Efficient Sparse Matrix Processing for Nonintrusive Load Monitoring (NILM)

Stephen Makonin, Ivan V. Bajić, Fred Popowich
Simon Fraser University, Burnaby, BC, Canada
Email: smakonin@sfu.ca, ibajic@sfu.ca, popowich@sfu.ca

***Abstract*—Nonintrusive load monitoring (NILM) is a process of discerning what appliances are running within a house from processing the power or current signal of a smart meter. Since appliance states are not observed directly, hidden Markov models (HMM) are a natural choice for modelling NILM appliances. However, because the number of HMM states grows rapidly with the number of appliances and their internal states, existing methods have relied on either simplifying the model (e.g. factorial HMM) or reducing the number of appliance states (e.g. considering only on/off states), all of which reduces the accuracy of NILM. In this paper we present a new NILM algorithm that can handle multiple internal appliance states while still keeping complexity in check by utilizing the sparsity of HMM emission and transition matrices. The results show overall accuracy results of 95% for both classification and estimation using the AMPds dataset.**

***Index Terms*—Load disaggregation, sparse matrix, super-state HMM, Viterbi algorithm**

I. Introduction

Nonintrusive load monitoring (NILM) is the process of discerning what appliances are running in a house from analyzing the power or current signal of the whole-house smart meter. NILM algorithms could be used to inform the occupants about what appliances are running within the home, and how much power these appliances consume, without the need of purchasing additional power monitoring sensors. With this information the occupants can make informed decisions about conserving power, whether motivated economically or ecologically (or both).

Since individual appliance internal states are not directly observed in the total power or current reading, hidden Markov models (HMM) have been a natural choice for modelling in NILM [1], [2]. Although there are a number of algorithms for HMM decoding, such as the Viterbi algorithm [3], the main practical obstacle to NILM is the complexity involved in state processing and enumeration (e.g. total states $= K^M$, where each of M appliances has K states). This exponential size of the state space has been a practical limitation to the wider application of NILM.

To curtail exponential complexity, Kolter [1] and Parsons [2], among others, have use factorial HMMs [4]. Each appliance is a separate multi-state Markov chain which closely reflects the different operational states of that appliance. For purposes of discussion, we define binary appliances (simple ON/OFF) as a *subset* of multi-state appliances.

On the other hand, Zeifman [5] proposed the *Viterbi algorithm with sparse transitions* (VAST), a modified version of the Viterbi algorithm, with multiple transition matrices (each a triple of appliances), which only disaggregated binary appliances (ON or OFF). Such an approach required him to approximate a multi-state appliance (e.g., the dishwasher) to be a binary appliance [6]. The majority of modern appliances are not binary, which severely limited what appliances VAST could disaggregate. It is worth noting that the 3-appliance transition matrices can still have zero-probability elements and that the sparsity in the emission matrix was not dealt with. One of the biggest disadvantages, when using multiple matrices, is loss of information on appliance dependence. For example, consider the condition that when a heat pump turns ON, the HVAC fan (on a separate breaker) increases its rotation speed, and then does the reverse when the heat pump turns OFF (as we have observed in our dataset [7]).

Our main contribution is an HMM-based disaggregation algorithm that deals with matrix sparsity in a much simpler way than Zeifman while preserving load dependency information that would have been lost by using multiple transition matrices. Our present work introduces the use of one HMM (with one transition and emissions matrix) where the super-state of this HMM could be considered the *house state* (Section II). Further, we introduce the *sparse Viterbi algorithm* for NILM, which offers an alternative, more accurate, solution exploiting matrix sparsity as compared to the earlier work of Zeifman (Section III). Our approach can handle multi-state appliances while keeping complexity in check, by making use of the observed sparsity in both emission and transition matrices of the HMM. This leads to higher accuracy compared to existing NILM algorithms. We show that the accuracy for our NILM algorithm scores well (Section IV).

