Label Correction and Event Detection for Electricity Disaggregation

Mark Valovage
Computer Science and Engineering
University of Minnesota
Minneapolis, Minnesota
valovage@cs.umn.edu

Maria Gini
Computer Science and Engineering
University of Minnesota
Minneapolis, Minnesota
gini@umn.edu

ABSTRACT

Electricity disaggregation focuses on identifying individual appliances from one or more aggregate signals. By reporting detailed appliance usage to consumers, disaggregation has the potential to significantly reduce electrical waste in residential and commercial sectors. However, application of existing methods is limited by two critical shortcomings. First, supervised learning methods implicitly assume error-free labels in training data, an unrealistic expectation for imperfectly-labeled consumer data. Second, supervised and unsupervised learning methods require parameters to be tuned to individual appliances and/or datasets, limiting widespread application. To address these limitations, this paper introduces the implementation of Bayesian changepoint detection (BCD) with necessary adaptations to electricity disaggregation. We introduce an algorithm to effectively apply BCD to automatically correct labels. We then apply BCD to event detection to identify transitions between appliances' on and off states. Performance is evaluated using 3 publicly available datasets containing over 250 appliances across 11 houses. Results show both BCD applications are competitive and in some cases outperform existing state-of-the-art methods without the need for parameter tuning, advancing disaggregation towards widespread, real-world deployment.

CCS Concepts

•Theory of computation → Theory and algorithms for application domains; •Social and professional topics → Sustainability;

Keywords

Electricity Disaggregation, Event Detection, Change Detection, Label Correction, Supervised & Unsupervised Learning

1. INTRODUCTION

Energy waste costs $130 billion and produces 1.1 gigatons of pollution each year in the United States alone [11]. In the residential sector, up to 20% of electrical waste could be eliminated through adjustments that have no impact on users' lifestyles [6, 13, 23]. However, sources of waste are difficult to identify since waste varies widely depending on appliance installation, maintenance, and usage [5, 9, 25, 28].

Software agents are playing an increasing role in creating more efficient use of electrical power, including smart grid management of renewable resources, improved load balancing, tariff pricing, and demand response [19]. However, limited attention has been devoted to designing agents in support of customer-centric waste reduction. We envision an energy management agent which performs real time energy monitoring, detects which appliances are operating, and provides usage feedback and recommendations to the user. If given authority, this agent can even turn off unused devices or shift operation of specific appliances to a later time.

The first step for such an agent is to obtain appliance-specific data. Although commercially available smart meter kits have the capacity to measure individual appliance signals, they require a meter for every appliance. This is impractical since meters cannot fit all plug configurations, require significant installation time, and the expense of installing a smart meter on every household appliance outweighs the potential waste reduced [6]. In contrast, electricity disaggregation, also called non-intrusive load monitoring (NILM) [12] or single point sensing [22], measures continuous aggregate electrical signals with a single meter.

Despite recent advances, existing disaggregation methods are hampered by unrealistic data assumptions. First, supervised learning methods assume all training samples are captured in isolation and are correctly labeled, assumptions that break down with consumer-labeled data. Second, performance of many algorithms depends on parameters optimized for a single appliance or dataset, and parameters are often tuned and evaluated on the same data [14]. Manually cleaning samples and tuning parameters to individual appliances cannot scale to wide-spread implementation, limiting real-world deployment of existing classification methods.

Contributions: This paper makes two contributions based on Bayesian changepoint detection (BCD). First, we present a method to correct user labeling errors and provide cleaner training samples for existing supervised classification methods. Second, we present an event detection method for unsupervised learning to identify transitions between appliances' on and off states. Both of these contributions are novel since they do not require parameter tuning or pre-processing to reduce noise. Results show performance comparable to manually tuned, state-of-the-art label correction and event detection methods, and we show these approaches can be performed in real time with inexpensive hardware, enabling real-world deployment of existing classification methods.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

2. RELATED WORK

Numerous supervised and unsupervised learning approaches have been applied to electricity disaggregation. Supervised algorithms include Additive Factorial Hidden Markov Models [17], Viterbi Algorithm with Sparse Transitions [29], Sparse Coding [16] coupled with Powerlets [8], and others [30].

In contrast, unsupervised methods disaggregate appliances without isolated training samples, which requires at least two distinct steps. First, the data stream must be parsed by detecting electrical events through methods such as Genetic K-means [10], Modified Greatest Likelihood Ratio (mGLR) [4], or Dirichlet Process Gaussian Mixture Models (DPGMM) [24]. After events are identified, they can be separated into different appliances using Temporal Motif Mining [24], Additive Factorial Approximate Map (AFAMAP) [17], Factorial or Iterative Hidden Markov Models [15, 21], and others [30].

However, all of the aforementioned supervised learning methods are provided training data cleansed by manual preprocessing, ignoring the impact of contaminated samples or mislabeled data on classification results. In addition, during unsupervised learning, proper appliance reconstruction depends on accurate event detection, which often relies on pre-tuned parameters [3] or domain-level heuristics [15].

This paper addresses these limitations and is organized as follows: Section 3 describes BCD [2] and our modifications. Section 4 details our label correction approach and results. Section 5 applies BCD to event detection and presents event detection results, and Section 6 concludes with future work.

3. CHANGEPOINT DETECTION

3.1 Original Algorithm

To address existing real-world disaggregation limitations, we use online Bayesian change detection (BCD) [2] which has been previously applied to identify changepoints in stock market indexes, coal mining accidents, well drilling, traffic congestion, and satellite fault data [2, 7, 26]. We adopt the following notations from BCD's original authors [2]: $X = x_1, x_2, ..., x_T$ is the input data stream, where x_t is the observed data value at time t. $x_{a:b}$ specifies the contiguous sequence of observations $x_a, x_{a+1}, ..., x_b$ between times a and b. A 'run sequence' r_t at time t of length L denotes the last L observations (i.e. $r_t = x_{(t-L+1):t}$), and $P(r_t|x_{1:t})$ denotes BCD's calculated probability that the run sequence r_t is the 'active' run sequence at time t given the previous observations $x_{1:t}$.

We restate the BCD algorithm [2] in Algorithm 1 for the reader's convenience; its general flow is as follows. BCD begins by assuming X can be segmented into non-overlapping partitions ρ_1, ρ_2, ..., ρ_I and identifies changepoints as time indexes that separate these partitions. After observing a new data point, x_t, BCD finds the most likely active run sequence $R_t = \arg\max_{r_t} P(r_t|x_{1:t})$ out of all possible run sequences r_t at time t, thereby identifying the current partition ρ_i. A changepoint is identified at time t *iff* $|R_t| = 0$ (i.e. if a new partition has been identified at x_t).

