Article

An Incremental Clustering Algorithm with Pattern Drift Detection for IoT-Enabled Smart Grid System

Zigui Jiang [1], Rongheng Lin [2,*] and Fangchun Yang [2]

[1] School of Software Engineering, Sun Yat-Sen University, Zhuhai 519082, China; jiangzg3@mail.sysu.edu.cn
[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; fcyang@bupt.edu.cn
* Correspondence: rhlin@bupt.edu.cn

Abstract: The IoT-enabled smart grid system provides smart meter data for electricity consumers to record their energy consumption behaviors, the typical features of which can be represented by the load patterns extracted from load data clustering. The changeability of consumption behaviors requires load pattern update for achieving accurate consumer segmentation and effective demand response. In order to save training time and reduce computation scale, we propose a novel incremental clustering algorithm with probability strategy, ICluster-PS, instead of overall load data clustering to update load patterns. ICluster-PS first conducts new load pattern extraction based on the existing load patterns and new data. Then, it intergrades new load patterns with the existing ones. Finally, it optimizes the intergraded load pattern sets by a further modification. Moreover, ICluster-PS can be performed continuously with new coming data due to parameter updating and generalization. Extensive experiments are implemented on real-world dataset containing diverse consumer types in various districts. The experimental results are evaluated by both clustering validity indices and accuracy measures, which indicate that ICluster-PS outperforms other related incremental clustering algorithm. Additionally, according to the further case studies on pattern evolution analysis, ICluster-PS is able to present any pattern drifts through its incremental clustering results.

Keywords: incremental learning; data stream clustering; load pattern; smart meter data

check for updates

Citation: Jiang, Z.; Lin, R.; Yang, F. An Incremental Clustering Algorithm with Pattern Drift Detection for IoT-Enabled Smart Grid System. *Sensors* **2021**, *21*, 6466. https://doi.org/10.3390/s21196466

Academic Editor: Antonino Laudani

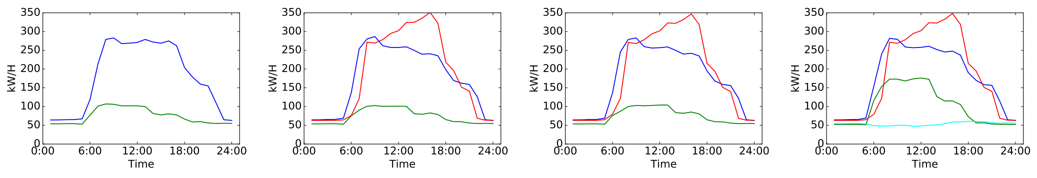
Received: 27 July 2021
Accepted: 24 September 2021
Published: 28 September 2021

1. Introduction

The smart grid system has been developing with the integration of massive new technologies, such as Internet of Things (IoT), Blockchain, and Artificial Intelligence (AI) [1–3]. Diverse IoT devices and frameworks are applied on smart grid to support data collection, transmission [4], real-time monitoring [5], etc. Blockchain technologies can provide decentralization, trust, and an incentive mechanism for improving the cybersecurity of smart grid system [6–8]. Compared with AI, the applications of AI methods including machine learning and deep learning are usually used to process and analyze data for decision-making, such as electric load forecasting [9,10], electric consumer categorization [11], and anomaly detection [12]. In such a smart grid system, the smart meter is an essential IoT device that records energy consumption data for further understanding, managing, planing, and optimizing power demands of electric consumers [13,14].

Smart meter data, also called electricity load data, are data streams that record the electricity consumption behaviors of consumers at regular intervals. They can be used for various studies and applications in smart grid, such as load forecasting, load profiling [15], anomaly detection [16,17], consumer categorization [18], and energy disaggregation [19]. In the studies of load profiling, one significant purpose is to extract the typical electricity consumption patterns, which is usually called load patterns, of every consumer based on load data clustering [20]. Most of works on load data clustering focus on the clustering problem of static load data. However, we notice that updating load patterns based on new load data is essential because electricity consumption behaviors may be changeable and inaccurate load patterns can cause wrong decisions. Although load patterns can be updated by conducting repetitive clustering on overall load data including the new ones, this leads to extra computation and storage, especially in batch-oriented data processing. In that case, incremental learning, which refers to learning from streaming data that arrive over time [21], can be a better solution as it can make full use of the historical information, reduce the training scale, and save training time [22]. Moreover, there are also some special clustering algorithms designed for data streams mining [23]. However, few of them are designed for high-dimensional smart meter data streams so that it is necessary to find out an effective incremental clustering algorithm to update load patterns, especially for end consumers with limited resources.

In real-world industry and our daily lives, the electricity consumption behaviors of consumers may change over time. Some consumers keep their patterns for a long period while others may change frequently. An example of load pattern drift is shown in Figure 1. Each curve denotes a typical load pattern of the same electricity consumer, and the curves in the same color in different subfigures indicate the same load pattern. This consumer has two typical load patterns from January to July, which means that this consumer has a stable electricity consumption behavior. Then, it can be observed that the load patterns drift twice. The first drift happened in August shown by the red curve in Figure 1b, which indicates that this consumer has a new electricity consumption behavior. The second drift happened in August shown by the cyan curve in Figure 1d. Then, this consumer has four electricity consumption behaviors since October.



(**a**) January–July (**b**) January–August (**c**) January–September (**d**) January–October

Figure 1. An example of electricity load patterns drift. (**a**) Load patterns of load data from January to July; (**b**–**d**) load patterns of load data that increase monthly comparing with the former one. All load patterns are obtained by daily load curve clustering.

Once we extract the load patterns from the static electricity load data in a certain period, these load patterns are fixed unless they are updated. It is possible that there are some new load patterns that denote consumer behavior drift in the following periods, so that we should update the previously obtained load patterns by adding the new ones. However, consumer behaviors are complex. It is still uncertain that all coming load patterns are new, which means that some load patterns may exist in the previously obtained load patterns and others may not. In that case, we cannot simply add each load pattern extracted from the new coming data or assign them to any existing load patterns. How to update load patterns accurately is the main challenge of our incremental clustering problem.

Therefore, this work proposes an incremental clustering algorithm with probability strategy, which is named ICluster-PS. We assume that this algorithm can deal with smart meter data streams to update load patterns efficiently for every end-consumer through facilities with limited time and space. The incremental clustering algorithm of ICluster-PS includes three phases: load pattern extraction, load pattern intergradation, and load pattern modification. Load pattern extraction is a preparation to extract load patterns from new electricity load data, which are preprocessed as daily load curves. Load pattern intergradation and modification is an novel approach for determining whether or not we should create a new load pattern and optimize K for the number of updated load patterns. A short paper of this work is published in [24], and we revise and extend it by adding more details of the algorithm, experiments and pattern evolution analysis in this paper. The main contributions of this work are summarized as follows:

- We consider the problem of load pattern update based on smart meter data streams, and propose an incremental clustering algorithm for continuously updating load patterns. It is significantly helpful for learning electricity consumption behaviors in smart grid field.
- In the incremental clustering algorithm, we propose a probability strategy on distance measure for optimizing the performance of incremental clustering, and also consider updating parameter to conduct continuous incremental clustering with new coming data.
- We evaluate both accuracy and clustering validity of our algorithm on a real-world dataset, which contains 17,776 commercial and residential electricity consumers in various districts. The results indicate that ICluster-PS is closed to the performance of the non-incremental clustering based on overall daily load curves and outperforms other related incremental clustering algorithm in terms of both clustering validity and accuracy.
- The load pattern evolution can be clearly presented by the incremental clustering results, in which we are able to detect any pattern drifts or anomalies of electricity consumers.

The rest of this paper is organized as follows. Section 2 briefly reviews the related works. Section 3 provides the preliminary for Section 4, which introduces the details of the proposed incremental clustering algorithm. Experimental settings are presented in Section 5, and results with evaluation are discussed in Section 6. Finally, we conclude this work in Section 7.

2. Related Work

This section briefly reviews the most relevant related works in terms of load pattern extraction, incremental learning algorithms, and data stream clustering. Electricity consumer load pattern extraction is one of the most important research areas in smart grid, while incremental learning and data stream clustering are two related research areas in machine learning and data mining. However, there are few works that consider the problem how to conduct an incremental learning for electricity consumer load pattern extraction. Some relevant research works are compared in Table 1.

Table 1. Comparison with existing research works.

Research Works	Year	Algorithms	Unsupervised	Data Types	Incremental	New Class
Xu et al. [22]	2018	SVM-based	No	Multiple	Yes	No
Jiang et al. [24]	2019	*K*-means-based	Yes	load data	No	-
Al-Otaibi et al. [25]	2016	feature construction	Yes	load data	No	-
Panapakidis et al. [26]	2015	*K*-means-based	Yes	load data	No	-
Marxer & Purwins [27]	2016	RF-based	No	image data	Yes	Yes
Zhang et al. [28]	2017	density-based	Yes	IoT data	Yes	Yes
Aggarwal et al. [29]	2004	*K*-means-based	Yes	data streams	-	-
Kriegel et al. [30]	2011	density-based	Yes	data streams	-	-
Zhang et al. [31]	2016	Fuzzy C-mean-based	Yes	data streams	-	-
Braverman et al. [32]	2017	*K*-median-based	Yes	data streams	-	No
Hyde et al. [33]	2017	density-based	Yes	data streams	Yes	-
Zhang et al. [34]	2019	density-based	Yes	trajectory streams	Yes	-
W. & Berrar [35]	2020	neural network-based	Yes	noisy data	-	-
Proposed work	2021	*K*-means-based	Yes	load data	Yes	Yes

Load Pattern Extraction. Load pattern extraction is an unsupervised clustering problem. There are two types of clustering methods for load data clustering: direct clustering and indirect clustering [15]. In direct clustering, load data are directly used in clustering without any additional dimension reduction or data preprocessing methods. There are many classical clustering algorithms for load data clustering, such as *K*-means, fuzzy

K-means, self-organizing map (SOM), and support vector clustering (SCV) [36–38]. As for indirect clustering, researchers usually pay more attentions to dimension reduction, feature extraction and feature construction methods for load data preprocessing. In [25], the authors constructed three new types of features. Their work indicates that the clustering performance of constructed features outperforms the one of default features. In [26], two variations of K-means algorithm with four proposed dimension reduction methods are applied to the clustering process in load profiling. A fused load curve clustering algorithm based on wavelet transform (FCCWT) is proposed in our previous work [39]. This algorithm first applies a multi-level wavelet transform to daily load curves for dimension reduction, and then fuse the K-means clustering results of both normalized approximation signals and detail signals, which are two outputs of wavelet transform, to gain an optimized clustering result.

