A Survey on Data Stream Clustering and Classification

Article *in* Knowledge and Information Systems · December 2014

DOI: 10.1007/s10115-014-0808-1

3 authors:

Hai-Long Nguyen
Nanyang Technological University
15 PUBLICATIONS **312** CITATIONS
SEE PROFILE

Yew-Kwong Woon
Airbus Group
24 PUBLICATIONS **293** CITATIONS
SEE PROFILE

Wee Keong Ng
Nanyang Technological University
435 PUBLICATIONS **5,178** CITATIONS
SEE PROFILE

A Survey on Data Stream Clustering and Classification

Hai-Long Nguyen and Wee-Keong Ng
Nanyang Technological University, Singapore
and
Yew-Kwong Woon
EADS Innovation Works, Singapore

Nowadays, with the advance of technology, many applications generate huge amounts of data streams at very high speed. Examples include network traffic, web click streams, video surveillance, and sensor networks. Data stream mining has become a hot research topic. Its goal is to extract hidden knowledge/patterns from continuous data streams. Unlike traditional data mining where the dataset is static and can be repeatedly read many times, data stream mining algorithms face many challenges and have to satisfy constraints such as bounded memory, single-pass, real-time response, and concept-drift detection. This paper presents a comprehensive survey of the state-of-the-art data stream mining algorithms. It identifies mining constraints and proposes a general model for data stream mining, and depicts the relationship between traditional data mining and data stream mining. Furthermore, it analyzes the advantages and limitations of data stream algorithms, and suggests potential areas for future research.

Categories and Subject Descriptors: []:

General Terms:

Additional Key Words and Phrases:

1. INTRODUCTION

Traditional data mining research mostly focused on mining resident and static data repositories. However, technological developments give rise to the emergence of data streams and changed the way people store, communicate and process data. Nowadays, many organizations generate large amounts of data at higher speed than ever. For example, on a daily basis, Google handles more than 100 million searches, NASA satellites generate around 1.5 TB images, and WalMart records more than 20 million transactions. The new *"intensive data"* research problem is: How can one model an infinite amount of continuous, rapid, and time-evolving data streams with a time-critical requirement?

In this survey, we focus on clustering and classification of data stream as they are the two most frequent forms of data mining. Clustering aims to group a dataset into subsets (clusters), where data objects within a cluster are 'similar' and data objects in different clusters are 'dissimilar' with respect to a given similarity measure. Classification is the process of finding a general model from known data, and then using this model to predict class labels for new data objects. Readers may find a good introduction of traditional data mining in a book of Jiawei Han [Han 2005].

There are much published work on data stream mining, including surveys that give overviews of various approaches. Gaber *et al.* introduced an illustrative survey

with theoretical foundations and basic algorithms on data stream mining [Gaber et al. 2005]. However, it may be hard to understand the survey as the authors did not give any clear distinction among these algorithms. From a statistical point of view, Gama and Rodrigues [Gama and Rodrigues 2009] focused on illustrating examples of data stream mining with specific algorithms and applications. Unfortunately, the presented algorithms are quite outdated. Aggarwal [Aggarwal 2009] attempted to give a broader overview of data stream mining, where more mining tasks and real applications were discussed. Nevertheless, it does not give readers a comprehensive understanding of current data stream mining methods.

In this paper, we introduce a comprehensive survey of data stream clustering and classification algorithms. We present preliminaries and overview of data stream mining in Section 2. Then, we discuss in details state-of-the-art algorithms, including their merits and limitations. We thoroughly classify them into different categories based on their approaches and derive the relationships between traditional mining algorithms and stream mining algorithms in Sections 3 and 4. We also analyze capabilities of each algorithm in terms of addressing constraints in a data stream setting. Finally, we discuss future research in Section 5 and conclude the survey in Section 6.

1.1 Sample Applications & Software of Data Stream Mining

Applications:

—*Mining query streams*: Searching the Web to retrieve information has become an essential activity of our everyday life. Existing search engines such as Google, Bing, and Yahoo handle millions of queries on a daily basis. Mining query streams to provide users better searching results has attracted much research work. For example, Zeng *et al.* clustered Web search results to facilitate users' quick browsing through search results [Zeng et al. 2004]. Users generally enter very short text queries, which may produce imprecise searching results due to users' own ambiguity. Chien and Immorlica studied temporal correlation among text queries over a period of time and refine users' search by suggesting alternative queries [Chien and Immorlica 2005].