To compute the probability distribution $P(r_t|x_{1:t})$ for all r_t at time t, BCD uses the following recursive formulation. After observing data points $x_{1:t-1}$, BCD has previously calculated the posterior probability distribution $P(r_{t-1}|x_{1:t-1})$ over all contiguous run sequences r_{t-1} containing the point x_{t-1}. Upon observing the new data point x_t (line 4), BCD uses Bayesian inference with the sample mean $\mu_{r_{t-1}}$ and sample variance $\sigma^2_{r_{t-1}}$ to compute the predictive probability, $\pi_t^{(r)}$, that x_t was drawn from r_{t-1} for all r_{t-1} (line 5). Next, BCD calculates each run sequence's growth probability, $P(r_t = r_{t-1}|x_{1:t})$, as a product of the predictive probability $\pi_t^{(r)}$ and the posterior probability $P(r_{t-1}|x_{1:t-1})$ (line 6). The probability for a new run sequence of length $r_t = 0$ is also included using a predetermined hazard function, $H(r_{t-1})$ (line 7). Finally, BCD normalizes the probabilities for all run sequences r_t to sum to 1 (line 8) and saves the distribution as $P(r_t|x_{1:t})$.

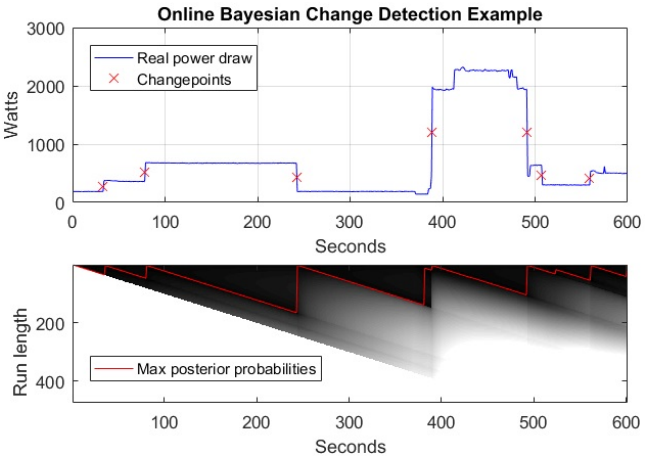

Figure 1: Example application of BCD to real power data. As the algorithm observes real power values (top), it computes run length probabilities at each time-step (bottom), represented as grayscale values. Darker pixels represent higher probabilities, and the maximum probability at each time-step is shown in red. A changepoint is identified when the sequence with the maximum probability changes between time-steps, shown as red x's (top).

3.2 Settings, Assumptions, and Modifications

Multiple electrical features have been explored for disaggregation including reactive power, voltage and current waveforms, and many others [30]. We limit this paper's scope to real power since it is the most common feature measured by existing smart meters and it enables exploitation of properties specific to real power consumption. An illustration of BCD applied to a real power data stream X is displayed in Figure 1. We leave the exploration of applying BCD to additional features to future work.

In this setting, each ρ_i represents a steady state power produced by a set of appliances, μ_i is the steady state power value, and σ_i^2 models the noise in that state. Under the assumption that each partition ρ_i is Gaussian centered around an unknown mean μ_i with unknown variance σ_i^2, we adopt the authors' recommended domain-agnostic values from their code for default mean $\mu_0 = 0$, default variance $\sigma_0^2 = 1$, and hazard function (which specifies the unweighted probability of a changepoint occurring) to $H(\tau) = \frac{1}{\lambda}$ with $\lambda = 200$ [1].

The original online BCD algorithm identifies a changepoint when the active run sequence at time t is the newly generated run sequence of length 0 (when $|R_t| = 0$) [2]. However, such an implementation needs to retroactively update previous changepoints identified, which is undesirable

Algorithm 1: Online Bayesian Changepoint Detection

Input: Data $X = x_1, x_2, ..., x_T$, hazard function $H(\tau)$.
Output: CP: Time indexes of identified changepoints.

1 $CP = \emptyset$
2 $t = 1$
3 **while** $t \leq T$ **do**
4 Observe new data point x_t
5 Compute predictive probabilities $\pi_t^{(r)} = P(x_t|r_{t-1})\forall r_{t-1}$
6 Compute growth probabilities $P(r_t = r_{t-1}|x_{1:t}) = P(r_{t-1}|x_{1:t-1})\pi_t^{(r)}(1 - H(r_{t-1}))$
7 Insert unweighted probability of a changepoint $P(r_t = 0) = H(r_{t-1})$
8 Normalize distribution $P(r_t)$ to sum to 1
9 Find maximum probability $R_t = \arg\max_{r_t} P(r_t|x_{1:t})$
10 **if** $R_t \neq R_t - 1$ **then**
11 *If a changepoint is found, save the time index*
12 Insert t into CP
13 $t = t + 1$
14 Return CP

Algorithm 2: Offline Bayesian Changepoint Detection

Input: Data $X = x_1, x_2, ..., x_T$, hazard function $H(\tau)$.
Output: CP: Time indexes of identified changepoints.

1 $CP = \emptyset$
2 $t = 1$
3 **while** $t \leq T$ **do**
4 Observe new data point x_t
5 Compute predictive probabilities $\pi_t^{(r)} = P(x_t|r_{t-1})\forall r_{t-1}$
6 Compute growth probabilities $P(r_t = r_{t-1}|x_{1:t}) = P(r_{t-1}|x_{1:t-1})\pi_t^{(r)}(1 - H(r_{t-1}))$
7 Insert unweighted probability of a changepoint $P(r_t = 0) = H(r_{t-1})$
8 Normalize distribution $P(r_t)$ to sum to 1
9 $t = t + 1$
10 *(Note that now $t = T$)*
11 **while** $t > 0$ **do**
12 Find maximum probability $R_t = \arg\max_{r_t} P(r_t|x_{1:t})$
13 $t = t - |R_t|$ (*Mark changepoint at the origin of R_t*)
14 Insert t into CP
15 Return CP

in an online implementation. To mitigate this, we modified the criterion of changepoint identification to $R_t \neq R_{t-1}$ (line 10), i.e. when the active run sequence changes between time-steps, which we found works better in this application.