Incremental Learning Algorithms. In recent years, incremental and online learning gain more attentions especially in big data and data stream areas [40,41]. There are many incremental learning algorithms based on ν-support vector regression, support vector machines (SVM), random forest (RF), neural networks, etc. [27,42–44]. An incremental support vector machine (ISVM) with Markov resampling (MR-ISVM) is introduced in [22] to study how dependent sampling methods influence the learning ability of ISVM. However, most of incremental learning algorithms study supervised classification without adding new classes. Although an incremental learning based on RF is studied to incrementally learn new classes for large-scale image [27], this method adds new classes into the trees without judging whether or not the coming classes are new. In [28], the authors proposed an incremental algorithm based on fast finding and searching of density peaks (CFS), named ICFKM, for clustering large data in industrial IoTs. Two challenges—how to integrate new clusters into the previous one and how to update the clustering centers—are solved in ICFKM, which seems to be useful for our incremental clustering problem. However, CFS has relatively strong subjectivity for selecting cluster centers based on the decision graph [45] so that it cannot applied in batch-oriented data processing. Moreover, CFS does not work well on relatively high-dimensional data. Many clusters may be missed by CPS because it only considers the global structure of data [46]. As time-series electricity load data have relatively high dimensions, ICFKM cannot be directly adopted for updating load patterns.

Data Stream Clustering. Clustering data streams requires the capability of partitioning observations continuously within limited memory and time [47]. Most data stream clustering algorithms consist of an online step that incrementally processes the data stream and produces summary statistics, and an offline step that summarizes data to generate clusters by traditional batch clustering algorithms [48]. There are various classic data stream clustering algorithms, such as Stream, CluStream, StreamKM++, DenStream, and HPstream. Both HPstream [29] and incPreDeCon [30] can deal with high-dimensional data streams. The former algorithm is based on K-means, while the later one is based on PreDecom which is a density-based clustering algorithm and requires too many parameters to be run efficiently. In [31], the authors introduced a data stream clustering based on Fuzzy C-mean algorithm and entropy theory. In [32], the authors developed algorithms for clustering high-dimensional dynamic data streams, whereas the algorithms are based on the assumption that no insertions of data that are already in the dataset, which may be not consistent with our load data. Meanwhile, the efficiency of these proposed algorithms is only evaluated by a 2D implementation. In [33], a fully online clustering algorithm is proposed for clustering evolving data streams into arbitrarily shaped clusters (CEDAS), which is also a density-based clustering algorithm. In [34], a density-based clustering algorithm called DStream-GC is designed for discovering gradual moving object clusters pattern from trajectory streams. In [35], a self-organizing incremental neural network (SOINN+) is developed for unsupervised learning clusters with arbitrary shapes from noisy data. Although some algorithms are incremental methods or declare that they can process high-dimensional data streams, their validity and efficiency on load pattern extraction and update require further evaluation. For example, density-based clustering algorithms may not achieve an excellent performance in the experiments of high-dimensional load curve clustering.

In summary, the works on load pattern extraction do not consider the incremental learning problem in their clustering algorithm, while the existing incremental learning or data stream clustering algorithms are not designed for load clustering. Therefore, it is essential to provide an incremental clustering algorithm for our load clustering problem.

3. Preliminary

Before introducing our method, we should first give the problem formulation and several important mathematical notations, shown in Table 2. We also briefly present the method used for load pattern extraction, which is the base of electricity consumer behavior learning.

Table 2. Several important mathematical notations.

Notations	Description
X	the set of overall daily load curves
X_s	the jth set of daily load curves, $0 \le s \le t$
x_{si}	the ith daily load curve in X_s, $1 \le i \le N_s$
d	No. of dimensions of daily load curves
t	No. of coming new daily load curves
N_s	No. of daily load curves in X_s
C_s	the set of clusters obtained from a load curve clustering on $[X_0, X_1, \cdots, X_s]$
C_{si}	the ith cluster in C_s, $1 \le i \le K_s$. The cluster center of C_{si} is μ_{si}
n_{si}	No. of daily load curves in C_{si}
A_s	the set of cluster centers, also called load patterns, of C_s
μ_{si}	the ith load patterns in A_s, referring to the cluster center of C_{si}, $1 \le i \le K_s$
K_s	No. of load patterns in A_s / clusters in C_s
P_s	the set of probabilities of load patterns A_s
p_{si}	the probability of μ_{si}, $p_{si} \in P_s$, $\sum_{i=1}^{K_s} p_{si} = 1$
X_0	the set of initial daily load curves, $X_0 \in X$
X_1	the first set of new daily load curves, $X_1 \in X$
A_0	the set of load patterns obtained from a load curve clustering on X_0
a_1	the set of load patterns obtained from a load curve clustering on X_1
iA_1	the set of load patterns obtained from load pattern intergradtion on $[A_0, a_1]$
A_1	the set of updated load patterns obtained from the incremental clustering on $[X_0, X_1]$

3.1. Problem Formulation

For an electricity consumer, let $X_0 = \{x_{01}, x_{02}, \ldots, x_{0N_0}\} \in \mathbb{R}^{d \times N_0}$ where x_{0i} is d-dimensional vector be the electricity load data and N_0 be the number of days contained in the dataset X_0. We can extract the load patterns from these data by conducting daily load curve clustering.

Definition 1 (Daily Load Curve)**.** *A daily load curve* $x_{0i} = <x_{0i1}, x_{0i2}, \ldots, x_{0id}>$ *where* $1 \le i \le N_0$ *is a d-dimensional vector that presents the electricity power consumption of one consumer in one day. It is recorded by a smart meter at a regular interval, which usually is 1 h, 30 min, or 15 min.*

Definition 2 (Load Pattern)**.** *Given a set of daily load curves* $X_0 = \{x_{01}, x_{02}, \ldots, x_{0N_0}\} \in \mathbb{R}^{d \times N_0}$*, we apply a load curve clustering to* X_0 *and obtain a set of clusters* $C_0 = \{C_{01}, C_{02}, \ldots, C_{0K_0}\}$*. Let* $A_0 = \{\mu_{01}, \mu_{02}, \ldots, \mu_{0K_0}\} \in \mathbb{R}^{d \times K_0}$ *be the set of cluster centers of* C_0*, and each* μ_{0i} *is called a load pattern that denotes one typical electricity power consumption behavior feature of the consumer. Every electricity consumer may have one or several load patterns.*

As X_0 contains N_0 daily load curves which are divided into K_0 clusters, let n_{0i}, $1 \leq i \leq K_0$, be the number of daily load curves contained in the cluster $C_{0i} \in C_0$, and we obtain $\sum_{i=1}^{K_0} n_{0i} = N_0$. Then, we can give the definition of the probabilities of load patterns.

Definition 3 (Probability of Load Pattern)**.** *The probability of a load pattern μ_{0i} denotes the percentage of the daily load curves represented by μ_{0i} in the whole daily load curve dataset X_0. Let $P_0 = \{p_{01}, p_{02}, \ldots, p_{0K_0}\}$ be the set of probabilities of load patterns A_0, then*

$$p_{0i} = \frac{n_{0i}}{N_0}, \tag{1}$$

where $\sum_{i=1}^{K_0} p_{0i} = 1$ and $1 \leq i \leq K_0$.

After obtaining a set of load patterns A_0 based on X_0, a new set of daily load curves $X_1 = \{x_{11}, x_{12}, \ldots, x_{1N_1}\} \in \mathbb{R}^{d \times N_0}$ comes due to the continuous electricity power consumption. We aim to obtain a set of updated load patterns $A_1 = \{\mu_{11}, \mu_{12}, \ldots, \mu_{1K_1}\} \in \mathbb{R}^{d \times K_1}$ based on the existing load patterns A_0 and the new daily load curves X_1. This means that we conduct an incremental clustering with X_1 and A_0 rather than an overall clustering with $[X_0, X_1]$.

As new sets of daily load curves continuously come, we can give a generalization of our incremental clustering problem. Let $A_{t-1} = \{\mu_{t-1,1}, \mu_{t-1,2}, \ldots, \mu_{t-1,K_{t-1}}\} \in \mathbb{R}^{d \times K_{t-1}}$ be the existing load patterns and $X_t = \{x_{t1}, x_{t2}, \ldots, x_{tN_t}\} \in \mathbb{R}^{d \times N_t}$ be the new set of daily load curves, we aim at proposing an incremental clustering algorithm that can obtain a set of updated load patterns $A_t = \{\mu_{t1}, \mu_{t2}, \ldots, \mu_{tK_t}\} \in \mathbb{R}^{d \times K_t}$, which equals or approximates to the load patterns extracted directly from overall daily load curves $X = \{X_0, X_1, \ldots, X_t\}$.

3.2. Load Pattern Extraction

Load pattern extraction is based on the clustering of daily load curves in this work. We adopt a fused load curve clustering algorithm called FCCWT [39] to extract the load patterns. The diagram of FCCWT is illustrated in Figure 2. This algorithm is designed specially for load clustering based on time-series electricity load data in our previous work. It conducts an indirect clustering, in which daily load curves are transformed into approximation signals and detail signals by a multi-level Harr wavelet before the load curve clustering for dimension reduction. Moreover, the approximation signals $X_{\alpha L}$ and detail signals $X_{\alpha H}$ are clustered separately and then fused to avoid information loss caused by the dimension reduction and improve the clustering performance. Although this algorithm is non-incremental, it provides a higher clustering validity comparing with other related methods.

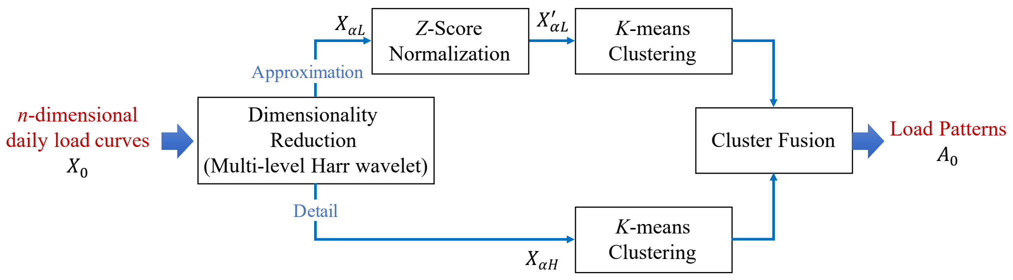


Figure 2. Diagram of non-incremental clustering algorithm FCCWT [39] for load pattern extraction.

4. Incremental Consumer Behavior Learning

In this section, we introduce the incremental clustering algorithm used for electricity consumption pattern learning. First, we present an overview of the incremental clustering algorithm. Second, we optimize this algorithm by a novel probability strategy in order to improve the incremental clustering performance. Third, several parameters are updated for the following continuous incremental clustering. Finally, we give the generalization of our optimized incremental clustering with the analysis of its asymptotic time complexity.

