Short-Term Energy Pattern Detection of Manufacturing Machines with In-Memory Databases - A Case Study

Christian Schwarz
Hasso Plattner Institute, University of Potsdam
Potsdam, Germany
Email: firstname.lastname@hpi.uni-potsdam.de

Felix Leupold, Tobias Schubotz
SAP Labs Inc.
Palo Alto, United States
Email: firstname.lastname@sap.com

***Abstract*—Todays energy companies mainly use generalized demand sets to predict the required amount of energy of their customers on a high aggregation level. This is sufficient in an energy consumption oriented power grid, having enough resources to produce and transmit the requested and produced amount of energy. With the increasing amount of renewable energy sources, the power grid evolves from a purely consumption controlled supply network to a production controlled grid. In that environment the need for detailed short term energy demand predictions increases. A first step to predict the demand of energy is to find generalizable patterns within the energy consumption data that can later on be used for early predictions on real time data. To study the possibility to predict energy patterns nearly real-time, we created an environment, where metering data is collected every second and used for real-time pattern matching. We developed and implemented pattern recognition algorithms that use the abilities of in-memory databases with the collected metering data in order to detect energy consumption patterns.**

***Keywords*-energy pattern recognition; machine learning; in-memory database;**

I. Introduction

In industries energy expenses can make up to 43% of all operational expenses. Many companies monitor their energy usage on a detailed level and reduced their energy consumption, for example by 58% in the aluminum industry since 1975 [1].

A study by the US department for energy has shown that 71% of private households changed their energy usage behavior after they got the possibility to monitor their energy consumption based on in-home displays, even if the expected savings are only between 5 to 15% [2], [3].

Today's energy companies use generalized demand sets to forecast the required amount of energy for a given period of time. The amount of data recorded within an Advanced Metering Infrastructure (AMI), that is expected to collect metering data every 15 minutes, would enable more detailed energy demand predictions [4]. In contrast, short-term demand prediction based on real-time smart metering data is not used to reduce the gap between demand and supply of energy. That gap comes with high costs and can lead to bankruptcies of energy companies in extreme cases [5]. One of the reasons is the amount of data produced within an AMI, where each smart meter produces more than 35,000 meter readings per year.

We evaluated how a new trend in the database market helps to handle the amount of data produced by an AMI: in-memory databases. In-memory databases offer the possibility to run analytical workloads on transactional data within seconds, the used column-oriented data layout is not optimal by default for write-intensive workloads like the smart grid [6]. To enable a high transactional throughput, like it occurs within an AMI, in-memory databases use techniques like write-optimized buffers and bulk loading [7].

In this paper, we will focus on short-term pattern recognition of manufacturing machines. Short-term pattern recognition in that case means, that we are able to detect usage patterns within real-time enabling the energy provider to adjust their energy production for the remaining period of the pattern. A completed consumption pattern describes an energy usage pattern that occurs when a machine creates a product. Therefore, we have installed a metering infrastructure that enables us to collect energy usage data from several devices within a one second interval. In our scenario we use the energy consumption data created by a fully automated coffee machine, as well as several other devices. We detect the consumption pattern of that machine and forecast its future short-term energy demand based on historically recorded data within the in-memory database. This coffee machine is able to produce different types of coffees creating a more or less unique energy footprint during its production period. The goal is to detect the produced coffee as soon as possible and to predict the amount of energy required within the remaining production time.

The rest of the paper is organized as follows. Section V presents related work in the area of energy data pattern recognition The used pattern recognition and prediction methods are presented in Section II, while the experimental setup is described in detail in Section III. The evaluation of our pattern recognition algorithms is presented in Section IV followed by the conclusion and outlook in Section VI.

II. Pattern Recognition

In general, the field of pattern recognition is associated with the automatic discovery of similarities in data, which is

 ISBN: 978-1-61208-189-2

achieved through machine learning. Pattern matching can be used for regression analysis and classification [8]. Regression analysis models a function from a set of data points in order to interpolate and extrapolate between this data set. Classification decides to which of n classes a given data set belongs.