An additional drawback of online BCD is that a time delay can exist between when a changepoint occurs and when online BCD identifies it. For example, suppose X contains two adjacent partitions, ρ_1 and ρ_2. If the mean difference $|\mu_1 - \mu_2|$ is small relative to their variances σ_1^2 and σ_2^2, it will take multiple observations from ρ_2 before BCD identifies a changepoint. In other words, if a changepoint occurs at x_t, BCD may not identify the changepoint until observing the data point x_{t+d} for some time delay $d > 0$. This time delay creates an offset of size d for low power changes, pushing the identified changepoint to the right of the actual changepoint.

As such, we introduce an offline formulation to the original online BCD algorithm, detailed in Algorithm 2. This calculates all probability distributions $P(r_t|x_{1:t})$ for all $t \in T$ (lines 3-9) before identifying any changepoints (lines 10-14), eliminating any time delay. Algorithm 2 can still run in near-online fashion, as described in the next section.

3.3 Computational Complexity

The time and space complexity of online BCD is $\Theta(T^2)$ (where $T = |X|$ is the number of data points) since at each time t, there are t different run sequence probabilities to update [2], making it impractical for long time horizons T.

Speedup can be achieved in two ways. First, dropping run sequences below a minimum probability threshold (such as 10^{-4}) reduces complexity to $\Theta(L^2)$, where L is the longest run sequence identified in X [2]. Alternatively, the data stream X can be segmented into smaller windows each of size W, reducing the complexity to $\Theta(W^2 \frac{T}{W}) = \Theta(WT)$.

When $W << T$, Algorithm 2 produces near-online results, and a small delay of a few seconds has a negligible effect in this domain. The minimum W required for accurate change detection depends on the sampling rate and underlying appliances. Tuning W for similar methods has been previously explored in [4]. Typically $W > 100$ data points, since setting W too small can yield too little information.

For label correction results in Section 4.3, we did not implement either speedup since the training samples had relatively short durations. For event detection results, we used the second speedup approach. The setting of W is described in Section 5.3.

4. SUPERVISED LABEL CORRECTION

4.1 Sources of Label Errors

The first challenge for supervised classification in electricity disaggregation is to automatically correct labels. A label is a timestamp with an appliance identifier identifying a state transition for that appliance. In this paper, we only consider two states: ON and OFF. However, this can be extended to include intermediate state transitions characteristic of Type II and Type III appliances, detailed in [30].

Imperfect labels in data is a common problem in many domains. Here we focus on the task of correcting poor labels for electricity disaggregation, since this has been completely ignored by other researchers. Outside of our previous work in [27], there has been no discussion of how to deal with incorrect labels or contaminated training samples. Researchers have simply discarded poor samples or manually adjusted erroneous labels in order to focus solely on classification. However, the assumption that a supervised learning method will receive perfectly labeled samples captured in isolation breaks down when the training data is captured by an untrained consumer, and manual corrections do not scale to widespread application. Our approach detailed below may apply to other problems, but will likely require adjustment to domain-specific properties.

Labels can contain errors for multiple reasons (see Figure 2 for examples). First, temporal offsets can occur due to improper clock synchronization or hardware latencies in experimental prototypes. Second, users capturing the data

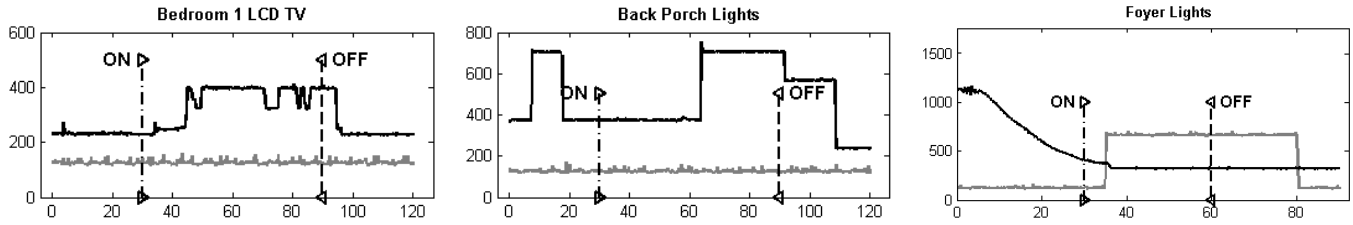

Figure 2: Sources of label error. Real power in both power phase A (black line) and power phase B (gray line) is displayed. Errors may stem from truncated shutdown sequences (left) or external events in the same power phase (middle) or opposite power phase (right). Samples shown also contain offsets from user error or improper synchronization.

Mean Power Increase (W) for Back Porch Lights		
	Before Correction	After Correction
Sample 1	229.7	331.8
Sample 2	145.0	333.15
Sample 3	144.0	330.7
Sample 4	143.1	N/A
μ (mean)	165.5	**331.9**
σ (standard dev.)	37.01	**1.0**
Simple Mean Classification ($\frac{d}{dt}x_t \in [0.9\mu, 1.1\mu]$)		
True Positives	0/4	**4/4**
False Positives	3	**8**
Simple Mean Classification ($\frac{d}{dt}x_t \in [\mu - 2\sigma, \mu + 2\sigma]$)		
True Positives	0/4	**4/4**
False Positives	8	**0**

Table 1: Impact of label errors on classification. Simple mean classification with raw user labels generates only false positives for both L_1 thresholds. Using corrected labels the algorithm finds all true positives using either threshold, and false positives are eliminated with the threshold $\pm 2\sigma$.

can introduce errors by mislabeling data. A consumer may simply forget to mark a device as off after a long period of operation. Third, transient shut down sequences are often truncated when users mark an appliance as off when they turned it off instead of when it stops drawing power. Finally, samples intended to be captured in isolation can be contaminated by other appliances. This typically occurs when a freezer, HVAC system, or other automated appliance turns on when a user is capturing a sample of another appliance.

To show the potential impact of erroneous labels, we use cross-validated simple mean classification on Kaggle's Belkin dataset[1], shown in Table 1. The Belkin dataset contains nearly 150 appliances from 4 houses with up to 4 samples per appliance, but here we use just the 4 samples for the back porch lights of house H3.

Simple mean classification is a supervised learning method that stores the average power consumption μ of each appliance during training. It then classifies an appliance as ON during testing if a power change $\frac{d}{dt}x_t$ is within a specified L_1 distance from μ, where $\frac{d}{dt}x_t$ is the discrete derivative of X evaluated at time t. Table 1 shows results for two L_1 thresholds (10% of the mean and two standard deviations).

Classification accuracy should be high since this is a simple appliance. The lights have only two states (ON and OFF) and significant power consumption of over 300 watts (W), yielding a distinct step function signature. However, training on the four samples without label correction, simple mean classification produces multiple false positives and false negatives for both L_1 intervals. This is due to a significant label offset and contamination of the fourth sample (shown in the middle image in Figure 2). In contrast, using corrected labels for the first 3 samples and discarding the contaminated fourth sample, simple mean classification correctly captures the 330 W operating power, discovers all four true positives, and eliminates false positives with the sufficiently tight interval $\pm 2\sigma$. This illustrates the importance of correctly labeled training samples.