4.1. Incremental Clustering Algorithm

As presented in Section 3, the inputs are the existing load patterns A_0 and new daily load curves X_1, while the output is the updated load patterns A_1. The main challenge of our problem is how to determine whether to create a new load pattern. As consumer behaviors are complex, it is uncertain that there are any different load patterns in X_1 comparing with A_0. We cannot conduct a simple clustering by regarding all $\mu_{0i} \in A_0$ as the cluster centers. As a result, a novel incremental clustering algorithm is proposed to intergrade the load patterns of X_1 into A_0. This model is able to determine whether integrating a load pattern into a μ_{0i} or keeping it as a new load pattern. An illustration of the incremental clustering algorithm is presented in Figure 3, which contains three phases: load pattern extraction, load pattern intergradation, and load pattern modification. As the example shown in Figure 3, the set of existing load patterns A_0, which is extracted from X_0, contains two load patterns. Then, we extract five new load patterns from new daily load curves X_1 and intergrade them with the two existing load patterns one by one. For the intergration of $\mu_{a_1,1}$, we obtain an existing load pattern and an intergrated load pattern. After five times of load pattern intergradation and one extra load pattern modification, we finally obtain four updated load patterns.

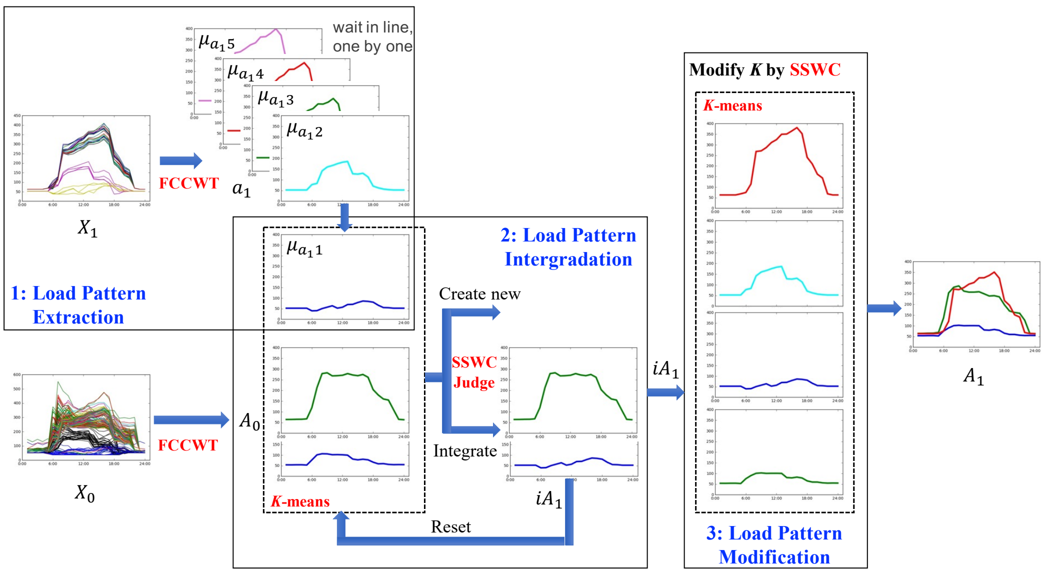


Figure 3. An illustration of the incremental clustering algorithm, including (1) load pattern extraction, (2) load pattern intergradation, and (3) load pattern modification. The inputs are the existing load patterns A_0 and new daily load curves X_1, and the output is the updated load patterns A_1.

Given a set of existing load patterns $A_0 = \{\mu_{01}, \mu_{02}, \ldots, \mu_{0K_0}\}$ and a set of new daily load curves $X_1 = \{x_{11}, x_{12}, \ldots, x_{1N_1}\}$, we describe these phases in detail.

Load pattern extraction. We need to process the new daily load curves $X_1 = \{x_{11}, x_{12}, \ldots, x_{1N_1}\}$ before the load pattern intergradation. A fused load curve clustering algorithm FCCWT [39], which is our previous work designed specially for daily load curve clustering, is applied to X_1. Then, we obtain the set of its load patterns $a_1 = \{\mu_{a_1 1}, \mu_{a_1 2}, \ldots, \mu_{a_1 K_{a_1}}\}$. The corresponding probability of a_1 is $P_{a_1} = \{p_{a_1 1}, p_{a_1 2}, \ldots, p_{a_1 K_{a_1}}\}$, where $p_{a_1 i} = n_{a_1 i}/N_1$ and $1 \le i \le K_{a1}$.

Load pattern intergradation. Let iA_1 denote the result of load pattern intergradation, and iA_1 is initialized as $iA_1 = A_0$. We combine the ith load pattern $\mu_{a_1 i} \in a_1$ with all load patterns in iA_1, which is denoted as $[iA_1, \mu_{a_1 i}]$. Then, two K-means clusterings are performed on $[iA_1, \mu_{a_1 i}]$ with $K = K_0$ and $K = K_0 + 1$, respectively. We evaluate their clustering results by the Simplified Silhouette Width Criterion (SSWC), which is one variant of Silhouette Width Criterion (SWC) index [39,49]. Then, the SSWC values of the clustering results when $K = K_0$ and $K = K_0 + 1$ are denotes as $\text{SSWC}_{K=K_0}$ and $\text{SSWC}_{K=K_0+1}$, respectively.

Given $[iA_1, \mu_{a_1 i}] = \{\mu_1, \mu_2, \ldots, \mu_{K_0}, \mu_{K_0+1}\}$ and its clustering result $C = \{c_1, c_2, \ldots, c_K\}$ with the set of corresponding cluster centers $\overline{A} = \{\overline{\mu}_1, \overline{\mu}_2, \ldots, \overline{\mu}_K\}$, the SSWC is calculated as the average of the Simplified Silhouette of the individual load pattern μ_j over $j = 1, 2, \ldots, K_0 + 1$.

$$\text{SSWC}_K = \frac{1}{K_0+1} \sum_{j=1}^{K_0+1} S_{\mu_j} = \frac{1}{K_0+1} \sum_{j=1}^{K_0+1} \frac{b_{c_r,\mu_j} - a_{c_r,\mu_j}}{\max\{a_{c_r,\mu_j}, b_{c_r,\mu_j}\}}, \tag{2}$$

where a_{c_r,μ_j} is the distance between μ_j and the center of cluster $c_r \in C$, while b_{c_r,μ_j} is the closest distance between μ_j and the centers of other clusters in C except for c_r. They are calculated as follows:

$$a_{c_r,\mu_j} = \text{dist}(\mu_j, \overline{\mu}_r), \tag{3}$$

$$b_{c_r,\mu_j} = \min\{\text{dist}(\mu_j, \overline{\mu}_w)\}, \tag{4}$$

where $1 \le r, w \le K$ and $r \ne w$. K refers to the parameter of clustering conducted on $[iA_1, \mu_{a_1 i}]$. Then, we obtain $\text{SSWC}_{K=K_0}$ and $\text{SSWC}_{K=K_0+1}$ according to Equations (2)–(4). There are two situations when comparing $\text{SSWC}_{K=K_0}$ and $\text{SSWC}_{K=K_0+1}$.

(1) $\text{SSWC}_{K=K_0} \ge \text{SSWC}_{K=K_0+1}$ implies that the clustering performance of $K = K_0$ is equal or superior to the performance of $K = K_0 + 1$. As a result, we do not keep the ith load pattern $\mu_{a_1 i}$ as a new load pattern, and adopt the set of cluster centers when $K = K_0$ as the integrating result of $[iA_1, \mu_{a_1 i}]$.

(2) $\text{SSWC}_{K=K_0} < \text{SSWC}_{K=K_0+1}$ implies that $K = K_0 + 1$ results in a better clustering performance than $K = K_0$ does. In that case, we keep the ith load pattern $\mu_{a_1 i}$ as a new load pattern, and adopt the set of cluster centers when $K = K_0 + 1$ as the integrating result of $[iA_1, \mu_{a_1 i}]$.

After the above comparison and judgment, the set iA_1 is reset with the integrating result of $[iA_1, \mu_{a_1 i}]$. Each $\mu_{a_1 i}$ over $i = 1, 2, \ldots, K_{a_1}$ with iA is integrated gradually according to this procedure. Finally, we obtain the intergraded set $iA_1 = \{i\mu_{11}, i\mu_{12}, \ldots, i\mu_{iK_{iA_1}}\}$.

Load pattern modification. We perform a further modification on the intergraded set iA_1 to obtain an optimal incremental clustering result. As the the number of load patterns generally is within the range $K \in [2, 10]$ [25,39], multiple K-means clusterings are applied to iA_1 with K in the range of 2 to $\min\{K_{iA_1}, 10\}$, where K_{iA_1} denotes the number of load patterns in iA_1. The SSWCs of $\min\{K_{iA_1}, 10\} - 1$ times clusterings are calculated and compared with each other. Then we select the K with the largest SSWC as the optimal parameter, and regard the set of cluster centers with the selected optimal K as our target set of updated load patterns A_1.

We outline the incremental clustering of A_0 and X_1 in Algorithm 1, including the three phase mentioned above. In Algorithm 1, Line 1 is for load pattern extraction, Lines 2–11 conduct load pattern intergradation, and Lines 12–16 are for load pattern modification.

Algorithm 1: The incremental clustering algorithm

```
   Input: a set of existing load patterns $A_0$, a set of new daily load curves $X_1$;
   Output: the set of updated load patterns $A_1$.
1  Apply FCCWT algorithm to $X_1$ to obtain the set of its load patterns
     $a_1 = \{\mu_{a_1 1}, \mu_{a_1 2}, \dots, \mu_{a_1 K_{a_1}}\}$;
2  Initialize $iA_1 = A_0$;
3  for each $\mu_{a_1 i}$ do
4      Combine $iA_1$ and $\mu_{a_1 i}$ as a set $[iA_1, \mu_{a_1 i}]$;
5      for $K = K_0, K_0 + 1$ do
6          Perform $K$-means clustering on $[iA_1, \mu_{a_1 i}]$;
7          Calculate the SSWC of the clustering result;
8      if $SSWC_{K=K_0} \geq SSWC_{K=K_0+1}$ then
9          Integrate $\mu_{a_1 i}$ into an existing load pattern by resetting $iA_1$ with the cluster
             centers of $K{=}K_0$;
10     else
11         Keep $\mu_{a_1 i}$ as an new load pattern by resetting $iA_1$ with the cluster centers
             of $K{=}K_0{+}1$;
12 for $K = 2, 3, \dots, \min\{K_{iA_1}, 10\}$ do
13     Perform $K$-means clustering on $iA_1$;
14     Calculate the SSWC of the clustering result;
15 Select the $K$ with the largest SSWC as $K_{opt}$;
16 Assign the cluster centers of $K$-means$(iA_1, K_{opt})$ to $A_1$;
17 return $A_1$.
```

4.2. Optimization via Probability Strategy

Assume $A'_1 = \{\mu'_{11}, \mu'_{12}, \dots, \mu'_{1K_1}\} \in \mathbb{R}^{d \times K'_1}$ is the load patterns extracted directly from the combined set $[X_0, X_1]$, then A'_1 is based on the non-incremental clustering of $N_0 + N_1$ daily load curves. On the other hand, the incremental clustering algorithm shown in Algorithm 1 is based on the fusion of load patterns from both A_0 and a_1, which refer to only $K_0 + K_{a_1}$ load patterns. Our purpose is to obtain an A_1 that equals or approximates to A'_1. However, the simply K-means clustering algorithm with Euclidean distance is not appropriate to achieve this purpose.