We use *supervised learning methods*, a technique that takes a training set of patterns for learning and undertakes to generalize from this training set to also identify untrained patterns [9]. Given an *input vector* X the algorithm calculates an *output vector* Y. The actual class of X is called the *target vector* T. In supervised learning the algorithm goes through a *learning* phase, where it is given a set of training vectors $X = \{X_1, X_2, ..., X_i\}$ with the corresponding target vectors $T = \{T_1, T_2, ..., T_i\}$. The algorithm then tries to find a function $y_k = Y(X_k)$ so that the deviation of the output vector for each x is minimal. During our project, we implemented multiple pattern matching algorithms from which we chose three to present within this paper: *inter-quartile range coverage (IQRC)*, a *multi class support vector machine (MCSVM)* and a *k-nearest neighbor (knn)* algorithm.

A. Inter-Quartile Range Coverage

We developed the IQRC pattern matching algorithm for our scenario to classify recorded patterns. Given a set of training vectors we calculate the upper and lower quartile for each dimension of all X_i that have an identical target vector T_k, which means they lie in the same class. The range between the upper and lower quartile is called *inter-quartile range (IQR)*. Given an input vector X, we sum up the dimensions of X that lie in the IQR of the same dimension in the training patterns. This is done for each classifier and compared against a threshold. If the threshold of classifier i is exceeded the algorithm identifies the input series as product i.

For classes with a high deviation amongst vectors the IQR will be larger than for classes with a small deviation. To take that into account, the value of a dimension lying in the IQR of a class is $\frac{1}{1+\|IQR\|}$, rating the values that lie in smaller IQR higher than values that lie in a greater IQR.

This step is done for each of the products. Our algorithm will output the product that exceeds its threshold the most. The algorithm can be formalized as follows: Let $X = (x_1, x_2, ..., x_n)$ be the input vector and j_i^k be a vector with all values from the training patterns of class k in dimension i. $\delta(k)$ denotes the threshold of class k.

$$y(x) = argmax_{k \in Classes} \left((\sum_{i=0}^{n} w(x_i, j_i^k)) - \delta(k) \right) \quad (1)$$

$$\text{where } w(x_i, j_i^k) = \begin{cases} \frac{1}{Q_{.75}(j_i^k) - Q_{.25}(j_i^k) + 1}, & \text{if } x_i \in IQR(j_i^k), \\ 0, & \text{else.} \end{cases} \quad (2)$$

$$\text{and } IQR(j_i^k) = \{w | w \in j_i^k \ \wedge \ Q_{.25}(j_i^k) < w < Q_{.75}(j_i^k)\} \quad (3)$$

If more than one beverage has an IQRC above a certain threshold, we chose maximum overstepping. Due to the relatively high warp amongst patterns, it is difficult to find a threshold that is exceeded by all positive but by none of the negative examples.

To optimize the thresholds we use a modified hill climbing algorithm [10]. The threshold for all products are initially chosen so high that none of the training patterns are recognized at all. We then order the beverages descending by the number of their occurrences in the training set and start to decrement their threshold until the overall matching performance reaches a maximum. This threshold is then fixed for the product and we continue with the next product. After processing all products, we start again with the first one. Contrary to the the first pass, the threshold of all other products are now at a fairly good value. We do this until all thresholds stay constant for one round that is, the optimum does not change anymore. In our scenario that happens after four iterations.

B. Multi-Class Supported Vector Machine

We also implemented a Support Vector Machine (SVM) algorithm inside the database management system [11]. Normally a SVM only separates between two classes, but there are approaches that extend the original SVM to solve multi-class classification problems as well. The most common approach is called *one-versus-all* [12]. Given there are n classes $c_1, c_2, ..., c_i, ..., c_n$ we will create a binary SVM that is trained with all patterns from c_1 for its first target and with the rest of the patterns for the other target. This is repeated with all of the n classes. In matching an incoming pattern is passed to each SVM. Ideally only one machine detects a positive result. If there is more than one SVM classifying the input as c_i, the one with the largest result vector is used. If there is no SVM classifying the input as c_i the one with the smallest negative result vector is chosen. Assuming n classes this approach needs n iterations.