4.2 Label Correction Algorithms

We now introduce the application of BCD to automatically correct labels. Although other change detection methods have previously been applied to electricity disaggregation, our application is unique in at least three ways. First, this is the first application of change detection to correct errors in user ON and OFF labels. Second, BCD does not require manual tuning of parameters. Third, this application has the capability to compute changepoints online, enabling label correction on streaming data as they are observed instead of being restricted to processing in batch.

Given a real power data stream X and user-labeled ON and OFF timestamps (t^u_{on}, t^u_{off}), BCD can extract changepoints from X, but using these changepoints to correct labels (t^u_{on}, t^u_{off}) is not straightforward. A simple approach is to use the changepoint candidate pair $(\hat{t}^{cand}_{on}, \hat{t}^{cand}_{off})$ that minimizes the Euclidean distance with the user labels (t^u_{on}, t^u_{off}), which we refer to as *naive proximal correction*, or *NaiveProx*.

NaiveProx has limited ability to correct labels. Although changepoints can correspond to transitions between ON and OFF states, they can also be produced by fluctuations in background noise or by transitions between different states during appliance operation. These can be steady state transitions of Type II appliances, such as a fan changing between high, medium and low settings, or continuous power changes from Type III appliances, such as changes in a TV's brightness. Furthermore, *NaiveProx* does not account for domain-level properties, such as a power increase for $\hat{t}^{cand}_{on}$, a power decrease for $\hat{t}^{cand}_{off}$, or similar beginning and ending real powers at the new labels $X(\hat{t}^{cand}_{on})$ and $X(\hat{t}^{cand}_{off})$.

Algorithm 3 introduces *online power-constrained proximal correction*, or *OnlinePCProx*, to account for these domain-level properties. The algorithm begins by using online BCD to calculate all changepoints, CP, for the training sample (line 1), builds all unique changepoint pairs (line 2), and sets a flag for when labels have been corrected (line 3). Then, as

[1]https://www.kaggle.com/c/belkin-energy-disaggregation-competition

Algorithm 3: Power-constrained Proximal Correction

Input: Real Power measurements for one training sample in one power phase (X), User-marked on and off timestamps (t^u_{on}, t^u_{off}), Power Threshold (ν).

Output: Corrected Timestamps ($t^{corr}_{on}, t^{corr}_{off}$).

1 $CP = BCD(X)$ *(Note: Can be online or offline BCD)*
2 $CP_{pairs} = \{(t^{cand}_{on}, t^{cand}_{off}) | t^{cand}_{on}, {}^{cand}_{off} \in CP, t^{cand}_{on} < t^{cand}_{off}\}$
3 $Labels\_Corrected = false$
4 **while** *Not(Labels_Corrected)* **and** $CP_{pairs} \neq \emptyset$ **do**
5 $(\hat{t}^{cand}_{on}, \hat{t}^{cand}_{off}) = \underset{(t^{cand}_{on}, t^{cand}_{off}) \in CP_{pairs}}{\arg\min} ||(t^{cand}_{on}, t^{cand}_{off}), (t^u_{on}, t^u_{off})||_2$
6 **if** $\frac{d}{dt}X(\hat{t}^{cand}_{on}) > 0$ **and** $\frac{d}{dt}X(\hat{t}^{cand}_{off}) < 0$ **then**
7 **if** $|X(\hat{t}^{cand}_{on}) - X(\hat{t}^{cand}_{off})| < \nu$ **then**
8 $Labels\_Corrected = true$
9 $t^{corr}_{on} = \hat{t}^{cand}_{on}$
10 $t^{corr}_{off} = \hat{t}^{cand}_{off}$
11 **else**
12 Remove $(\hat{t}^{cand}_{on}, \hat{t}^{cand}_{off})$ from CP_{pairs}
13 **if** $Labels\_Corrected = true$ **then**
14 return $(t^{corr}_{on}, t^{corr}_{off})$
15 **else**
16 *No valid labels found, mark sample as contaminated.*

in *NaiveProx*, it finds the pair of changepoints $(\hat{t}^{cand}_{on}, \hat{t}^{cand}_{off})$ closest to the user on and off labels (t^u_{on}, t^u_{off})(line 5). However, in contrast to *NaiveProx*, it then incorporates the real power draw characteristic of appliances by requiring the candidate ON and OFF labels satisfy two axiomatic constraints.

First, the direction of the power change must be appropriate: a power increase for the ON timestamp $\hat{t}^{cand}_{on}$ and a power decrease for an OFF timestamp $\hat{t}^{cand}_{off}$. This is performed by evaluating $\frac{d}{dt}X$, the first order discrete derivative of X with respect to time, at each changepoint time (line 6). Second, the total change in power must sum to nearly zero (line 7). This is checked using the threshold $\nu = 30W$ instead of a direct comparison with zero to mask background noise. If the labels were successfully corrected, the algorithm then returns the new ON and OFF label pair $(t^{corr}_{on}, t^{corr}_{off})$ (lines 13-14). If it cannot find a valid pair, it marks the sample as contaminated (lines 15-16).

Note the threshold ν is a parameter that could be tuned to optimize performance. However, setting $\nu = 30W$ has been a common practice in previous research [4, 24, 27] since the vast majority of appliances operate above this power level, and any waste from appliances below 30 W is negligible.

Algorithm 3 can use offline BCD instead of online BCD in (line 1) to avoid changepoint time delay discussed in Section 3.2. We refer to this as *offline power-constrained proximal correction*, or *OfflinePCProx*. Also, note that *NaiveProx* is a special case of Algorithm 3 which can be represented by replacing lines 4-16 with just line 5 and returning the result.

4.3 Experimental Setup

Most datasets are unsuitable for label correction evaluation since they are manually post-processed to remove labeling errors prior to being made public, and the original, uncorrected labels are not available. Instead, we chose to use the Kaggle Belkin dataset, since it contains user labeling errors [27] including label offsets, contaminated samples, truncated shutdown sequences, and other mislabeled events.

The Belkin dataset contains disjoint training and testing data. Training data contains 463 training samples from 147 appliances across 4 different houses. Aggregate power measurements are recorded at a 5 Hz rate, and each sample has user ON and OFF timestamped labels. Testing data is similar, but labels remain hidden to the public since the dataset was designed for online competition. As such, we limit the experiments for this paper to the fully labeled training data.