It should be considered that the load patterns usually have different probabilities so that we should not treat them equally in the incremental clustering. Thus, an optimized distance measure with probability strategy is proposed for Algorithm 1, in which Euclidean distance measure is replaced with the proposed measure when performing both K-means clustering and SSWC calculation shown in Equation (3). It is assumed that this probability strategy can optimize Algorithm 1 to achieve an ideal A_1. Given a set of load patterns $A = \{\mu_1, \mu_2, \dots, \mu_K\}$ with the set of corresponding probability $P = \{p_1, p_2, \dots, p_K\}$, where $p_i = n_i/N$ and N is the number of daily load curves that A refers to, the optimized distance with probability strategy between μ_i and μ_j is calculated as follows:

$$\text{dist}_p(\mu_i, \mu_j) = p_i N p_j N ||\mu_i - \mu_j||_2 = n_i n_j ||\mu_i - \mu_j||_2, \tag{5}$$

where n_i and n_j denote the numbers of daily load curves that μ_i and μ_j represent, respectively.

The cluster center $\overline{\mu}_r$ in K-means clustering with Euclidean distance is calculated as the mean of the objects that contained in the cluster:

$$\overline{\mu}_r = \frac{1}{m_r} \sum_{\mu \in C_r} \mu_r, \tag{6}$$

where m_r is the number of μ contained in the cluster C_r. We set the probability of $\overline{\mu}_r$ with $p_{\overline{\mu}_r} = 1/N$ when performing K-means clustering with the optimized distance. As a result, the optimized distance with probability strategy between μ_i and $\overline{\mu}_r$ is calculated as follows:

$$\text{dist}_p(\mu_i, \overline{\mu}_r) = p_i N p_{\overline{\mu}_r} N ||\mu_i - \overline{\mu}_r||_2 = n_i ||\mu_i - \overline{\mu}_r||_2. \tag{7}$$

Similarly, the calculation of cluster center shown in Equation (6) should be rewritten as

$$\overline{\mu}_r = \frac{1}{\sum_{\mu \in C_r} p_r N} \sum_{\mu \in C_r} p_r N \mu_r = \frac{1}{\sum_{\mu \in C_r} n_r} \sum_{\mu \in C_r} n_r \mu_r, \tag{8}$$

where n_r denotes the number of daily load curves that the load pattern μ_r refers to, and $\sum_{\mu \in C_r} n_r$ denotes the total number of daily load curves that all $\mu \in C_r$ refer to.

4.3. Updating Parameters

As new daily load data continuously grow with the electricity power consumption of consumers, we should update several essential parameters after one incremental clustering for the preparation of the next incremental clustering. The sets X_0 and X_1 contain N_0 and N_1 daily load curves, respectively. Their combined set $[X_0, X_1]$ contains $N_0 + N_1$ daily load curves totally. The incremental clustering on $[X_0, X_1]$ gives the set of updated load patterns A_1 and the set of its corresponding clustering result $C_1 = \{C_{11}, C_{12}, \ldots, C_{1K_1}\}$. Let $P_1 = \{p_{11}, p_{12}, \ldots, p_{1K_1}\}$ be the set of corresponding probabilities of A_1, the probability of $\mu_{1r} \in A_1$ for the rth cluster C_{1r} is updated as

$$p_{1r} = \frac{n_{1r}}{N_0 + N_1}, \tag{9}$$

where n_{1r} is the number of daily load curves that the load pattern μ_{1r} represents. We update n_{1r} as follows:

$$n_{1r} = \sum_{\mu_0 \in C_{1r}} n_{0r} + \sum_{\mu_{a_1} \in C_{1r}} n_{a_1 r}, \tag{10}$$

where $\sum_{\mu_0 \in C_{1r}} n_{0r}$ denotes the total number of daily load curves that all $\mu_{0i} \in A_0$ belonging to $C_1 r$ represent, and $\sum_{\mu_{a_1} \in C_{1r}} n_{a_1 r}$ denotes the one that all $\mu_{a_1 i} \in a_1$ belonging to $C_1 r$ represent. After the updating of P_1, A_1 is ready to be conducted in another incremental clustering with the next coming data set X_2.

4.4. Generalization of Incremental Clustering

In practice, there are continuous coming new daily load data sets $X_1, X_2, \ldots, X_t$. The generalization of incremental clustering algorithm, which is based on the existing load patterns A_0 and new daily load curves $X = \{X_1, X_2 \ldots, X_t\}$, is outlined in Algorithm 2. For n_{sr} and p_{sr} in the generalized algorithm, the updating equations shown in Equation (9) and Equation (10) become

$$p_{sr} = \frac{n_{sr}}{\sum_{i=0}^{s} N_i}, \tag{11}$$

$$n_{sr} = \sum_{\mu_{s-1} \in C_{sr}} n_{s-1,r} + \sum_{\mu_{a_s} \in C_{sr}} n_{a_s r}, \tag{12}$$

where N_i is the number of daily load curves that X_i contains, and μ_{s-1} and μ_{a_s} are the load patterns that belong to A_{s-1} and a_s, respectively.

In Algorithm 2, the incremental clustering is continuously performed with the coming of X_s. This means that it is performed immediately once X_s comes without waiting all $X_{s+1}, X_{s+2}, \ldots, X_t$ come. Therefore, we can obtain the updated load patterns A_s in time, and then the algorithm is paused until X_{s+1} comes.

```
Algorithm 2: The generalization of Algorithm 1
  Input: a set of existing load patterns $A_0$ referring to $N_0$ daily load curves; the set
         of probabilities $P_0$; $t$ set of new daily load curves $X_1, X_2 \dots, X_t$, each $X_s$
         contains $N_s$ daily load curves;
  Output: the updated load patterns $A_1, A_2, \dots, A_t$.
1 for $X_s$ over $s = 1, 2, \dots, t$ do
2     Perform Algorithm 1 with the probability strategy based on $A_{s-1}$ and $X_s$ to
       obtain $A_s$;
3     Update $n_{sr}$ for each $\mu_{sr}$ in $A_s$ by Equation (12);
4     Update $p_{sr}$ for each $\mu_{sr}$ in $A_s$ by Equation (11);
5     return $A_s$.
```

4.5. Complexity Analysis

The time complexity of FCCWT is $O(NKT)$ where N is the number of daily load curves, K is the number of clusters and T is the number of iterations needed until convergence [39]. The time complexity of K-means is $O(NdKT)$ while the one of SSWC calculation is $O(NdK)$, where d is the size of dimensions of daily load curves. As the default of maximum T is usually set as 100, 200, or 300, we assume that all Ts in Algorithm 1 are the same so that the time complexity can be analyzed more easily. Moreover, we also assume that all Ks adopt the maximum value 10 due to $K \in [2, 10]$.

Based on the above assumptions, the asymptotic time complexities of load pattern extraction, load pattern intergradation and load pattern modification in Algorithm 1 are $O(KN_1T)$, $O((2K^3 + 3K^2 + K)d(T+1))$ and $O(Kd(T+1)\sum_{k=2}^{K} k)$, respectively. Therefore, the asymptotic time complexity of Algorithm 1 is

$$\begin{aligned} &O\Big(KN_1T + (2K^3 + 3K^2 + K)d(T+1) + Kd(T+1)\sum_{k=2}^{K} k\Big) \\ &= O\Big(KN_1T + (2K^3 + 3K^2 + K\sum_{k=1}^{K} k)d(T+1)\Big). \end{aligned} \tag{13}$$

The time complexity of updating parameters is $O(K)$ so that the asymptotic time complexity of Algorithm 2 is

$$O\Big((KT\sum_{s=1}^{t} N_s + (2K^3 + 3K^2 + K\sum_{k=1}^{K} k)td(T+1) + Kt)\Big), \tag{14}$$

where $O(KT\sum_{s=1}^{t} N_s)$ is the time complexity of t times FCCWT performed on X_s, $O((2K^3 + 3K^2 + K\sum_{k=1}^{K} k)td(T+1))$ is the time complexity of t times load pattern intergradation and modification, and $O(Kt)$ is the time complexity of t times parameter updating.

As for non-incremental clustering, the time complexity of t times FCCWT on $[X_0, X_1, \cdot, X_s]$ over $s = 1, 2, \cdots, t$ is

$$\begin{aligned} &O\Big((N_0 + \sum_{s=0}^{1} N_s + \sum_{s=0}^{2} N_s + \cdots + \sum_{s=0}^{t} N_s)KT\Big) \\ &= O\Big(((t+1)N_0 + tN_1 + (t-1)N_2 + \cdots + N_t)KT\Big), \end{aligned} \tag{15}$$

which is sensitive to the size of t. Similarly, the time complexity of t times non-incremental clustering algorithm K-means on the same data is $O(((t+1)N_0 + tN_1 + (t-1)N_2 + \cdots + N_t)dKT)$. Comparing Equation (15) with the time complexities of two non-incremental clustering algorithms, it is suggested that the incremental clustering saves time and reduces the clustering scale when t is relatively large.

5. Experimental Settings

This section presents the experimental settings including datasets, evaluation criterion, and comparison methods in details. In evaluation criterion, an weighted mean error measure is proposed to evaluate the accuracy of the load patterns extracted by incremental clustering.

5.1. Datasets

The dataset used in the experiment refers to 14,976 commercial and 2800 residential electricity consumers in 936 counties of United States (Available online: https://openei.org/datasets/files/961/pub/ (accessed on 24 June 2019). Eight of 2808 residential consumers have missing data so that only the data of 2800 residential electricity consumers are used in the experiment). It contains 24-value daily load data over one year and records the electricity power consumption at every 1 h from 1:00 to 24:00 per day. As the proposed algorithm is designed for learning the electricity consumption patterns of a single consumer, the data of one electricity consumer can be regarded as a sub-dataset that leads to a sub-experiment. As a result, we conduct 17,776 sub-experiments totally. Moreover, three situations are considered for every sub experiment. We select 3 months, 6 months, and 9 months daily load data as the initial set X_0, respectively. The remaining data are divided by month and then regarded as $X_1, X_2, \cdots, X_t$. For example, in the case of $t = 3$, daily load data from January to September are selected as X_0, and the data of October, November, and December are regarded as X_1, X_2, and X_3, respectively.

5.2. Evaluation Criterion

We employ two types of measures including clustering validity indices and accuracy measures as the evaluation criterion in the experiment. Moreover, we propose an weighted mean minimum error measure for the accuracy measures.

Clustering validity indices. The clustering performance of the proposed method is also evaluated by diverse clustering validity indices including Davies–Bouldin index (DB), Dunn validity index (DVI), and SWC.