C. K-Nearest Neigbor

We have seen that patterns we collected have a considerable variance for the same target vector. The clusters of each classifier are therefore rather big and overlapping. Nonetheless, since irregularities seem to be the rule, we might find the corresponding class by looking for the pattern that is closest to the input pattern. This way even in a subspace with many patterns of class A, a pattern of class B varying from the others might still be found. This is what the knn algorithm does. In our case k is set to 1. Given an input vector X, knn returns the classification y from the trained vector $\bar{X}$ for which the squared distances $d(X, \bar{X})$ is minimal [13].

An advantage of the nearest neighbor algorithm is that it requires less computational power than the other algorithms, because the calculation of squared distances is not expensive. The training phase only creates value series and does not involve learning.

 ISBN: 978-1-61208-189-2

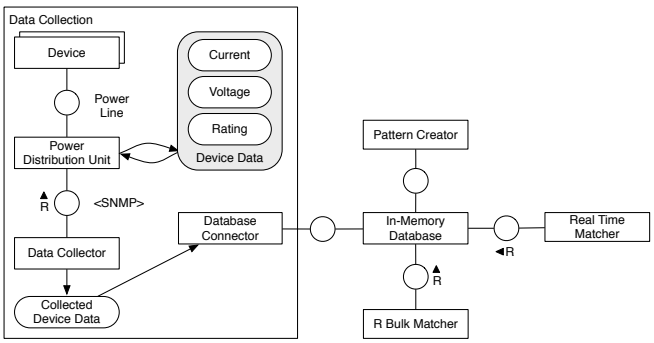

Figure 1. Experimental setup as FMC block diagram

III. Experimental Setup

To be able to perform short-time energy demand prediction detailed data like voltage, current and power consumption is necessary. These information need to be collected and stored in a database where further processing takes place. This section gives a detailed overview about the experimental setup, the data collection and energy demand prediction.

A. Data Collection

In smart grids, the (AMI) is expected to be applied for collecting necessary data. It is a multi-level architecture, where smart meter readings are transmitted from smart meters over concentrators to a central system, meaning a database. In order to detect the short energy usage patterns of our coffee machine, we increased the measuring granularity. In our experiment, we assume that every installed device can be measured independently from each other. This might not yet be the case in private households, but companies typically install multiple smart meters in order to monitor different consumers like air conditioning, lightning, office and manufacturing devices independently.

Because it is nearly impossible to get self-measuring devices, we used an Emerson Network Power Rack Power Distribution Unit (PDU) with a Liebert MPX control module which is capable of measuring the voltage, current and rating of every single power plug [14]. Depicted in Figure 1, to the PDU, we connected several devices, the PDU in turn is connected to a local area network and can be queried via the Simple Network Management Protocol (SNMP). There are in total 18 devices connected to the PDU. This setup results in an approximately 16,000 times higher data volume than the AMI proposes.

The data collector queries the PDU as often as possible to collect the current data for each device. This data is then transferred into the in-memory database. Table I depicts the used table schema for `device_readings`. Detected patterns in the stream of readings are written into the table `pattern_recognition`. The resulting transmission interval from PDU to database is between 0.5 and 2 seconds depending on the current network load.

Table I
Schema of the tables used for prediction

DEVICE_READINGS		
DEVICE_ID	:	INTEGER
DATETIME	:	INTEGER
VALUE	:	DOUBLE

PATTERN_RECOGNITION		
DATETIME	:	INTEGER
VALUE	:	DOUBLE
PRODUCT	:	VARCHAR

B. Data Processing

1) Hardware Setup: We use an instance of SAP's in-memory database with a column-oriented data layout that is best suited for analytical workloads [6]. Our implementation uses the GNU R interface of the databases development version [15]. The database is installed on a HP ProLiant DL580 G7 series server that is equipped with four Intel Nehalem X7560 CPUs and 256 GB main memory at 1066 MHz. The server runs a 64-bit version of openSUSE 11.2 (kernel 2.6.31.14).