We compare *NaiveProx*, *OnlinePCProx*, and *OfflinePCProx* to our previously introduced *power-constrained fixed increment adjustment*, abbreviated *PCFI* [27]. Unlike the three BCD-based methods introduced here, *PCFI* performs noise reduction prior to label correction and requires a preset parameter to specify the adjustment increment.

4.4 Performance Metrics

A correctly relabeled training sample should capture the entire window of an appliance's operation. As such, we define a sample as correctly relabeled using the equations below. These ensure the corrected ON label (t^{corr}_{on}) occurs before the actual ON label (t^{actual}_{on}), the corrected OFF label (t^{corr}_{off}) occurs after the actual OFF label (t^{actual}_{off}), and both are within a specified distance δ of the actual labels.

$$Corr(t^{corr}_{on}) = \begin{cases} Correct, & \textbf{if } t^{corr}_{on} \in [t^{actual}_{on} - \delta, t^{actual}_{on}] \\ Incorrect, & \text{otherwise} \end{cases}$$

$$Corr(t^{corr}_{off}) = \begin{cases} Correct, & \textbf{if } t^{corr}_{off} \in [t^{actual}_{off}, t^{actual}_{off} + \delta] \\ Incorrect, & \text{otherwise} \end{cases}$$

$$Relabeling = \begin{cases} Correct, & \textbf{if } (Corr(t^{corr}_{on}) \textbf{ and } Corr(t^{corr}_{off})) \\ Incorrect, & \text{otherwise} \end{cases}$$

Since the Belkin dataset only contains uncorrected ON and OFF labels, we visually established the actual ON and OFF labels for each training sample prior to label correction. Although this process is time-consuming, it is inherently obvious where the actual ON and OFF labels are for the overwhelming majority of training samples. In the handful of cases where the actual ON and OFF labels was in doubt, newly generated labels were recorded as incorrect.

We experimented with $\delta \in [0.5, 5]$ seconds and observed similar results. Results detailed below are for $\delta = 2$ seconds which equates to 10 sampling points at Belkin's 5 Hz sampling rate. We then use the percent of samples correctly relabeled to compare label correction methods.

4.5 Label Correction Results

Figure 3 displays results for label correction, comparing the three Bayesian change detection based methods introduced in this paper to *PCFI* [27] using the Kaggle Belkin dataset. Note that variation of performance between houses is expected, since each house has appliances with different characteristics, further amplified by differing user behavior.

Unsurprisingly, *NaiveProx* performs poorly since it produces new labels from changepoints based solely on their distance from user labels. Any small changes in power near a user label are returned as the new ON or OFF label, and there are no guarantees these labels possess the right change in direction, constitute real power draw, or sum to 0 watts.

OnlinePCProx and *OfflinePCProx* perform significantly better. *OnlinePCProx* results in slightly lower accuracy due to the potential time delay d detailed in Section 3.2, which can push a new label outside accepted threshold δ for low power appliances. In contrast, *OfflinePCProx* retroactively identifies the origin of each run sequence after it has been identified, and its results are comparable to *PCFI* [27].

These results are significant since they are achieved in an automated fashion without requiring prior noise reduction or parameter tuning. *OnlinePCProx* and *OfflinePCProx* leave some samples uncorrected, but they successfully correct the overwhelming majority, and correction of 100% of samples is not necessary. While the number of training samples required for accurate classification varies depending on the supervised learning method, providing fewer correctly labeled samples produces better classification accuracy than more incorrectly labeled samples, as illustrated earlier in Table 1.

Furthermore, it is unrealistic to expect any algorithm to correct all training samples. Table 2 lists some of the causes of uncorrected labels using *OfflinePCProx*. Label correction is infeasible for eighty of these samples since they have significant obstacles for any label correction method. Obstacles include very low power draw (15 W or less) making appliance signatures indistinct from baseline background noise, contamination by external events, or gross mislabeling where either the timestamps for both ON and OFF labels are identical or labels occur inside a different appliance sample.

Lastly, BCD-based methods can be used in real time with inexpensive hardware. Belkin's training samples range from 40 seconds to 1.5 hours, and Algorithm 3 on a conventional laptop completed in under 5 seconds for the smallest samples and in a few minutes for the longest samples.

5. UNSUPERVISED EVENT DETECTION

Although supervised learning methods yield high classification accuracy, they can be cumbersome to implement. To obtain isolated training samples, consumers must turn off all appliances and switch each appliance on one at a time. Some appliance cycles last an hour or longer (such as dishwashers, washing machines, or dryers), and most homes contain 30-40 different appliances, meaning consumers must dedicate 8-12 hours or longer to obtain even a few isolated training samples for each appliance. Many appliances cannot be turned off since they perform essential functions (such as refrigerators, freezers, or HVAC systems), and automated appliances in commercial buildings are so numerous that capturing isolated training samples may not be feasible. These are the motivations behind unsupervised disaggregation [30].

As mentioned in Section 2, unsupervised learning in electricity disaggregation consists of, at a minimum, two steps: The real power data stream must first be segmented to identify significant events. This process is referred to as event detection [4, 20]. After events have been identified, they are reconstructed into individual, previously unknown appliances. The remainder of this section focuses on the first step of event detection.

5.1 Event Detection Algorithms

We evaluate BCD's ability to detect electrical events in real power data and compare its performance against modified greatest likelihood ratio (mGLR) and Dirichlet process Gaussian mixture models (DPGMM).

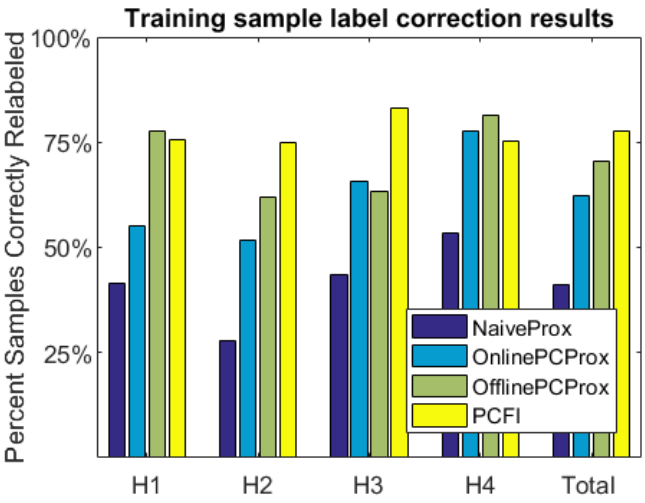

Figure 3: Supervised label correction algorithms applied to all four houses H1-H4 in the Kaggle Belkin dataset.