Let $C_s = \{C_{s1}, C_{s2}, \cdots, C_{sK_s}\}$ be the corresponding clustering results of A_s, the clustering validity indices of C_s follow the equations below:

$$\mathrm{DB}(C_s) = \frac{1}{K_s}\sum_{r=1}^{K_s}\max_{w\neq r}\left\{\frac{\overline{C}_{sr} + \overline{C}_{sw}}{||\mu_{sr} - \mu_{sw}||}\right\}, \tag{16}$$

$$\mathrm{DVI}(C_s) = \frac{\min\limits_{0<r\neq w<K_s}\left\{\min\limits_{\forall x_i\in C_{sr},\forall x_j\in C_{sw}}\left\{||x_i - x_j||_2\right\}\right\}}{\max\limits_{0<r\leq K_s}\max\limits_{\forall x_i,x_j\in C_{sr}}\left\{||x_i - x_j||_2\right\}}, \tag{17}$$

$$\mathrm{SWC} = \frac{1}{N}\sum_{j=1}^{N}\frac{b_{C_{sr},x_j} - a_{C_{sr},x_j}}{\max\{a_{C_{sr},x_j}, b_{C_{sr},x_j}\}}, \tag{18}$$

where $\overline{C}_{sr}$ and $\overline{C}_{sw}$ is the average within-group distance for C_{sr} and C_{sw}, respectively; x_i and x_j denote two daily load curves contained in $[X_0, X_1, \ldots, X_s]$, respectively; $N = \sum_{i=0}^{s} N_i$, a_{C_{sr},x_j} denotes the mean distance of x_j to all other daily load curves in C_{sr}; and b_{C_{sr},x_j} denotes the minimum mean distance of x_j to all daily load curves in C_{sw}, $w \neq r$.

Accuracy measures. As we aim to obtain $A_s = \{\mu_{s1}, \mu_{s2}, \cdots, \mu_{sK_s}\}$ that equals or approximates to $A'_s = \{\mu'_{s1}, \mu'_{s2}, \cdots, \mu'_{sK'_s}\}$, which is the load patterns extracted directly from $[X_0, X_1, \cdots, X_s]$, we employ the accuracy measures for time-series forecasting to evaluate the load patterns in A_s comparing with those in A'_s. Both scale-dependent and percentage-based measures are employed, including Normalized Root Mean Square Error (NRMSE), Mean Absolute Error (MAE), and Symmetric Mean Absolute Percentage Error (sMAPE) [50,51]. However, both A_s and A'_s contain several load patterns so that we propose a weighted mean minimum error based on the numbers of load patterns in A_s and A'_s.

$$\mathrm{NRMSE}(\mu'_{si}, \mu_{sj}) = \sqrt{\frac{1}{d}\sum_{l=1}^{d}\left(\frac{\mu'_{si,l} - \mu_{sj,l}}{\mu'_{si,l}}\right)^2}, \tag{19}$$

$$\mathrm{MAE}(\mu'_{si}, \mu_{sj}) = \frac{1}{d}\sum_{l=1}^{d}|\mu'_{si,l} - \mu_{sj,l}|, \tag{20}$$

$$\mathrm{sMAPE}(\mu'_{si}, \mu_{sj}) = \frac{1}{d}\sum_{l=1}^{d}\frac{2 \cdot |\mu'_{si,l} - \mu_{sj,l}|}{|\mu'_{si,l}| + |\mu_{sj,l}|)}. \tag{21}$$

(1) $K'_s \leq K_s$ indicates that the incremental clustering may cause extra load patterns. We calculate the minimum error for each $\mu_{sj} \in A_s$, which is the error between μ_{sj} and its most similar load pattern $\mu'_{si} \in A'_s$. Moreover, we weight the mean error by K_s/K'_s due to the extra load patterns.

$$E(A'_s, A_s) = \frac{K_s}{K'_s}\frac{1}{K'_s}\sum_{j=1}^{K'}\min_{\mu_{sj}\in A_s}\{\mathrm{meas}(\mu'_{si}, \mu_{sj})\}, \tag{22}$$

where $\mathrm{meas}(\mu'_{si}, \mu_{sj})$ can be NRMSE, MAE, or sMAPE shown in Equation (19)–(21).

(2) $K'_s > K_s$ indicates that the incremental clustering misses some load patterns. We calculate the minimum error for each $\mu'_{si} \in A'_s$, which is the error between μ'_{si} and its most similar load pattern $\mu_{sj} \in A_s$. Similarly, we weight the mean error by K'_s/K_s due to the missing load patterns.

$$E(A'_s, A_s) = \frac{K'_s}{K_s}\frac{1}{K_s}\sum_{i=1}^{K}\min_{\mu'_{si}\in A'_s}\{\mathrm{meas}(\mu'_{si}, \mu_{sj})\}. \tag{23}$$

According to the definitions of these indices and measures, smaller Errors indicate higher accuracy and smaller DB indicates better clustering performance. On the contrary, the larger the DVI and SWC are the better the clustering performance is.

5.3. Comparison Methods

We adopt two algorithms FCCWT and K-means to conduct non-incremental clustering on $[X_0, X_1, \cdots, X_s]$ over $s = 1, 2, \cdots, t$, and then regard the load patterns with the optimal clustering performance as the baseline for evaluating the accuracy of other incremental clustering methods. Moreover, we also compare the clustering performance of the non-incremental clustering algorithms with our proposed method and other related incremental clustering methods. Methods compared in the experiments are summarized in Table 3.

Table 3. Summary of comparison methods.

Method	Description	Incremental	Probability Strategy (PS)
FCCWT	The method designed for daily load curve clustering [39]	no	no
K-means	The common K-means algorithm	no	no
ICluster-PS	The proposed method designed for daily load curve clustering	yes	yes
ICluster	The proposed method without PS	yes	no
IK-means-PS	The incremental method that adopts K-means with PS	yes	yes
IK-means	The incremental method that adopts K-means without PS	yes	no
HPStream	The algorithm for high-dimensional data streams [29]	yes	no

6. Results and Evaluation

In this section, we first present and discuss the general incremental clustering performance and accuracy of comparison methods on data of all consumers. Then, a commercial consumer is randomly selected as a case for electricity consumption behavior patterns analysis. We also compare the mean runtime of incremental and non-incremental clustering algorithms to support the time complexity analysis in the former section. Furthermore, we conduct pattern evolution analysis based on the incremental clustering results of another randomly selected residential consumer.

6.1. Incremental Clustering Performance

We conduct the experiments of Algorithm 2 with $t = 3$, $t = 6$, and $t = 9$, which means that three, six, and nine incremental clustering processes shown in Algorithm 1 are performed in one experiment, respectively. Both clustering performance and accuracy of the methods are compared for the incremental clustering performance. Although there are various types of consumers in the dataset, we still use the mean performance of all consumers to evaluate the comparison methods because most of the evaluation criteria are percentage-based.

Table 4 shows the mean clustering performance comparison of the methods on the data of 17,776 electricity consumers. The former two methods are non-incremental clustering methods while the later five methods are incremental. We first compare the mean clustering performance of incremental methods. According to the definitions of three clustering validity indices shown in Equation (16)–(18), the larger the DVI and SWC are the better the clustering performance is, while a smaller DB indicates better clustering performance. The optimal results of incremental methods are displayed in bold. The proposed method ICluster-PS shows the smallest DB values and largest SWC values in Table 4. Although the DVI values of ICluster-PS are slightly lower than the ones of HPStream when $t = 3$ and $t = 6$, the average clustering performance of ICluster-PS is optimal in all compared incremental clustering methods. Therefore, these results indicate that the proposed method ICluster-PS outperforms other incremental methods. The largest improvement of clustering performance comparing with other incremental clustering methods is 44.2%. On the other hand, ICluster-PS still requires improvement due to its lower clustering performance compared with the non-incremental clustering FCCWT and *K*-means that conduct clustering directly on overall daily load curves.

Table 4. Mean clustering performance comparison of the methods.

Method	***t* = 3**			***t* = 6**			***t* = 9**		
	DB$^-$	**DVI**$^+$	**SWC**$^+$	**DB**$^-$	**DVI**$^+$	**SWC**$^+$	**DB**$^-$	**DVI**$^+$	**SWC**$^+$
FCCWT *	0.9033	0.1267	0.5846	0.9066	0.1316	0.5854	0.8932	0.2056	0.5985
K-means *	1.0333	0.1487	0.4778	1.0283	0.1519	0.4802	1.0058	0.2240	0.5110
ICluster-PS	**1.3159**	0.0624	**0.3350**	**1.2556**	0.0707	**0.3856**	**1.1599**	**0.1322**	**0.4199**
ICluster	1.3369	0.0596	0.3212	1.2821	0.0672	0.3457	1.1924	0.1317	0.3823
I*K*-means-PS	1.8056	0.0403	0.1129	1.6337	0.0576	0.2776	1.4048	0.0899	0.3655
I*K*-means	1.7872	0.0422	0.1193	1.6177	0.0578	0.2704	1.4390	0.0917	0.3435
HPStream	2.2739	**0.0740**	0.3314	1.9913	**0.0781**	0.3478	1.9131	0.0887	0.3506
Improvement	26.4%	−15.7%	1.1%	22.4%	−9.5%	10.9%	19.4%	44.2%	19.8%

*: non-incremental method; $^-$: the minimum is the optimal; $^+$: the maximum is the optimal.

As FCCWT presents the optimal clustering performance in Table 4, we decide to adopt the load patterns obtained from FCCWT as the baseline for accuracy measure. Then, we can calculate the mean errors of the five incremental methods based on Equation (22) and Equation (23) using three different accuracy measures. The results, which are shown in Table 5, indicate that ICluster-PS has the optimal performance as the minimum error denotes the highest accuracy. The improvement of accuracy is between 29.8% and 66.0% comparing with other incremental clustering methods.

Table 5. Mean error comparison of the methods.

Method	***t* = 3**			***t* = 6**			***t* = 9**		
	NRMSE	**MAE**	**sMAPE**	**NRMSE**	**MAE**	**sMAPE**	**NRMSE**	**MAE**	**sMAPE**
ICluster-PS	**0.0436**	**10.0744**	**0.0332**	**0.1008**	**20.3269**	**0.0784**	**0.1363**	**32.2105**	**0.1059**
ICluster	0.1172	28.7795	0.0907	0.1585	32.2034	0.1239	0.1820	40.9046	0.1424
I*K*-means-PS	0.0777	17.4945	0.0608	0.1520	29.9669	0.1210	0.1870	40.0440	0.1470
I*K*-means	0.1161	29.6179	0.0910	0.1939	38.5883	0.1533	0.2204	45.8908	0.1728
HPStream	0.2509	61.8134	0.1888	0.2489	58.7053	0.1942	0.2654	63.7557	0.2082
Improvement	62.4%	66.0%	63.5%	48.0%	47.3%	48.9%	38.2%	29.8%	38.7%

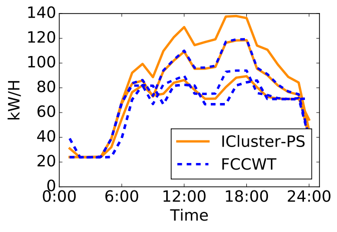
According to the results shown in Tables 4 and 5, the better clustering performance and smaller errors of methods with probability strategy compared with those without the strategy prove the optimization of our proposed probability strategy. Moreover, the incremental clustering algorithm of ICluster-PS, especially load pattern intergradation and modification, improves both clustering performance and the accuracy of *K*-means based on the comparisons between ICluster and *K*-means with or without probability strategy. As for the three groups of mean errors with different t, it is noticed that the mean errors increase with the rise of t, which means that the errors may increasingly rise over the continuous incremental clustering. However, the three groups of the mean clustering performance present an opposite tendency. Therefore, it can be only suggested that the load patterns updated by incremental clustering may tend to deviate from the load patterns obtained by FCCWT over time.