2) Data for Training and Testing: In order to match patterns amongst the monitored energy consumption we use a set of features of that data to perform our matching algorithms on. Beside the raw data for electronic power consumption in watt hours [Wh], we also use the following more abstract features of the gathered data:

- Number of peaks
- Greatest delta
- Total amount of energy used
- Pattern duration
- Gradients values
- Moving average, and
- Histogram

3) Precision Benchmarking: We divide the pattern set into two parts, a training set with input and target vectors and a set with input and target vectors used for testing the performance. We use a cross validation technique to achieve reliable results. The pattern set of each product is split into five parts. In each iteration of the algorithm, we use four chunks for training and one for testing purposes. The overall performance is calculated by taking the average of all iterations. This technique is called *leave-some-out cross validation* [16].

4) Performance Benchmarking: For the daemon, we calculated the average time for one cycle in the algorithm over multiple hours of operation. When we repeated the measurement for the different algorithms, we measured the same time slots on different days. We define one cycle as querying the database for new data plus the time used for matching given there is a pattern detected.

 ISBN: 978-1-61208-189-2

Algorithm	Shortest duration	Longest duration	Average Duration
knn	3ms	806ms	9ms
SVM	3ms	1116ms	10ms
IQRC	3ms	1547ms	13ms

Table II
ITERATION TIMES AND THEIR DEVIATION DURING REAL-TIME DETECTION

IV. EVALUATION

A. Real-Time Performance

For interactive applications the critical limit for interaction response is two seconds [17]. Therefore, queries that are triggered by the real-time matcher have to be answered within this time interval. Real-time matching is implemented as a daemon. Although there is no critical time limit for this daemon, having as many matching cycles as possible allows discovering the patterns really close to their occurrences.

When the machine is idle, one cycle takes about three milliseconds. If the coffee machine is currently producing, it still takes less than a second. Table II shows the cycle times for the different algorithms. We can see that nearest neighbor matching performs best. Matching with Support Vector Machine takes about 30% percent longer. IQRC matching needs about twice the time compared to nearest neighbor. The reason is that we have to optimize the thresholds for all products individually in each cycle as they depend on the pattern length. Still, all algorithms have satisfying performance, as they are below the critical limit of two seconds.

B. Bulk Pattern Recognition

In addition, we analyzed the performance for bulk pattern recognition. Figure 2 shows the execution times of the MCSVM algorithm, implemented as a GNU R function, depending on the number of readings in the `device_readings` table for different numbers of used cores using nearest neighbor matching. The values represent the averages of ten measurements with a standard deviation of 11%. The execution times for support vector lie within the standard deviation of the execution times for knn. Therefore, we conclude that both algorithms perform equally well.

The comparison shows that the bulk matching scales linearly for more than 1000 values. This is not surprising, since we are iterating over device readings, which gives the algorithm a complexity of $\mathcal{O}(n)$.

The *rminer* package, which is used in the GNU R implementation does not parallelize its computation, therefore we parallelized the execution by distributing the available input values among the number of processors. Each of the n cores could then independently work on $\frac{1}{n}$th of the total values.

We also laid focus on parallel execution, utilizing all cores of our target system. As we can see, the usage of multiple CPUs decreases the execution time. However, this speedup is not linear as we might expect, because of the massive parallelization. According to Ahmdal's law the speedup is determined by the serial fraction of the algorithm [18]. In our case, this fraction is determined by the initialization

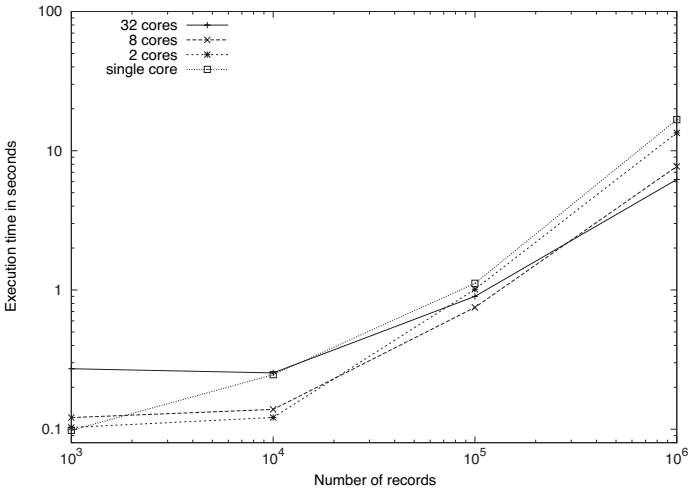

Figure 2. Comparison of execution times for bulk pattern recognition depending on the number of reading for different number of CPU cores

of the matcher and the merging of different parts of the `device_readings table`. We found out that the merging of one part of the list for a total of one million datasets took 10 to 20 ms. Although we tried to parallelize the merge, the jump from eight to 32 processes, increases the execution time for less than 200,000 value. The overhead in the merge is not outrun by the smaller number of device readings, each process has to analyze. However, 32 cores outperform eight cores for 200,000 readings. With an increasing number of readings, the gain from executing the computing expensive operations in parallel increases.