II. HMM for NILM

We model a house with M appliances as an HMM $\lambda = \{\mathbf{P}_0, \mathbf{A}, \mathbf{B}\}$ having a row-vector of initial prior probabilities $\mathbf{P}_0$ of length K, a $K \times K$ transition matrix $\mathbf{A}$, and a $K \times N$ emission matrix $\mathbf{B}$, where K is the number of whole-house states (or super-states), and N is the number of possible observations. If $t-1$ and t represent the previous and the current time instants, the entries of $\mathbf{A}$ and $\mathbf{B}$ are defined as $\mathbf{A}[i,j] = p(S_t = j | S_{t-1} = i)$, $\mathbf{B}[j,n] = p(y_t = n | S_t = j)$; where S_t is the super-state and y_t is the observation, both at time t.

The super-state is composed of the states of all M appliances: $S_t = (X_t^{(1)}, X_t^{(2)}, ..., X_t^{(M)})$, where random variable

$X_t^{(m)}$ is the internal state of the m-th appliance at time t. For example, a dishwasher may have 4 internal states that consist of {OFF, WASH, RINSE, DRY}. The total number of super-states is $K = \prod_{m=1}^{M} K^{(m)}$ where $K^{(m)}$ is the number of internal states of the m-th appliance.

As the state of an appliance changes so too does its power or current draw (instantaneous readings). Let $y(\cdot)$ be the current draw of the corresponding internal appliance state, so that $y\left(x_t^{(m)}\right)$ is the current draw of the m-th appliance in state $x_t^{(m)}$. For notational convenience, we assume that the current values are non-negative integers, i.e., $y\left(x_t^{(m)}\right) \in \{0, 1, ..., N\}$; in practise, these would not necessarily be integers, but would still be constrained to a discrete set of possible readings of the current meter. The observed measurement at time t from the smart meter is the sum of the current draws of individual appliances: $y_t = \sum_{m=1}^{M} y\left(x_t^{(m)}\right)$.

A. State Quantization

Model parameters, such as appliance state probabilities $p(X_t^{(m)} = x_t^{(m)})$ as well as conditional probabilities in $\mathbf{A}$ and $\mathbf{B}$, can be obtained from existing NILM datasets (e.g. AMPds [7]). In general, even though the assumed full set of possible current draws is $\{0, 1, ..., N\}$, all appliances have fewer than N states. To model this, for the m-th appliance, we quantize the set $\{0, 1, ..., N\}$ into $K^{(m)}$ bins such that the first bin contains 0 (and corresponds to the OFF state), while other bins are centred around the peaks of the empirical probability mass function (PMF) of the current draw $p_{Y_m}(n) = p\left(y\left(X_t^{(m)}\right) = n\right)$, $n \in \{0, 1, ..., N\}$, obtained from the dataset. A peak in the PMF is identified when the slope on the left, $p_{Y_m}(n) - p_{Y_m}(n-1)$, is positive, the slope on the right, $p_{Y_m}(n+1) - p_{Y_m}(n)$, is negative, and $p_{Y_m}(n) > \epsilon$, where $\epsilon = 0.00021$ used to ensure that small peaks (noise) are not quantized as states.

A simple example of such quantization is given in Table I, where $N = 6$, but only two states are identified after quantization, hence $K^{(m)} = 2$. These states are indexed by $k^{(m)} \in \{0, 1\}$ and are centred around the values $n = 0$ and $n = 3$. The quantized states are denoted $\widehat{X}_t^{(m)}$. The current draws of the quantized states are stored as a $K^{(m)}$-dimensional vector, denoted $\mathbf{y}_{peak}^{(m)}$, whose elements are the locations of the peaks in the original PMF. For the example in Table I, $\mathbf{y}_{peak}^{(m)}[0] = 0$ and $\mathbf{y}_{peak}^{(m)}[1] = 3$. The probability of a given quantized state is simply the sum of probability masses in the corresponding bin, hence for the above example, $p\left(y\left(\widehat{X}_t^{(m)}\right) = 0\right) = p(k^{(m)} = 0) = 0.45$ and $p\left(y\left(\widehat{X}_t^{(m)}\right) = 3\right) = p(k^{(m)} = 1) = 0.55$. Since $K^{(m)} \leq N$, it can be seen that state quantization will increase the sparsity of $\mathbf{A}$ and $\mathbf{B}$.