In summary, the proposed incremental clustering algorithm, ICluster-PS, can achieve an acceptable accuracy with mean error less than 10% and an improved clustering validity via its designed model and probability strategy. This result indicates that we can provide an efficient response when consumers require consumption analysis via smart meter or other facilities with limited resource. Although our experiments set the data of one month as X_s, it can be set optionally by consumers in practical application.

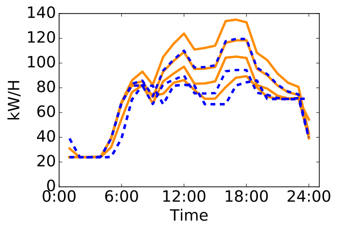
6.2. Case Analysis

A random electricity consumer is selected to be analyzed in detail for a further discussion of the proposed method and electricity consumer behaviors. The selected consumer is a full service restaurant, which have three typical load patterns based on the overall daily load curves. Figure 4 illustrates the load patterns obtained by ICluster-SP and FCCWT in the experiment when $t = 6$. Each subfigure presents both the incremental and non-incremental cluster centers of the data $[X_0, X_1, \cdots, X_s]$, where $1 \leq s \leq t$. The load patterns in solid line style denote the incremental cluster centers of ICluster-SP, while those in dashed line style denote the non-incremental cluster centers of FCCWT.

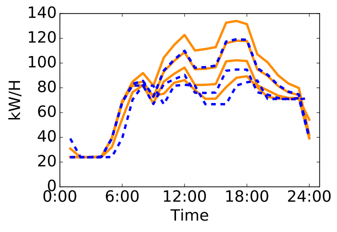
According to the clustering performance shown in Table 4, the load patterns of FCCWT are regarded as the accurate results. Note that these accurate load patterns are relatively stable and there is no distinct electricity consumption behavior drift happening to this consumer from July to December. The three typical load patterns of this consumer are distinct in terms of power degrees, starting time of the increase in the morning and ending time at night. The possible reasons for these distinctions are daylight saving time and seasonal influence. As for the incremental clustering results, their load patterns drift once on August shown in Figure 4b. Therefore, we can find out three typical load patterns in Figure 4a and four typical load patterns in other subfigures. These updated load patterns show similar patterns as the accurate ones if the power degrees of them are not taken into account. However, the distinct starting time of the increase in the morning shown by the accurate ones are not revealed by those of ICluster-PS until December, shown in Figure 4f.



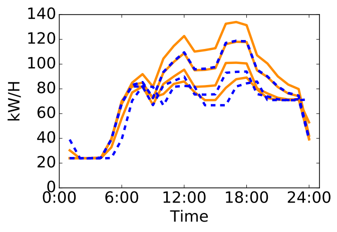
(**a**) January–July



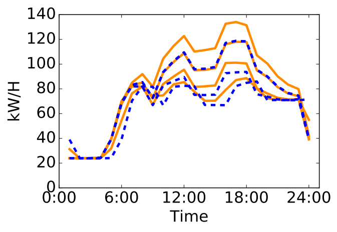
(**b**) January–August



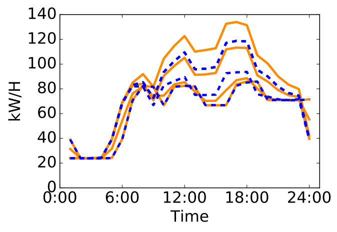
(**c**) January–September



(**d**) January–October



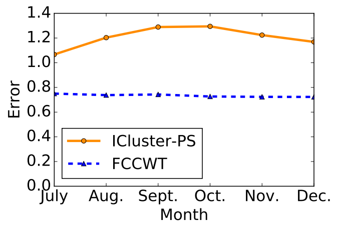
(**e**) January–November



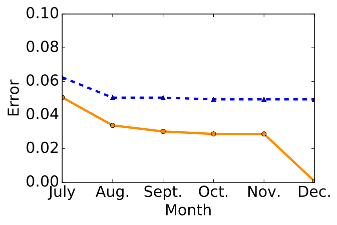
(**f**) January–December

Figure 4. Load pattern comparison between ICluster-PS and FCCWT of one randomly selected electricity consumer when $t = 6$. (**a**–**f**) Curves presenting the load patterns based on incremental or non-incremental clustering of $[X_0, X_1, \cdots, X_s]$ over $s = 1, 2, \cdots, t$.

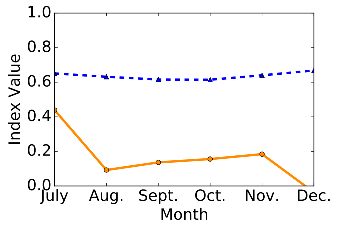
In addition, we evaluate the load patterns by the same accuracy measures and clustering validity indices used in the former evaluation, the results of which are illustrated in Figure 5. Each curve contains six values which refer to the evaluation of load patterns in Figure 4a–f, respectively. Figure 5a–c presents the clustering performance of both ICluster-PS and FCCWT. FCCWT shows a relatively stable clustering performance while ICluster-PS shows slight fluctuation. The optimal clustering performance, especially for DVI and SWC, of ICluster-PS is presented in July. On the other hand, Figure 5d–f denotes the accuracy measures of ICluster-PS comparing with FCCWT so that there is only one curve in each subfigure. All three curves show an increase at first and then decrease after August. Different from the presentation of its clustering performance, their optimal accuracies are shown in December, which are in accord with the results shown in Figure 4.



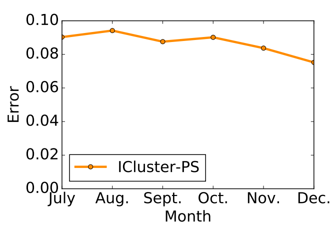
(**a**) DB



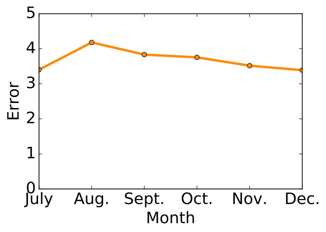
(**b**) DVI



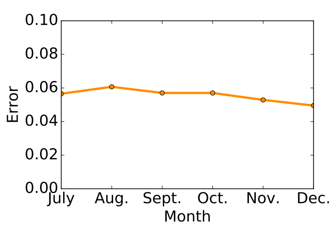
(**c**) SWC



(**d**) NRMSE



(**e**) MAE



(**f**) sMAPE

Figure 5. Errors and clustering validity indices comparisons between ICluster-PS and FCCWT of one randomly selected electricity consumer when $t = 6$. Each curve contains six values referring to the evaluation of load patterns in Figure 4a–f, respectively.

Based on the observation of this case, ICluster-PS can achieve incremental clustering for load pattern updating, although it may provide an slightly unstable performance in terms of accuracy and clustering validity. This result is acceptable for providing efficient and effective updated electricity consumption patterns with time and space constraints.

6.3. Runtime Comparison

Apart from the time complexity analysis of both incremental and non-incremental clustering algorithms in Section 4, we also compare their runtime in the experiment to support this analysis. The algorithms, which are written in Python and run on 64-bit Windows 10 operating system with Intel Core i5-5300U CPU and 8 GB RAM, are performed on the data of 16 commercial consumers in a same randomly selected county. Figure 6 shows the mean runtime comparison of the methods when $t = 9$. The comparison methods include the proposed incremental clustering algorithm ICluster-PS, and two non-incremental algorithms, FCCWT and K-means. Each algorithm is run 100 times in every clustering, which means that we run $16 \times 9 \times 100 \times 3$ times non-incremental or incremental clustering algorithms totally. According to Figure 6, it can be noticed that the runtime of ICluster-PS is stable and around 0.3 s while the the runtime of other two non-incremental clusterings increase with the rise of t. This result proves the time complexity analysis in Section 4, which is that the incremental clustering saves time when t is relatively large because it reduces the clustering scale. The runtime curve of ICluster-PS shows some slight fluctuations, which are caused by the small differences of the data in every month.

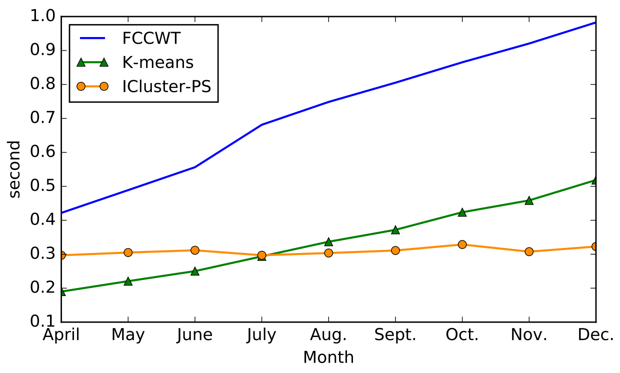
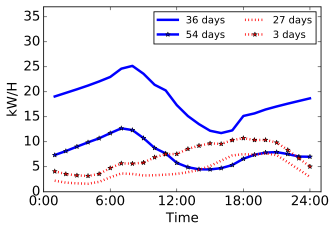


Figure 6. Mean runtime comparison of three methods on the data of 16 commercial consumers in 100 time experiments when $t = 9$ (nine clusterings from April to December).

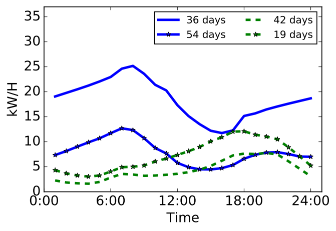
6.4. Pattern Evolution Analysis

We assume that our incremental clustering algorithm can be used to investigate the electricity consumption pattern evolution over time when conducting load pattern updates. As a result, we randomly select a residential consumer, who may have less stable consumption patterns than a commercial consumer, as a case for pattern evolution analysis. Figure 7 shows the updated load patterns of the selected residential consumer from April to December, which means that $t = 9$ is set in the experiment of this case analysis. Each subfigure denotes the load patterns of one incremental clustering with one month adding new data based on the load patterns of previous months. For example, Figure 7a indicates the load patterns updated by the first incremental clustering based on the existed load patterns of January to March and new daily load data of April, Figure 7b indicates the load patterns updated by the second incremental clustering based on the load patterns shown in Figure 7a and new daily load data of May, etc. In Figure 7, we use curves with different colors, line styles, and markers to distinguish various types or meanings of updated load patterns. The curves in blue and solid line style denote the load patterns that exist in last month, which means that these load patterns are not affected by new adding data and do not drift in current month. The curves in green and dashed line style denote the load patterns that are updated by new adding data in current month and have drifts comparing with the ones in last month. The curves in red and point line style denote the load patterns which are completely new and only refer to the days in current month. Markers on curves are only used to label different load patterns.