C. Precision Results

Fig 3 shows the hit rates for all features using the presented inter-quartile range coverage, k-nearest neighbor and Support Vector Machine algorithms. For the IQRC we were not able to perform the benchmark with the histogram feature, because all patterns of one product result in one histogram. They do not have any deviations or quartiles. The lower boundary for the matching performance with eight different products is 12.5% which would be the accuracy of chance. The PDU, which was used for measuring the consumption data, has an accuracy of $\pm 10\%$ for its measurements [14].

As we can see in Figure 3, the multidimensional features consumption, gradient and moving average perform equally. They are also the best performing features in total. The more dimensions are available, the more information can be used for classification, which lead to higher hit ratios.

One dimensional features perform significantly worse than multidimensional features.

The histogram feature could only be implemented for knn and SVM. Though it performs slightly better than the presented one dimensional feature, it is still noticeably worse than the multidimensional features. Reason is that the measured values hardly differ in size the histogram has a lot of different values with low frequencies.

 ISBN: 978-1-61208-189-2

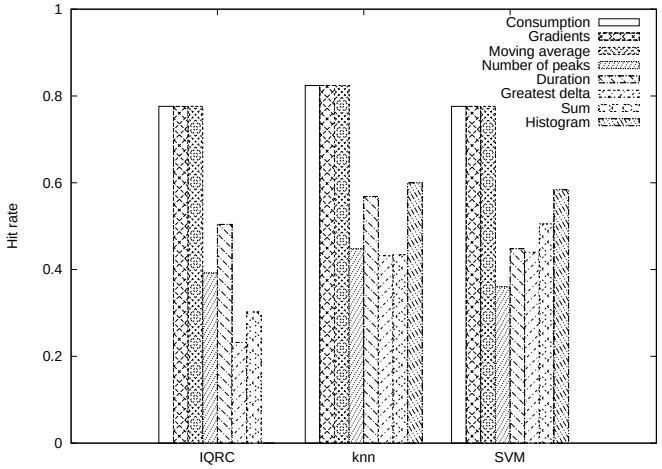

Figure 3. The comparison of three algorithms with different data features

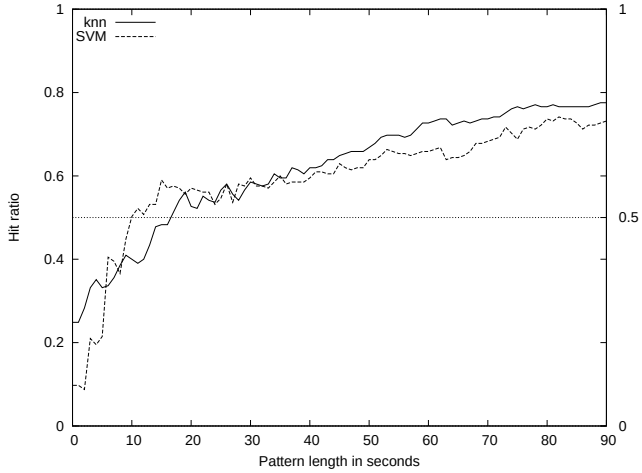

Figure 4. Hit rate depending on the length of the input vector

If we compare the algorithms with each other, we can see that the knn performs slightly better than the other two. It performs about 5 - 10 % better than the other in all features except for the greatest delta and sum feature. For multidimensional features the IQRC algorithm has the same hit rate as SVM. Nonetheless, IQRC outperforms SVM slightly considering the number of peaks and duration features. SVM on the other hand has the strongest results for the greatest delta and sum criteria. The implementation of multi-class SVM using *one-versus-all* is susceptible to misclassification if all machines calculate a negative result [19]. Due to the high deviation amongst patterns in our scenario, this case occurs more often. Therefore the overall accuracy of SVM is not as high as expected.