The super-state corresponding to quantized internal states is $\widehat{S}_t = (\widehat{X}_t^{(1)}, \widehat{X}_t^{(2)}, ..., \widehat{X}_t^{(M)})$. The quantized super-states can be indexed linearly in terms of the indices of the quantized internal states $k^{(1)}, k^{(2)}, ..., k^{(M)}$ as follows:

TABLE I
AN EXAMPLE OF PMF AND STATE QUANTIZATION

n	0	1	2	3	4	5
count(n)	900	80	100	620	200	100
$p_{Y_m}(n)$	0.45	0.04	0.05	0.31	0.10	0.05
$\mathbf{y}_{peak}^{(m)}$	0	3				
$k^{(m)}$	0	1				
$p(k^{(m)})$	0.45	0.55				

$$k = k^{(M)} + \sum_{m=1}^{M-1} \left(k^{(m)} \cdot \prod_{i=m+1}^{M} K^{(i)} \right). \tag{1}$$

Based on (1), one can also extract individual appliance state indices $k^{(m)}$ from k by iteratively extracting the remainder of division of k by partial products $\prod_{i=1}^{m} K^{(i)}$, starting with $k^{(M)}$.

B. Exponentially Large but Sparse Matrices

As the number of appliances to disaggregate increases, the number of super-states K grows exponentially, and so too do the dimensions of $\mathbf{A}$ and $\mathbf{B}$. However, in the case of NILM these matrices are very sparse. In fact, so sparse that using a HMM with full super-state space is a practical option. The theoretical sparsity of $\mathbf{B}$ is clear from its definition. Since for each specific super-state $s_t = (x_t^{(1)}, x_t^{(2)}, ..., x_t^{(M)})$ there is exactly one output $y_t = y\left(x_t^{(1)}\right) + y\left(x_t^{(2)}\right) + \cdots + y\left(x_t^{(M)}\right)$, each row of $\mathbf{B}$ should contain exactly one non-zero element (and that element is equal to 1). However, with quantization, this might not hold exactly, as the current draw may fluctuate slightly within a quantized state even if the super-state does not change. Therefore, we may observe several non-zero values in a given row of $\mathbf{B}$, which of course should still sum up to 1. So the best-case sparsity of $\mathbf{B}$, defined as the fraction of zero entries, can be calculated as $1 - \frac{K}{K \cdot N} = 1 - \frac{1}{N}$.

The sparsity in $\mathbf{A}$ refers to the fact that there are relatively few possible transitions from any given super-state. While this is not as obvious as in the case of $\mathbf{B}$, it can be appreciated by realizing that multi-state appliances usually operate in *cycles* that determine the sequence of their states. For example, a possible state sequence for a dishwasher could be OFF $\rightarrow$ WASH $\rightarrow$ RINSE $\rightarrow$ DRY $\rightarrow$ OFF. Meanwhile, DRY $\rightarrow$ WASH $\rightarrow$ OFF would not make much sense.

To give a real-world example, we examined one year's worth of appliance data in the AMPds dataset [7] and found that $\mathbf{A}$ was 43.2% sparse while $\mathbf{B}$ was 97.3% sparse, when using two appliances, clothes dryer (CDE) and kitchen fridge (FGE), each having 3 quantized internal states ($3^2 = 9$ super-states). The house had a 200A service, so $N = 200$. In this scenario, the best-case sparsity for $\mathbf{B}$ would be $1 - \frac{1}{200}$ or 99.5%.

