have seen that using *median* to divide ranges preserves the most the entropy of the time series, making it well suitable for classification goals. But for detecting small variations, it would not perform so well as it works pretty much like a low-pass filter. Therefore we would like to look into an *utility-driven* horizontal segmentation method that could optimize the performances of a chosen analytics with predefined properties or background knowledge from experts. In the context of smart meters, however, we would have to deal with contradicting goals of customers that want to protect their privacy and the utility companies that may want to do more precise computations.

5. CONCLUSIONS

In this work, we have introduced three different methods to express smart meter data (hence, time series data) as symbols, namely: *median*, *distinctmedian*, and *uniform*. We aim to solve the data analytics problem in the smart grid due to very large data size generated by smart meter, which soon will exceed any consumption data generated so far, and its detail measurement which expose privacy risk to customers. Our symbolic representation reduce data numerosity and obscure smart meter detail measurements by representing them as symbols (where each symbol is a range of values).

Three mechanisms proposed in this paper have been shown to perform well in experiments for data analytics purposes on top of real data. Although their performances vary, depending on the algorithm used to perform the analytics task, in general they are comparable to the analytics performance of the real data. Experiment results show that the symbolic approach can be a promising direction to face the challenge of data numerosity and data privacy inherited by smart meter data. Although, we have explored some approaches in

horizontal segmentation, there is still window of opportunities to develop more advanced strategies in horizontal or vertical segmentation, including on the fly symbol table or utility-driven segmentation with background knowledge.

6. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no: 288322. We also thank Andrea Hess for useful comments and suggestions.

7. REFERENCES

[1] R. Baraniuk. Compressive sensing [lecture notes]. *Signal Processing Magazine, IEEE*, 24(4):118 –121, July 2007.

[2] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. Smart*: An open data set and tools for enabling research in sustainable homes. In *Proceedings of the 2012 Workshop on Data Mining Applications in Sustainability (SustKDD 2012)*, Aug. 2012.

[3] L. Bindschaedler, M. Jadliwala, I. Bilogrevic, I. Aad, P. Ginzboorg, V. Niemi, and J. Hubaux. Track me if you can: On the effectiveness of context-based identifier changes in deployed mobile networks. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, 2012.

[4] CER. Smart metering trial data publication, 2012.

[5] W. Charytoniuk, M. Chen, and P. Van Olinda. Nonparametric regression based short-term load forecasting. *Power Systems, IEEE Transactions on*, 13(3):725 –730, aug 1998.

[6] B.-J. Chen, M.-W. Chang, and C.-J. Lin. Load forecasting using support vector machines: a study on eunite competition 2001. *IEEE Transactions on Power Systems*, 19(4):1821 – 1830, Nov. 2004.

[7] S. Fan, K. Methaprayoon, and W.-J. Lee. Multiregion load forecasting for system with large geographical area. *IEEE Transactions on Industry Applications*, 45(4):1452 –1459, July-aug. 2009.

[8] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick. Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2232–2245, 2010.

[9] T. Guo, Z. Yan, and K. Aberer. An adaptive approach for online segmentation of multi-dimensional mobile data. In *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDE '12, pages 7–14, New York, NY, USA, 2012. ACM.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[11] H. Hippert, C. Pedreira, and R. Souza. Neural networks for short-term load forecasting: a review and evaluation. *IEEE Transactions on Power Systems*, 16(1):44 –55, Feb. 2001.

[12] S.-J. Huang and K.-R. Shih. Short-term load forecasting via arma model identification including non-gaussian process considerations. *IEEE Transactions on Power Systems*, 18(2):673 – 679, May 2003.

[13] J. Z. Kolter and M. J. Johnson. Redd: A public data set for energy disaggregation research. In *Proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, 2011.

[14] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, Oct. 2007.

[15] X. Liu, Z. Lin, and H. Wang. Novel Online Methods for Time Series Segmentation. *IEEE Trans. Knowl. Data Eng.*, 20(12):1616–1626, 2008.

[16] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE Transactions on Smart Grid*, 1(3):320 –331, Dec. 2010.

[17] S. C. Pandian, K. Duraiswamy, C. C. A. Rajan, and N. Kanagaraj. Fuzzy approach for short term load forecasting. *Electric Power Systems Research*, 76(6-7):541 – 548, 2006.

[18] A. Papalexopoulos and T. Hesterberg. A regression-based approach to short-term system load forecasting. *IEEE Transactions on Power Systems*, 5(4):1535 –1547, Nov. 1990.

[19] N. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24 –38, May 2009.

[20] J. Shieh and E. Keogh. isax: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 623–631, New York, NY, USA, 2008. ACM.

[21] J. W. Taylor. Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, 204(1):139–152, July 2010.

[22] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. Leveraging smart meter data to recognize home appliances. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 190 –197, Mar. 2012.

[23] T. K. Wijaya, K. Larson, and K. Aberer. Matching demand with supply in the smart grid using agent-based multiunit auction. In *Proceedings of The E6 (Energy in Communication, Information, and Cyber-physical Systems) workshop at COMSNETS 2013*, Jan. 2013.