D. Short-Term Energy Consumption Prediction

If a pattern is detected, its subsequent values can be used for predicting the future consumption of a device. The earlier we recognize the product that is currently produced, the result gets more useful because the predicted remaining interval gets longer. Early matching has to be performed on incomplete consumption data and is therefore not as accurate as matching after the complete consumption. Fig 4 shows the accuracy of the knn and SVM algorithm depending on the length of the patterns. If we pass a pattern with length n we cut all training patterns down to that length and apply the algorithms.

We consider a hit rate of 0.5 to be sufficient in order to speak of successful pattern recognition. There are eight possible beverages, a hit rate bigger than 50% would be four times better than chance. As we can see, we break the 0.5 accuracy line at approximately 20 seconds. This means approximately one third of the pattern are sufficient for pattern recognition. If we transfer that finding to industrial manufacturing processes that take multiple hours, this forecasting range is valuable for utility companies, as it is sufficiently long for trading e.g. at the EEX spot market [20].

After twenty seconds of production we are able to identify the product. This means we can predict the succeeding ten to seventy seconds using the information from our trained patterns. The easiest way to predict values for the short term is to take the nearest neighbor and predict its course for the next values. We decided to use the knn approach for short term prediction also because its accuracy outruns the SVM's in the long run. An SVM approach could be implemented using the values of the closest support vector of the corresponding class as a prediction.

When we predict the subsequent consumption of a pattern after 20 seconds, we have an average deviation of 25%. After 40 seconds the deviation falls below 20%. This accuracy might not be convincing at first sight. Considering that the consumption values of the coffee machine even under load lie between 0.1 and 0.8 watt seconds, a predicted value that only differs by .01 watt seconds may lead to a deviation of 10%. Therefore we would have to predict three decimal places correctly to fall below that number. If we subtract measuring errors of the PDU [14], we end up with a deviation around 10%. For high performance industrial machines, the consumption is higher than for the coffee machine. Therefore the precision in industrial use cases is expected to be higher.

V. Related Work

Smart grids are considered as the continuation of the classical power grid in the information age [21]. In order to avoid different conflicting standards amongst its participant countries, the European Union has instantiated the Smart Meter Coordination Group (SMCG) [22]. OPENmeter is a project that is financed by the European Union and has proposed the AMI to be used for the smart grid [23]. On top of that manufacturers found a platform called Open Metering Systems that collaborates with SMCG and also assumes the AMI [22], [24]. Regardless of its changes, our experimental setup is comparable to this AMI.

Collected smart meter data can be used to to optimize consumer contracts [25]. More detailed monitoring of power data implies, that the energy consumption of current machines used in a data center highly depend on the computed task [26].

 ISBN: 978-1-61208-189-2

Predicting the energy consumption for medium and shorter terms has been done using artificial neural-networks [27], [28]. Related work has been focussing on comparing different algorithms for efficient pattern matching over event streams [29].

The Support Vector Machine is one of the most popular algorithms used in machine learning. It was introduced by Vapnik in 1992 and has proved to provide significantly better classification performance than other algorithms in most use cases [11], [8]. SVM can solve binary classification problems by finding an n-dimensional function that is able to distinct all data points of one class from another. A discussion on the various multi-class implementations of SVMs is done by Duan and Keerthi [12].

VI. Conclusion and Future Work

In our experiment we have shown, that it is possible to determine the type of coffee, that is produced by a coffee machine only based on a series of smart meter readings. We used those readings from a small dataset to train our different pattern recognition algorithms. The used in-memory database has shown that it is indeed able to process the collected amount of data, which was approximately 471 million tuples within the `device_readings` table and match it with the pre-calculated patterns in real time, meaning running a complete detection cycle below two seconds.

We have shown, that bulk pattern recognition can be scaled to multiple CPUs to enable an energy providing company with the possibility to cluster and aggregate the consumption data of multiple customers at a time. In the future unsupervised learning algorithms need to be applied to automatically create patterns from the energy consumption data. Thus, clustering customers in multiple groups helps companies to optimize their energy rate offers.