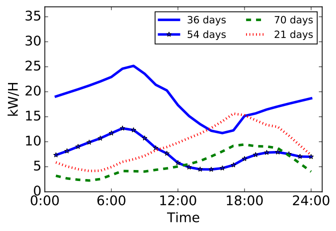
Moreover, we draw another figure, Figure 8, to illustrate the pattern evolution of the case shown in Figure 7. In Figure 8, each circle with a number denotes a cluster or load pattern, and the number inside the circle denotes the number of days that the load pattern refers to. There are three types of circles, which represent existed load patterns, updated load patterns and new load patterns, respectively. The plus and number shown on an arrow denote the number of new days added to the load pattern after one incremental clustering. In fact, Figure 8 is in accordance with Figure 7. The first column in left of Figure 8 indicates two load patterns extracted by non-incremental clustering with load data from January to March. Other nine columns, each of which denotes four load patterns updated by an incremental clustering with adding new load data in current month and the load patterns in last month (shown in left column), are corresponding to Figure 7a–i, respectively.



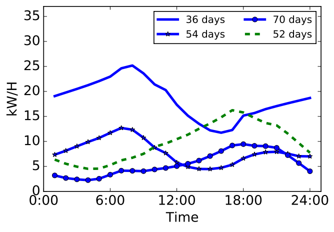
(**a**) January–April



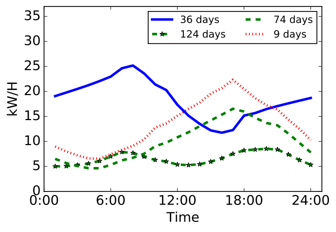
(**b**) January–May



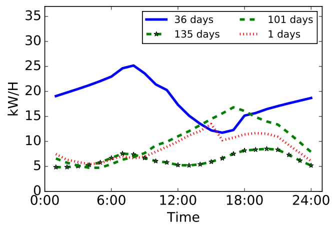
(**c**) January–June



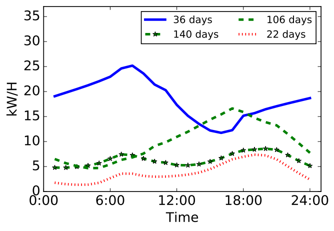
(**d**) January–July



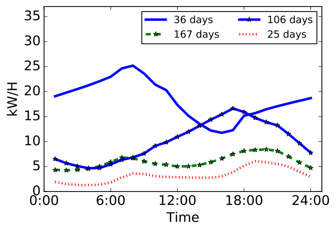
(**e**) January–August



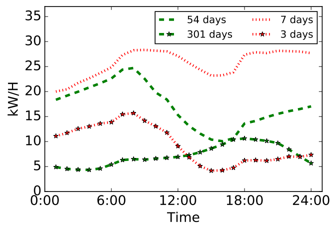
(**f**) January–September



(**g**) January–October



(**h**) January–November



(**i**) January–December

Figure 7. Updated load patterns of a residential consumer from April to December ($t = 9$). Each subfigure shows the load patterns updated by one incremental clustering with one month adding new data based on the load patterns of previous months. Load patterns in blue and solid line style denote existed patterns, those in green and dashed line style denote updated patterns, and others in red and point line style denote new patterns.

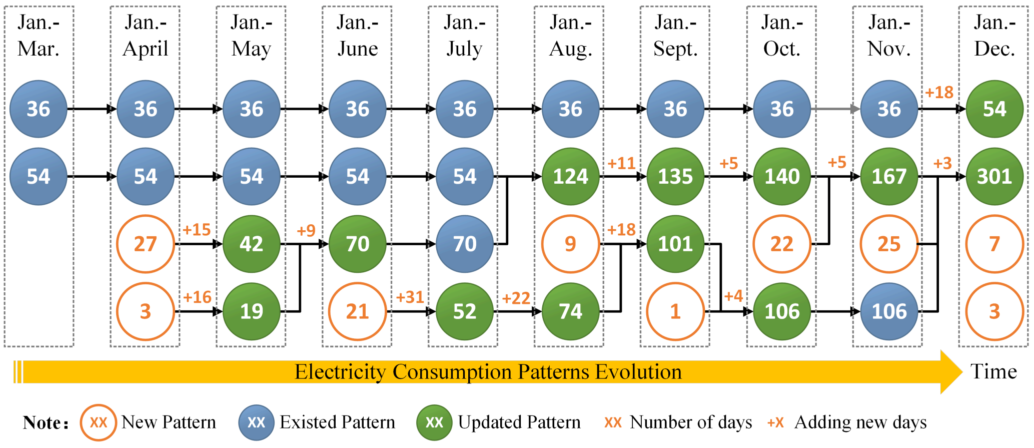

Jan.- Mar.
Jan.- April
Jan.- May
Jan.- June
Jan.- July
Jan.- Aug.
Jan.- Sept.
Jan.- Oct.
Jan.- Nov.
Jan.- Dec.
Electricity Consumption Patterns Evolution
Time
Note: XX New Pattern XX Existed Pattern XX Updated Pattern XX Number of days +X Adding new days

Figure 8. Electricity consumption patterns evolution of the case showed in Figure 7. The first column (left) denotes the load patterns extracted by non-incremental clustering with data from January to March. Other columns, each of which denotes the load patterns updated by incremental clustering based on one month adding new data and the load patterns shown in left column, are corresponding to Figure 7a–i, respectively.

The electricity consumption pattern evolution of the residential consumer is presented clearly according to Figures 7 and 8. The residential consumer has two load patterns, which refer to 36 and 54 days, in the first three months of the year based on a non-incremental clustering with the data from January to March. Note that the load pattern with 36 days is unchanged until December. There are 18 new days in December that have similar shape with this load pattern so that they are added in this pattern and the number of days included in this pattern becomes 54. Then, it can be found that this load pattern drifts slightly based on the comparison of the curve with 36 days shown in Figure 7h and the curve with 54 days shown in Figure 7i. Another load pattern with 54 days at first is unchanged until August. Then, it is continually updated and merged with new days or other existed load patterns, and finally becomes a load pattern with 301 days (Jan–Dec), which is presented by the green dashed curve with star markers shown in Figure 7i. In total, nine new load patterns emerge in April, June, August, September, October, November, and December. Most of them are merged with other load patterns in next month. For instance, a new load pattern with nine days emerges in August, then it is merged with 18 new days in September and the other load pattern with 74 days (Jan–Aug), and finally becomes an updated load pattern that refers to 101 days (Jan–Sept). We note that some load patterns are merged after one or several incremental clustering. Why do different load patterns become one after one or a few months? The reason is that the increase in the number of data samples leads to the change of the optimal clustering results.

Based on the pattern evolution analysis and the further analysis on the dates of all days included in every load pattern, we can find out when and how this residential consumer drifts electricity consumption behaviors. In that case, this consumer can have a clear understand of her/his electricity demand and make an effective response to it. On the other hand, electricity suppliers or other agencies can also detect any anomalies once electricity consumers, especially commercial or industrial consumers, drift their consumption patterns significantly.

7. Conclusions and Future Work

This paper aims to achieve efficient demand response and consumer segmentation for both electricity end consumers and suppliers by incremental consumer behavior learning. It supposes that an effective incremental clustering algorithm would constantly updated load pattern data for electricity consumers with limited resource. Moreover, the incremental clustering algorithm should reduce the training scale and save time comparing with non-incremental clustering algorithms.

Therefore, we propose an incremental clustering algorithm with probability strategy, ICluster-PS, for updating load patterns based on smart meter data. We also provide parameter updating and algorithm generalization to ICluster-PS in order to continuously perform our algorithm with new coming data. The proposed algorithm is evaluated on real-world data. The experimental results prove the accuracy and validity of our incremental clustering algorithm, especially load pattern intergradation, modification, and probability strategy. It has less time complexity and runtime than non-incremental clustering algorithm. On the other hand, although ICluster-PS cannot provide load patterns that are the same as those extracted directly from the overall electricity load data, it achieves acceptable updated results when saving time, reducing the clustering scale and even making full use of the historical information. It also outperforms other related incremental algorithms or data stream clustering algorithms.

Moreover, we conduct additional case study of pattern evolution analysis by using our proposed algorithm. The analysis results indicate that our algorithm is able to detect load pattern drifts through its updated load patterns. In the future work, we plan to improve the performance of the incremental clustering algorithm and employ incremental consumer behavior learning for automatic and real-time load pattern evolution analysis and detection.

Author Contributions: Conceptualization and methodology, Z.J. and R.L.; software, Z.J.; validation, Z.J.; writing—original draft preparation, Z.J.; writing—review and editing, R.L., F.Y.; supervision, F.Y.; project administration, R.L. and F.Y.; funding acquisition, Z.J. and R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the China Postdoctoral Science Foundation Grand (2019TQ0373), and the National Natural Science Foundation of China (No. 62002393).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this work are publicly available on OpenEI at https://openei.org/datasets/files/961/pub/.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kumar, N.M.; Chand, A.A.; Malvoni, M.; Prasad, K.A.; Mamun, K.A.; Islam, F.; Chopra, S.S. Distributed Energy Resources and the Application of AI, IoT, and Blockchain in Smart Grids. *Energies* **2020**, *13*, 5739. [CrossRef]
2. Bera, B.; Saha, S.; Das, A.K.; Vasilakos, A.V. Designing Blockchain-based Access Control Protocol in IoT-enabled Smart-grid System. *IEEE Internet Things J.* **2021**, *8*, 5744–5761. [CrossRef]
3. Singh, S.K.; Cha, J.; Kim, T.W.; Park, J.H. Machine Learning based Distributed Big Data Analysis Framework for Next Generation Web in IoT. *Comput. Sci. Inf. Syst.* **2021**, *18*, 597–618. [CrossRef]
4. Tightiz, L.; Yang, H. A Comprehensive Review on IoT Protocols' Features in Smart Grid Communication. *Energies* **2020**, *13*, 2762. [CrossRef]
5. Melo, G.C.G.D.; Torres, I.C.; Araújo, Í.B.Q.D.; Brito, D.B.; Barboza, E.D.A. A Low-cost IoT System for Real-time Monitoring of Climatic Variables and Photovoltaic Generation for Smart Grid Application. *Sensors* **2021**, *21*, 3293. [CrossRef] [PubMed]
6. Aderibole, A.; Aljarwan, A.; Rehman, M.H.U.; Zeineldin, H.H.; Mezher, T.; Salah, K.; Damiani, E.; Svetinovic, D. Blockchain Technology for Smart Grids: Decentralized NIST Conceptual Model. *IEEE Access* **2020**, *8*, 43177–43190. [CrossRef]
7. Zhuang, P.; Zamir, T.; Liang, H. Blockchain for Cybersecurity in Smart Grid: A Comprehensive Survey. *IEEE Trans. Ind. Informatics* **2020**, *17*, 3–19. [CrossRef]
8. Moni, M.; Melo, W.; Peters, D.; Machado, R. When Measurements Meet Blockchain: On Behalf of an Inter-NMI Network. *Sensors* **2021**, *21*, 1564. [CrossRef]
9. Kim, J.; Moon, J.; Hwang, E.; Kang, P. Recurrent Inception Convolution Neural Network for Multi Short-term Load Forecasting. *Energy Build.* **2019**, *194*, 328–341. [CrossRef]
10. Hafeez, G.; Alimgeer, K.S.; Khan, I. Electric Load Forecasting based on Deep Learning and Optimized by Heuristic Algorithm in Smart Grid. *Appl. Energy* **2020**, *269*, 114915. [CrossRef]