Incorrect Labels	H1	H2	H3	H4	Total
Total incorrect labels	25	51	45	23	144
Can be corrected	8	22	26	8	64
Cannot be corrected	17	29	19	15	80
Low power (<15W)	12	10	13	13	48
Sample contaminated	4	3	4	2	13
Gross mislabeling	1	16	0	0	17
Sporadic power draw	0	0	2	0	2

Table 2: Causes of uncorrected labels using *OfflinePCProx*.

mGLR is a change detection method that uses log likelihood ratios in conjunction with a voting scheme to identify events. It has five tunable parameters which were optimized for performance on the BLUED dataset in [4].

DPGMM is an iterative clustering method that treats real power values in X as an infinite-mixture GMM. For each discovered Gaussian, it determines steady-state power levels by computing its mean. It then recasts observed power values X to the nearest steady-state power level and identifies changepoints as transitions between these states [24].

Following BCD's event detection, we filtered out small changepoints caused by background noise using the well-established power threshold $\nu = 30W$, the same threshold used by mGLR and DPGMM [4, 24]. Finally, since mGLR and DPGMM are offline methods, we restrict the BCD results in this section to offline computation (Algorithm 2) and use the windowed speedup introduced in Section 3.3 while noting this produces near-online results.

5.2 Performance Metrics

A sample x_t is defined as a positive if an event is detected and as a negative if no event is detected. Given a predefined δ, a positive x_t is a true positive *iff* $\exists x_{gte}$ s.t. $|x_t - x_{gte}| < \delta$, where x_{gte} is a ground truth event. Otherwise, it is a false positive. Conversely, a negative x_t is a true negative *iff* $\forall x_{gte}\ |x_t - x_{gte}| > \delta$, and is a false negative otherwise. Similar to label correction, results below use $\delta = 2$ seconds.

Due to the overwhelming number of true negatives inherent in event detection, some accuracy metrics yield poor results [4]. As such, we chose four metrics that do not rely on true negatives. The first, $F\_Measure$, is defined in (1) using

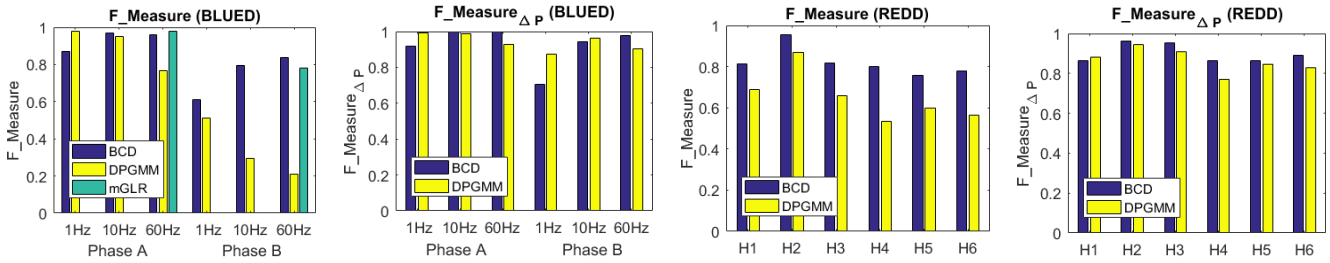

Figure 4: Weighted and unweighted F_Measures for event detection on the BLUED (left) and REDD (right) datasets. In general, BCD performs as well as or better than DPGMM and mGLR without requiring parameter tuning or noise reduction. The impact of different sampling rates can be seen in BLUED. As the sampling rate increases, DPGMM's accuracy decreases due to numerous false positives caused by higher noise. In contrast, BCD's performance increases or stays the same.

precision and recall. TP, FP, and FN are the sets of true positives, false positives, and false negatives, respectively.

$$Precision = \frac{|TP|}{|TP| + |FP|}, \quad Recall = \frac{|TP|}{|TP| + |FN|}$$

$$F\_Measure = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{1}$$

Second, we introduce a power-weighted $F\_Measure_{\Delta P}$ in (2). ΔP_{tp}, ΔP_{fp}, and ΔP_{fn} are the power changes for each true positive, false positive, and false negative, respectively.

$$\Delta P_{TP} = \sum_{tp \in TP} |\Delta P_{tp}|, \quad \Delta P_{FP} = \sum_{fp \in FP} |\Delta P_{fp}|, \quad \Delta P_{FN} = \sum_{fn \in FN} |\Delta P_{fn}|$$

$$Precision_{\Delta P} = \frac{\Delta P_{TP}}{\Delta P_{TP} + \Delta P_{FP}}, \quad Recall_{\Delta P} = \frac{\Delta P_{TP}}{\Delta P_{TP} + \Delta P_{FN}}$$

$$F\_Measure_{\Delta P} = \frac{2}{\frac{1}{Precision_{\Delta P}} + \frac{1}{Recall_{\Delta P}}} \tag{2}$$

We also include two error metrics from [4] that have superior performance in evaluating event detection: True Positive Percent ($\psi_{Percent}$) and Total Power Change ($\psi_{\Delta P}$).

$\psi_{Percent}$ measures the ratios of true positives and false positives to total ground truth events, GTE, as in (3). An optimal detector finds all true positives and produces no false positives, so the error rate $\psi_{Percent}$ is computed as the Euclidean distance between the detector and the point (1,0).

$$\psi_{Percent} = ||(1,0) - (\frac{|TP|}{|GTE|}, \frac{|FP|}{|GTE|})||_2 \tag{3}$$

Finally, $\psi_{\Delta P}$ measures the sum of the power changes of all false positives and false negatives, defined in (4). Since an optimal detector would produce no false positives or false negatives, the evaluation for $\psi_{\Delta P}$ again uses the Euclidean norm, but this time from the point (0,0).

$$\psi_{\Delta P} = ||(\Delta P_{FP}, \Delta P_{FN})||_2 \tag{4}$$

Note that by the definition of a true positive above, there can be multiple true positives for a single ground truth event, making it possible that $|TP| > |GTE|$. This is reasonable since the exact time when an appliance changes states is not always clear, making the actual x_{gte} somewhat subjective. Many false positives can also make $\psi_{Percent} > 100\%$ [4].

5.3 Experimental Setup

The Belkin dataset is poorly suited to evaluate event detection since its training data only contains isolated appliances and its testing data labels are hidden. Instead, we use the publicly available BLUED (Building-Level fUlly-labeled dataset for Electricity Disaggregation) and REDD (Reference Energy Disaggregation Dataset) datasets [3, 18].