Sparsity of $\mathbf{A}$ and $\mathbf{B}$ can be used to simplify computations involved in Viterbi-based state decoding, as will be described in the next section. In addition, storage requirements can be significantly reduced. To this end, we employ the *Harwell-Boeing sparse matrix format* [8] (Algorithm 1) to store $\mathbf{A}$

and $\mathbf{B}$ in compressed form. For 11 appliances with 34,560 super-states, uncompressed $\mathbf{A}$ requires about 9.6 GB, while compressed $\mathbf{A}$ requires only 93.8 kB. We also define a function to returning a list of tuples of all non-zero probability elements (see Algorithm 2) which will be used for optimizing the Viterbi algorithm in Section III. It is worth noting that we have found relatively identical results occurred when examining the sparsity of the REDD dataset [9].

Algorithm 1 COMPRESS($\mathbf{M}$)

```
1: $val, row\_idx \leftarrow []$
2: $col\_ptr \leftarrow [1]$
3: for $col = 1 \rightarrow \mathbf{M}[0].\mathbf{length}()$ do
4:     for $row = 1 \rightarrow \mathbf{M}.\mathbf{length}()$ do
5:         if $\mathbf{M}[row, col] \neq 0.0$ then
6:             $val.\mathbf{append}(\mathbf{M}[row, col])$
7:             $row\_idx.\mathbf{append}(row)$
8:         end if
9:     end for
10:    $col\_ptr.\mathbf{append}(row\_idx.\mathbf{length}() + 1)$
11: end for
12: return $(val, row\_idx, col\_ptr)$
```

Algorithm 2 COLUMN-VECTOR($\mathbf{M}, col$)

```
1: $v \leftarrow []$
2: for $i = \mathbf{M}.col\_ptr[col] \rightarrow \mathbf{M}.col\_ptr[col + 1] - 1$ do
3:     $v.\mathbf{append}((\mathbf{M}.row\_idx[i], \mathbf{M}.val[i]))$
4: end for
5: return $v$
```

Algorithm 3 SPARSE-VITERBI($K, \mathbf{P}_0, \mathbf{A}, \mathbf{B}, y_{t-1}, y_t$)

```
1: $\mathbf{P}_{t-1}[k], \mathbf{P}_t[k] \leftarrow 0.0, k = 1, 2, ..., K$
2: for $(j, p_b) \in$ COLUMN-VECTOR$(\mathbf{B}, y_{t-1})$ do
3:     $\mathbf{P}_{t-1}[j] \leftarrow \mathbf{P}_0[j] \cdot p_b$
4: end for
5: for $(j, p_b) \in$ COLUMN-VECTOR$(\mathbf{B}, y_t)$ do
6:     $(i, p_a)[] \leftarrow$ COLUMN-VECTOR$(\mathbf{A}, j)$
7:     $\mathbf{P}_t[j] \leftarrow \max_{(i,p_a)}(\mathbf{P}_{t-1}[i] \cdot p_a \cdot p_b)$
8: end for
9: return $\arg\max(\mathbf{P}_t)$
```

III. SPARSE VITERBI ALGORITHM

The standard Viterbi algorithm is well suited for largely populated matrices. However, when used with sparse matrices, there is an extensive amount of naive probability calculations involving zero-probability terms. Harwell-Boeing matrix compression not only reduces the amount of storage, but it also allows us to avoid calculating zero-probability terms. Taking advantage of matrix sparsity to avoid unnecessary calculations forms the basis of our algorithm called *sparse Viterbi algorithm* (see Algorithm 3), which is based on a greedy version of the Viterbi algorithm [10, pp. 352].

Let y_{t-1} and y_t be the total current measurements at times $t-1$ and t. The goal is to infer the quantized super-state $\widehat{s}_t$ (or, equivalently, its index k_t), from which we will determine the quantized internal states. This will be achieved by decoding the internal states' index from the super-state index using (1). These posterior probabilities are stored in vector $\mathbf{P}_{t-1}$ as part of *initialization* (Algorithm 3, lines 1–4), directly from [10]

$$\mathbf{P}_{t-1}[j] = \mathbf{P}_0[j] \cdot \mathbf{B}[j, y_{t-1}], j = 1, 2, ..., K. \quad (2)$$