11. Singh, S.; Yassine, A.; Benlamri, R. Consumer Segmentation: Improving Energy Demand Management through Households Socio-analytics. In Proceedings of the 2019 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress, Fukuoka, Japan, 5–8 August 2019; pp. 1038–1045.
12. Rahimi, A.; Shahrestani, A.; Ramezani, S.; Zamani, P.; Tehrani, S.O.; Moghaddam, M.H.Y. Filter Based Time-Series Anomaly Detection in AMI using AI Approaches. In Proceedings of the 2021 5th International Conference on Internet of Things and Applications (IoT), Isfahan, Iran, 19–20 May 2021; pp. 1–6.
13. Chen, Y.Y.; Chen, M.H.; Chang, C.M.; Chang, F.S.; Lin, Y.H. A Smart Home Energy Management System Using Two-Stage Non-Intrusive Appliance Load Monitoring over Fog-Cloud Analytics Based on Tridium's Niagara Framework for Residential Demand-Side Management. *Sensors* **2021**, *21*, 2883. [PubMed]
14. Singh, S.K.; Pan, Y.; Park, J.H. OTS Scheme based Secure Architecture for Energy-Efficient IoT in Edge Infrastructure. *CMC-Comput. Mater. Contin.* **2021**, *66*, 2905–2922. [CrossRef]
15. Wang, Y.; Chen, Q.; Kang, C.; Zhang, M.; Wang, K.; Zhao, Y. Load Profiling and Its Application to Demand Response: A Review. *Tsinghua Sci. Technol.* **2015**, *20*, 117–129. [CrossRef]
16. Zheng, Z.; Yang, Y.; Niu, X.; Dai, H.N.; Zhou, Y. Wide and Deep Convolutional Neural Networks for Electricity-theft Detection to Secure Smart Grids. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1606–1615. [CrossRef]
17. Khelifi, N. A Cryptographic-based Approach for Electricity Theft Detection in Smart Grid. *Comput. Mater. Contin.* **2020**, *63*, 97–117.
18. Jiang, Z.; Lin, R.; Yang, F. A Hybrid Machine Learning Model for Electricity Consumer Categorization Using Smart Meter Data. *Energies* **2018**, *11*, 2235. [CrossRef]
19. Liu, W.; He, J.; Li, M.; Jin, R.; Zhang, Z. An Efficient Supervised Energy Disaggregation Scheme for Power Service in Smart Grid. *Intell. Autom. Soft Comput.* **2019**, *25*, 585–593. [CrossRef]
20. Chicco, G. Overview and Performance Assessment of the Clustering Methods for Electrical Load Pattern Grouping. *Energy* **2012**, *42*, 68–80. [CrossRef]
21. Gepperth, A.; Hammer, B. Incremental Learning Algorithms and Applications. In *European Symposium on Artificial Neural Networks*; ESANN: Bruges, Belgium, 2016.
22. Xu, J.; Xu, C.; Zou, B.; Tang, Y.Y.; Peng, J.; You, X. New Incremental Learning Algorithm with Support Vector Machines. *IEEE Trans. Syst. Man, Cybern. Syst.* **2018**, *49*, 2230–2241. [CrossRef]
23. Nguyen, H.L.; Woon, Y.K.; Ng, W.K. A Survey on Data Stream Clustering and Classification. *Knowl. Inf. Syst.* **2015**, *45*, 535–569. [CrossRef]
24. Jiang, Z.; Lin, R.; Yang, F. Incremental Electricity Consumer Behavior Learning Using Smart Meter Data. In Proceedings of the 2019 4th International Conference on Big Data and Computing, Guangzhou, China, 10–12 May 2019; pp. 54–59.
25. Al-Otaibi, R.; Jin, N.; Wilcox, T.; Flach, P. Feature Construction and Calibration for Clustering Daily Load Curves from Smart-meter Data. *IEEE Trans. Ind. Informatics* **2016**, *12*, 645–654. [CrossRef]
26. Panapakidis, I.P.; Alexiadis, M.C.; Papagiannis, G.K. Enhancing the Clustering Process in the Category Model Load Profiling. *IET Gener. Transm. Distrib.* **2015**, *9*, 655–665. [CrossRef]
27. Marxer, R.; Purwins, H. Unsupervised Incremental Nnline Learning and Prediction of Musical Audio Signals. *IEEE/ACM Trans. Audio, Speech, Lang. Process.* **2016**, *24*, 863–874. [CrossRef]
28. Zhang, Q.; Zhu, C.; Yang, L.T.; Chen, Z.; Zhao, L.; Li, P. An Incremental CFS Algorithm for Clustering Large Data in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2017**, *13*, 1193–1201. [CrossRef]
29. Aggarwal, C.C.; Han, J.; Wang, J.; Yu, P.S. A Framework for Projected Clustering of High Dimensional Data Streams. In Proceedings of the 30th International Conference on Very Large Data Bases, VLDB Endowment, Toronto, ON, Canada, 31 August–3 September 2004; pp. 852–863.
30. Kriegel, H.P.; Kröger, P.; Ntoutsi, I.; Zimek, A. Density based Subspace Clustering Over Dynamic Data. In *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 387–404.
31. Zhang, B.; Qin, S.; Wang, W.; Wang, D.; Xue, L. Data Stream Clustering based on Fuzzy C-Mean Algorithm and Entropy Theory. *Signal Process.* **2016**, *126*, 111–116. [CrossRef]
32. Braverman, V.; Frahling, G.; Lang, H.; Sohler, C.; Yang, L.F. Clustering High Dimensional Dynamic Data Streams. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 576–585.
33. Hyde, R.; Angelov, P.; MacKenzie, A.R. Fully Online Clustering of Evolving Data Streams into Arbitrarily Shaped Clusters. *Inf. Sci.* **2017**, *382–383*, 96–114. [CrossRef]
34. Zhang, Y.; Ji, G.; Zhao, B.; Sheng, B. An Algorithm for Mining Gradual Moving Object Clusters Pattern from Trajectory Streams. *Comput. Mater. Contin.* **2019**, *59*, 885–901. [CrossRef]
35. Wiwatcharakoses, C.; Berrar, D. SOINN+, A Self-organizing Incremental Neural Network for Unsupervised Learning from Noisy Data Streams. *Expert Syst. Appl.* **2020**, *143*, 113069. [CrossRef]
36. Mets, K.; Depuydt, F.; Develder, C. Two-stage Load Pattern Clustering Using Fast Wavelet Transformation. *IEEE Trans. Smart Grid* **2016**, *7*, 2250–2259. [CrossRef]
37. Wang, Y.; Chen, Q.; Kang, C.; Xia, Q. Clustering of Electricity Consumption Behavior Dynamics Toward Big Data Applications. *IEEE Trans. Smart Grid* **2016**, *7*, 2437–2447. [CrossRef]

38. Shaukat, M.A.; Shaukat, H.R.; Qadir, Z.; Munawar, H.S.; Kouzani, A.Z.; Mahmud, M. Cluster Analysis and Model Comparison Using Smart Meter Data. *Sensors* **2021**, *21*, 3157. [CrossRef]
39. Jiang, Z.; Lin, R.; Yang, F.; Wu, B. A Fused Load Curve Clustering Algorithm based on Wavelet Transform. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1856–1865. [CrossRef]
40. Hao, S.; Zhao, P.; Hoi, S.C.; Miao, C. Learning Relative Similarity from Data Streams: Active Online Learning Approaches. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, Melbourne, Australia, 18–23 October 2015; pp. 1181–1190.
41. Losing, V.; Hammer, B.; Wersing, H. Incremental On-line Learning: A Review and Comparison of State of the Art Algorithms. *Neurocomputing* **2018**, *275*, 1261–1274. [CrossRef]
42. Chen, C.P.; Liu, Z. Broad Learning System: An Effective and Efficient Incremental Learning System without the Need for Deep Architecture. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 10–24. [CrossRef] [PubMed]
43. Chaudhari, A.; Mulay, P. A Bibliometric Survey on Incremental Clustering Algorithm for Electricity Smart Meter Data Analysis. *Iran J. Comput. Sci.* **2019**, *2*, 197–206. [CrossRef]
44. Zubaroğlu, A.; Atalay, V. Data Stream Clustering: A Review. *Artif. Intell. Rev.* **2021**, *54*, 1201–1236. [CrossRef]
45. Xu, D.; Tian, Y. A Comprehensive Survey of Clustering Algorithms. *Ann. Data Sci.* **2015**, *2*, 165–193. [CrossRef]
46. Du, M.; Ding, S.; Jia, H. Study on Density Peaks Clustering based on K-nearest Neighbors and Principal Component Analysis. *Knowl.-Based Syst.* **2016**, *99*, 135–145. [CrossRef]
47. Ghesmoune, M.; Lebbah, M.; Azzag, H. State-of-the-art on Clustering Data Streams. *Big Data Anal.* **2016**, *1*, 13. [CrossRef]
48. Barddal, J.P.; Gomes, H.M.; Enembreck, F.; Barthès, J.P. SNCStream+: Extending a High Quality True Anytime Data Stream Clustering Algorithm. *Inf. Syst.* **2016**, *62*, 60–73. [CrossRef]
49. Vendramin, L.; Campello, R.J.; Hruschka, E.R. On the Comparison of Relative Clustering Validity Criteria. In Proceedings of the 2009 SIAM International Conference on Data Mining, SIAM, Sparks, NV, USA, 30 April–2 May 2009; pp. 733–744.
50. Chen, C.; Twycross, J.; Garibaldi, J.M. A New Accuracy Measure based on Bounded Relative Error for Time Series Forecasting. *PLoS ONE* **2017**, *12*, 1–23. [CrossRef] [PubMed]
51. Lusis, P.; Khalilpour, K.R.; Andrew, L.; Liebman, A. Short-term Residential Load Forecasting: Impact of Calendar Effects and Forecast Granularity. *Appl. Energy* **2017**, *205*, 654–669. [CrossRef]