BLUED provides event labels corrected through manual post-processing. Power is captured at 60 Hz and spans one week on a house with 43 appliances [3]. To measure the effect of sampling rate on event detection, we used the original 60 Hz and also downsampled power to 1 Hz and 10 Hz.

REDD [18] contains power measurements from 6 houses at a sampling rate of roughly 0.25 Hz, but contains individual data streams for each appliance instead of ground truth event labels. As such, we first performed event detection using BCD and DPGMM on the individual appliance data streams and labeled an event as a ground truth event if it was discovered by both. We then compared these ground truth labels with events detected from the aggregate real power measurements to determine event detection accuracy.

For fairest evaluation, we did not recreate mGLR and instead used results reported in [4]. This limits comparisons of BCD and mGLR in Figures 4 and 5 to the BLUED dataset using two of the previously defined metrics (F_Measure and $\psi_{Percent}$) at a 60 Hz sampling rate. However, these comparisons are significant since mGLR's results were obtained using parameters optimized for the BLUED dataset.

In contrast, we re-implemented DPGMM since its intermediate event detection results were not reported in [24]. We omitted the pre-processing noise reduction step which uses a heuristic-based window filter since its parameters were tuned for individual appliances and were not published in [24].

The authors of mGLR experimented with window sizes of $W \in [120, 330]$ on the BLUED dataset [4] and optimized $W = 120$ for their results reported in [3] and shown in Figure 4 and Figure 5. For BCD and DPGMM, we experimented with window sizes $W \in [100, 3,000]$ in increments of 100 data points. $W > 3,000$ significantly increased runtime for both methods while providing no noticeable improvement.

BCD's results showed little variation for varying W, since the run sequence probabilities from the previous time window carried over to the current window. In contrast, DPGMM's performance varied widely over differing values of W since it discovered different clusters depending on where the data stream was segmented. Event detection results use $W =$ 1,000 data points since it provided the best results for DPGMM.

5.4 Event Detection Results

Figure 4 displays event detection results using $F\_Measure$ and $F\_Measure_{\Delta P}$ on both power phases A and B of BLUED using sampling rates of 1 Hz, 10 Hz, and 60 Hz (left) and

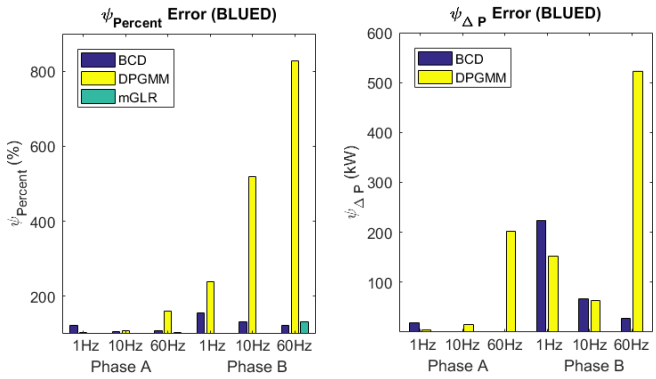

Figure 5: Error rates in event detection on BLUED using unweighted error, $\psi_{Percent}$ (left) and power-weighted error, $\psi_{\Delta P}$ (right). BCD's error is near zero in some cases and its $\psi_{Percent}$ error is similar to mGLR at 60 Hz, while DPGMM's error is larger on phase B and at higher sampling rates due to numerous low power false positives.

REDD's six houses (right). mGLR is only compared using 60 Hz in the leftmost figure for reasons described above.

On BLUED, all methods perform better in general on power phase A than on phase B. This is expected, since phase A contains simpler appliances that are more isolated, while phase B contains noisier appliances with concurrent operation [3]. On REDD, variation between houses is likely due to different appliance characteristics and user behavior. These results are statistically significant and support the hypothesis that BCD performs as well as or better than mGLR and DPGMM on both datasets using a p-value of 0.05.

$\psi_{Percent}$ and $\psi_{\Delta P}$ are displayed for BLUED in Figure 5 and for REDD in Figure 6. These results are included since both error rates were developed specifically for this domain [4]. Both show BCD's error rate is the same or lower than competing methods and are also statistically significant.

Note that BCD's performance generally improves with higher sampling rates on BLUED in Figures 4 and 5. In contrast, DPGMM's performance actually decreases with higher sampling rates, and DPGMM performs better with the power-weighted metrics $F\_Measure_{\Delta P}$ and $\psi_{\Delta P}$ than with unweighted metrics $F\_Measure$ and $\psi_{Percent}$. This behavior is caused by large numbers of low power false positives generated by DPGMM, illustrated in Figure 7.

Recall that DPGMM treats the power spectrum as a Gaussian mixture model. Under noisy conditions, it can falsely cluster small power variations into different power states and identify changes between these false states as events. While this is likely amplified by omitting the heuristic noise reduction step described in Section 5.1, it is not realistic to tune parameters for such an approach to individual appliances. BCD avoids this noise handicap since every possible run sequence is modeled with its own mean and variance.

We observed efficient runtimes for BCD on both datasets. While BCD and DPGMM are both $\Theta(T^2)$ algorithms, BCD consistently completed 20-40 times faster than DPGMM. Although coding implementations influence runtimes, this performance fits with computational analysis. BCD requires a single computation to update each existing run sequence r_t, while DPGMM iteratively clusters points until its GMM converges or it reaches a maximum number of iterations.

Finally, BCD does not require parameter tuning. The

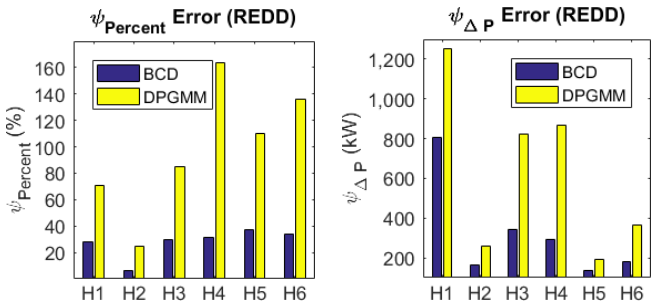

Figure 6: Error rates in unsupervised event detection on the six houses of the REDD dataset using unweighted $\psi_{Percent}$ and $\psi_{\Delta P}$. Similar to results on BLUED, DPGMM generates large numbers of low power false positives on all houses.