The computation is reduced by only considering non-zero elements of $\mathbf{B}[j, y_{t-1}]$ in (2), to be clarified below. We now calculate the posterior probabilities for the current time period (the *recursion* step, Algorithm 3, lines 5–8)

$$\mathbf{P}_t[j] = \max_{i=1}^{K}(\mathbf{P}_{t-1}[i] \cdot \mathbf{A}[i, j] \cdot \mathbf{B}[j, y_t]), j = 1, 2, ..., K. \quad (3)$$

We *terminate* (Algorithm 3, line 9) to find the most likely current super-state index $k_t = \arg\max(\mathbf{P}_t)$. This algorithm is called each time we need to disaggregate a reading, using a *sliding window* of observations. For example, we disaggregate $t = \{1, 2\}$, then $t = \{2, 3\}$, $t = \{3, 4\}$, and so on. Disaggregation only begins when the first 2 observations are received from the meter. For our purposes, we are not interested in the prediction of the super-state from $t - 1$ (the *backtracking* step). Once the k_t is determined feedback is sent to the occupant – this makes it final, no turning back time. Again, zero-probability terms are avoided in the calculation. The zero-probability terms are effectively ignored due to the Harwell-Boeing matrix compression method, which only stores non-zero elements of the corresponding matrices. A call to the function COLUMN-VECTOR() (Algorithm 3, lines 2, 5, and 6) only returns non-zero elements, which removes zero probability terms from the calculations.

IV. EXPERIMENTS

To show the benefits of taking advantage of sparsity, we used the AMPds dataset [7], which contains 1 year's worth of appliance data (524,544 readings at 1 minute intervals). Our tests showed that with 2 appliances, each having 3 states (9 super-states), for every 1 section of code execution (Algorithm 3, line 7) of sparse Viterbi, the conventional Viterbi algorithm would, on average, execute the same section 72.7 times. Figure 1 shows the time taken to disaggregate all 524,544 readings from 2 to 11 appliances (AMPds data) by conventional and sparse Viterbi algorithms. The savings are considerable, especially for more than 4 appliances (135 super-states). For example, for 2 appliances, the execution time was reduced from 16.8 seconds to 11.4 seconds (32% faster) and for 11 appliances (34,560 super-states) the execution time was 94 minutes. Tests were done on a MacBook Pro machine with a 2.6 GHz Intel Core i5 processor.

We test our NILM algorithm with data from the AMPds [7] dataset (524,544 readings or events, our algorithm runs on-line so each reading is an event). We use the instantaneous current measurement (I) at dA precision (1,572,864 super-states) because it fluctuates less than power [7]. However, our algorithm is agnostic of measurement type and we could have used real or apparent power. To mitigate bias we used

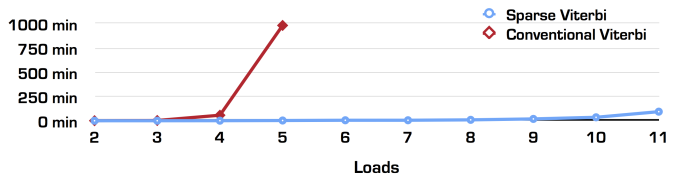

Fig. 1. Runtime (in minutes) comparison of conventional and sparse Viterbi algorithms when disaggregating 524,544 readings.

10-fold cross-validation. We ran 2 tests: denoised and noised. For denoised, the whole-house current draw measurement was denoised so that it equalled the summation of the current draw from the 11 appliances chosen for disaggregation. For noised, the whole-house current draw measurement was not modified. The results are listed in Table II where results are reported as "denoised% / noised%". For example, "94.5 / 91.4" would read 94.5% on the denoised test and 91.4% on the noised test. The ON column is the load's % time ON. The *Events* column shows the number of state changes. Results show a very high degree of accuracy for most appliances in both tests. The Basement, Washer, Dishwasher, and Fridge loads scored low due to having states that operate with the same current draw. For our noised test, 40.9% of the total aggregate data was noise (i.e. from other loads, not the 11 loads in Table II).