If a manufactured good can be determined by its energy footprint, those pattern recognition techniques might also be used to detect an early machine break down in the area of predictive maintenance. This can reduce the costs caused by unscheduled machine downtime within a productive environment.

Future work will include revalidating the results with different types of manufacturing machines, consuming a higher amount of energy and producing different energy usage footprints.

References

[1] T. N. Project, "Energy Consumption," *Intermediate Energy Infobook*, pp. 1–5, 2011.

[2] L. A. Butler, "In-Home Display Pilot," *US Department of Energy - Energy Efficiency and Renewable Energy*, pp. 1–9, Jul. 2011.

[3] S. Darby, "The Effectiveness of Feedback on Energy Consumption," *Environmental Change Institute, University of Oxford*, pp. 1–24, Apr. 2006.

[4] The OPENmeter Consortium, "Report on the identification and specification of functional, technical, economical and general requirements of advanced multi-metering infrastructure, including security requirements," *Deliverables*, June 2009.

[5] R. Weron, *Modeling and forecasting electricity loads and prices*, ser. a statistical approach. Wiley, Dec. 2006.

[6] H. Plattner, "A common database approach for OLTP and OLAP using an in-memory column database," *Proceedings of the 35th SIGMOD International Conference on Management of Data*, Jun. 2009.

[7] J. Krueger, M. Grund, C. Tinnefeld, and H. Plattner, "Optimizing write performance for read optimized databases," *Database Systems for Advanced Applications*, 2010.

[8] S. Marsland, *Machine Learning An Algorithmic Perspective*. Chapmann&Hall/CRC, 2009.

[9] C. M. Bishop, *Pattern Recognition And Machine Learning*. Springer, 2007.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*, ser. Prentice Hall series in artificial intelligence. Prentice Hall, 2002.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, pp. 273–297, 1995.

[12] K. Duan and S. S. Keerthi, "Which is the best multiclass svm method ? an empirical study," in *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, 2005, pp. 278–285.

[13] *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.

[14] E. E. Co., "User manual - network interface card for the liebert rack pdu family of power distribution products," Monitoring For Business-Critical Continuity, Tech. Rep., 2009.

[15] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011.

[16] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection." Morgan Kaufmann, 1995, pp. 1137–1143.

[17] Galitz and W. O., *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. Wiley & Sons, 2007.

[18] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Proc. AFIPS 1967 Spring Joint Computer Conf. 30 (April), Atlantic City, N.J.*, pp. 483–485, 1967.

[19] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines." *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.

[20] "Transparency in energy markets," European Energy Exchange AG, Tech. Rep., 2011.

[21] M.-P. Schapranow, R. Kühne, A. Zeier, and H. Plattner, "Enabling Real-Time Charging for Smart Grid Infrastructures using In-Memory Databases." IEEE, pp. 1040–1045.

[22] Arbeitsgemeinschaft Für Sparsame und Umweltfreundlichen Energieverbrauch E.V., "Smart Meter - Intelligente Zähler."

[23] "Requirements of AMI," OPENmeter, Tech. Rep., 2009.

[24] H. Baden and P. .Gabriel, "Open metering system specification," Open Metering System Group, Tech. Rep., 2011.

[25] R.-I. C. Chi-Cheng Chuang, Jimi Y. C. Wen, "Consumer Energy Management System: Contract Optimization using Forecasted Demand."

[26] M. M. G. P. Antonio Vetro', Luca Ardito, "Monitoring IT Power Consumption in a Research Center: Seven Facts."

[27] S. A. and Kalogirou, "Applications of artificial neural-networks for energy systems," *Applied Energy*, vol. 67, no. 1-2, pp. 17 – 35, 2000.

[28] P. A. Gonzalez and J. M. Zamarreto, "Prediction of hourly energy consumption in buildings based on a feedback artificial neural network," *Energy and Buildings*, vol. 37, pp. 595 – 601, 2005.

[29] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman, "Efficient pattern matching over event streams," in *Proceedings of the 2008 SIGMOD International Conference on Management of Data*, 2008, pp. 147–160.

 ISBN: 978-1-61208-189-2