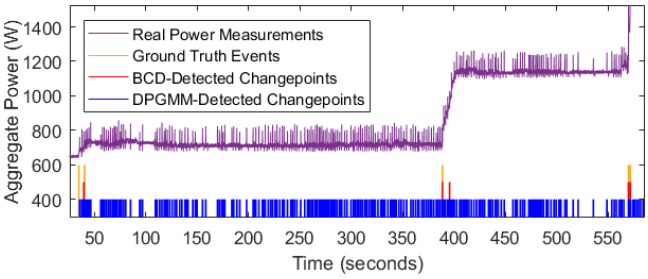

Figure 7: Example of event detection under very noisy conditions. Ground truth events (orange tick marks) occur at 34, 50, 388, 570, and 571 seconds. BCD (red tick marks) generates 1 false positive and 1 false negative, while DPGMM (blue tick marks) produces over 500 false positives.

domain-agnostic values for its algorithmic parameters are sufficient (detailed in Section 3.2). BCD is robust to various window sizes W since run sequence probabilities carry over to new windows, and the event detection threshold $\nu = 30W$ is commonly established [3, 24, 27]. In addition, BCD completed in less than 6 hours for BLUED's 1 week of data on a conventional laptop using highest sampling rate of 60 Hz, enabling real-time computation with inexpensive hardware.

6. CONCLUSIONS AND FUTURE WORK

We have introduced the application of Bayesian change detection to label correction and event detection in electricity disaggregation. BCD's performance is on par with or better than state-of-the-art methods using multiple metrics on a variety of houses and appliances, and it can be run near-online and in real time. BCD also does not require parameter tuning to specific appliances or datasets, is robust to varying noise, and its performance improves with higher sampling rates. Our BCD implementation and other resources are available at `www.cs.umn.edu/~valovage/AAMAS-2017`.

Future label correction work includes recovering data from contaminated samples and developing datasets to model user label errors. Additional electrical features can also be incorporated, and the complexity to reconstruct appliances from detected events during unsupervised learning remains a challenging problem. Finally, as poor labeling is not unique to electricity disaggregation, the application of BCD to additional domains should be explored further to produce robust, scalable solutions that do not require parameter tuning.

REFERENCES

[1] Bayesian online changepoint detection. `http://hips.seas.harvard.edu/content/bayesian-online-changepoint-detection`. Harvard Intelligent Probabilistic Systems website.

[2] R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[3] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges. BLUED: A fully labeled public dataset for event-based non-intrusive load monitoring research. In *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, pages 1–5, 2012.

[4] K. D. Anderson, M. E. Bergés, A. Ocneanu, D. Benitez, and J. M. Moura. Event detection for non-intrusive load monitoring. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 3312–3317. IEEE, 2012.

[5] M. Berges, E. Goldman, H. S. Matthews, and L. Soibelman. Training load monitoring algorithms on highly sub-metered home electricity consumption data. *Tsinghua Science & Technology*, 13:406–411, 2008.

[6] K. Carrie Armel, A. Gupta, G. Shrimali, and A. Albert. Is disaggregation the holy grail of energy efficiency? The case of electricity. *Energy Policy*, 2012.

[7] G. Comert and A. Bezuglov. An online change-point-based model for traffic parameter prediction. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1360–1369, 2013.

[8] E. Elhamifar and S. Sastry. Energy disaggregation via learning 'powerlets' and sparse coding. In *Proc. AAAI Conf. on Artificial Intelligence*, 2015.

[9] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel. Disaggregated end-use energy sensing for the smart grid. *Pervasive Computing, IEEE*, 10(1):28–39, 2011.

[10] H. Gonçalves, A. Ocneanu, M. Bergés, and R. Fan. Unsupervised disaggregation of appliances using aggregated consumption data. In *The 1st KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, 2011.

[11] H. C. Granade, J. Creyts, A. Derkach, P. Farese, S. Nyquist, and K. Ostrowski. Unlocking energy efficiency in the US economy. *McKinsey & Company*, 2009.

[12] G. W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[13] S. Heck and H. Tai. Sizing the potential of behavioral energy-efficiency initiatives in the US residential market. *McKinsey & Company*, 2013.

[14] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215. ACM, 2004.

[15] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han. Unsupervised disaggregation of low frequency power measurements. In *SIAM International Conference on Data Mining*, volume 11, pages 747–758. SIAM, 2011.

[16] J. Z. Kolter, S. Batra, and A. Y. Ng. Energy disaggregation via discriminative sparse coding. In *Advances in Neural Information Processing Systems*, pages 1153–1161, 2010.

[17] J. Z. Kolter and T. Jaakkola. Approximate inference in additive factorial HMMs with application to energy disaggregation. In *International Conf. on Artificial Intelligence and Statistics*, pages 1472–1482, 2012.

[18] J. Z. Kolter and M. J. Johnson. REDD: A public data set for energy disaggregation research. In *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, volume 25, pages 59–62, 2011.

[19] F. H. Malik and M. Lehtonen. A review: Agents in smart grids. *Electric Power Systems Research*, 131:71–79, 2016.

[20] A. N. Milioudis, G. T. Andreou, V. Katsanou, K. I. Sgouras, and D. P. Labridis. Event detection for load disaggregation in smart metering. In *Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, pages 1–5. IEEE, 2013.

[21] O. Parson, S. Ghosh, M. Weal, and A. Rogers. Non-intrusive load monitoring using prior models of general appliance types. In *Proc. AAAI Conf. on Artificial Intelligence*, 2012.

[22] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *International Conf. on Ubiquitous Computing*, pages 271–288. Springer, 2007.

[23] J. Seryak and K. Kissock. Occupancy and behavioral affects on residential energy use. In *Proc. of the Solar Conference*, pages 717–722. American Solar Energy Society; American Institute of Architects, 2003.

[24] H. Shao, M. Marwah, and N. Ramakrishnan. A temporal motif mining approach to unsupervised energy disaggregation: Applications to residential and commercial buildings. In *Proc. AAAI Conf. on Artificial Intelligence*, 2013.

[25] R. H. Socolow. The twin rivers program on energy conservation in housing: Highlights and conclusions. *Energy and Buildings*, 1(3):207–242, 1978.

[26] R. Turner. Bayesian change point detection for satellite fault prediction. *Diverse Engagement: Drawing in the Margins*, page 213, 2010.

[27] M. Valovage and M. Gini. Automatic label correction and device prioritization in single household electricity disaggregation. *International Workshop on Artificial Intelligence for Smart Grids and Smart Buildings, at AAAI*, 2016.

[28] M. Zeifman and K. Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Transactions on Consumer Electronics*, 57(1):76–84, 2011.

[29] M. Zeifman and K. Roth. Viterbi algorithm with sparse transitions (VAST) for nonintrusive load monitoring. In *IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG)*, pages 1–8. IEEE, 2011.

[30] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012.