TABLE II
ACCURACY TESTING RESULTS (DENOISED/NOISED)

Load	FS f-score	Estimation	ON	Events
Basement	53.6 / 40.2	99.0 / 69.3	23.0	6404
Dryer	99.7 / 99.6	90.8 / 92.2	100.0	1826
Washer	21.5 / 3.2	60.2 / 57.4	2.6	9961
Dishwasher	60.8 / 14.3	86.3 / 85.9	2.7	4394
Fridge	76.2 / 49.2	99.4 / 99.2	45.1	43500
HVAC/Fan	97.9 / 96.2	99.7 / 98.8	100.0	2531
Garage	99.9 / 99.9	99.8 / 99.9	100.0	228
Heat Pump	98.1 / 97.0	99.1 / 88.8	100.0	8291
Home Office	95.8 / 96.0	88.1 / 82.1	100.0	11044
Ent/TV	88.9 / 85.0	98.6 / 91.5	100.0	13314
Wall Oven	99.8 / 99.6	84.0 / 85.0	100.0	396
Overall	94.5 / 91.4	97.4 / 96.8	100.0	101889

A. Accuracy Measuring Methods

Table II has 2 accuracy measuring methods: Estimation and FS f-score. The estimation measure is based on Kolter and Johnson [9]. FS f-score or *finite-state f-score* [11] is our classification measure (inspired by [12], itp and atp). We wanted a way to measure non-binary classifications. As f-score is relevant for binary appliances (OFF/ON), it is not for multi-state appliances. We split tp (true-positives) into two: itp (inaccurate true-positives) and atp (accurate true-positives). The use of itp allows us to partially penalize a misclassification that is not binary in nature. To calculate itp we use the equation

$$itp = \frac{|\hat{x}_t^{(m)} - x_t^{(m)}|}{K^{(m)}}, \tag{4}$$

where $\hat{x}_t^{(m)}$ is the estimated state from appliance m at time t, $x_t^{(m)}$is the ground truth state, and $K^{(m)}$ is the number of states for appliance m. To determine the atp we use the equation $atp = 1 - itp$.

V. CONCLUSIONS

We presented a sparse Viterbi algorithm that can disaggregate with a high degree of overall accuracy (95% for both classification and estimation) using the AMPds dataset. The practicality of using a super-state HMM is enabled by exploiting sparsity in transition and emissions matrices and preserving the conditional probability between separate appliances.

ACKNOWLEDGMENTS

Research partly supported by grants from the National Sciences and Engineering Research Council (NSERC) of Canada, and the Graphics, Animation, and New Media Network of Centres of Excellence (GRAND NCE) of Canada.

REFERENCES

[1] J. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," *Journal of Machine Learning Research - Proceedings Track*, vol. 22, pp. 1472–1482, 2012.

[2] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012.

[3] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[4] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Machine Learning*, vol. 29, no. 2-3, pp. 245–273, 1997.

[5] M. Zeifman and K. Roth, "Viterbi algorithm with sparse transitions (VAST) for nonintrusive load monitoring," in *Computational Intelligence Applications In Smart Grid (CIASG), 2011 IEEE Symposium on*, 2011.

[6] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 1, pp. 23 –31, 2012.

[7] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic, "AMPds: A Public Dataset for Load Disaggregation and Eco-Feedback Research," in *Electrical Power and Energy Conference (EPEC), 2013 IEEE*, 2013.

[8] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Sparse matrix test problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 15, no. 1, pp. 1–14, 1989.

[9] J. Kolter and M. Johnson, "REDD: A Public Data Set for Energy Disaggregation Research," in *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, 2011.

[10] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman & Hall/CRC, 2009.

[11] S. Makonin and F. Popowich, "Nonintrusive Load Monitoring (NILM) Accuracy: How To and Best Practises for Reporting," *Energy Efficiency*, [in submission].

[12] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *11th International Conference on Data Mining*, 2010, pp. 747–758.