—*Network monitoring*: Internet includes many routers that are connected and communicate with each other by send IP packets. To manage such networks, we need to analysis traffic data to discover usage patterns and unusual activities in real time. Traffic data are recorded in the form of log files at many levels, such as packet logs containing source and destination IP addresses; flow logs storing the number of packets sent, start time, end time, and protocol; and SNMP logs aggregate the number of bytes sent over each link every few minutes [Muthukrishnan 2003]. An example of network management is to detect and prevent malicious attacks in a large Internet service provider network. A data stream classifier is required to classify in real time different kinds of attacks, such as denial-of-service (DOS), unauthorized access from a remote machine (R2L), unauthorized access to local super-user privileges (U2R), surveillance and other probing attacks.

—*Sensor networks*: A sensor network consists of spatially distributed autonomous sensors that cooperatively monitor an environment. These sensors can sense physical values about the environment, such as temperature, sound, vibration,

pressure, humidity and light; they can cooperatively pass their data through the network to a center. Sensor networks are involved in many real-life applications, such as, traffic monitoring, smart homes, habitat monitoring, and healthcare [Gama and Gaber 2007]. For example, modern hospitals are equipped with patient monitoring system to improve health care quality and staff productivity. Many body sensors, which are connected to critically ill patients, can produce massive physiological data, such as temperature, electrocardiogram, pulse oximetry, and blood pressure. Since sensor devices only store updated data and human eyes cannot detect these signal subtleties, the system must analyze healthcare data streams in real time and extract meaningful information for medical professionals, such as clinical rules to identify significant events [Sow et al. 2010].

—*Social network streams*: Online social networks (OSNs) have become more and more popular; for example, Facebook, Twitter, and LinkedIn have millions of active users. Such networks generate tremendous online data streams, such as text, multimedia, linkage, interaction. There is much research on social network mining. For example, stream clustering methods are used to detect communities, and monitor their evolution in social networks. They can explain how communities emerge, expand, shrink, and evolve in a social network. Moreover, stream classification algorithm help to classify different types of users, or categorize discussion topics in social networks.

Software: There is some useful, open-source software for data stream mining research.

—WEKA[1]: WEKA is the most well-known data mining software within the academic environment. WEKA includes a collection of learning algorithms such as data pre-processing, classification, regression, clustering, association rules, and visualization.

—Massive Online Analysis (MOA)[2]: MOA is based on the WEKA framework that is designed for data stream learning. It includes many online learning algorithms for evolving data streams; for example, very fast decision tree [Domingos and Hulten 2000], and ensemble learning [Oza 2005]. Moreover, MOA provides data stream generators, such as SEA concepts, STAGGER, and rotating hyper-plane.

—RapidMiner[3]: RapidMiner is another open-source software for data mining. RapidMiner is more powerful than WEKA as it includes all algorithms in WEKA and other advanced algorithms. Moreover, it is more intuitive as it is able define a mining process as a series of operators and provides more visualization tools.

2. OVERVIEW OF DATA STREAM MINING

We define a data stream $\mathcal{DS}$ as a sequence of data objects or samples: $\mathcal{DS} = (x_1, x_2, ..., x_i, ...)$, where x_i is the i-th arrived data object. Each data object x_i has a label $y_i \in \mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ when classifying data stream, and there is no label when clustering data stream.

[1]http://www.cs.waikato.ac.nz/ml/weka
[2]moa.cs.waikato.ac.nz
[3]http://rapid-i.com

2.1 Constraints & a general model

Data streams have intrinsic characteristics, such as possibly infinite volume, chronological order and dynamical changes. For example, Google processes more than 100 million searches daily, each of which is attached with a time stamp; and these searches are changed according to different hot topics at different times. **TODO: examples don't quite illustrate**

	Traditional Data Mining	Data Stream Mining
Number of passes	multiple	single
Time	unlimited	real-time
Memory	unlimited	bounded
Number of concepts	one	multiple
Result	accurate	approximate

Table I. Comparisons between traditional data mining and data stream mining.

Table I shows comparisons between traditional data mining and data stream mining. Traditional data mining is able to scan datasets many times; execute with unlimited time and memory; has only one concept; and needs to produce fairly accurate results. On the other hand, data stream mining may produce approximate results and has to satisfy constraints, such as *single-pass, real-time response, bounded memory,* and *concept-drift detection*:

—*Single-pass :* Unlike traditional data mining that may read static datasets repetitively many times, each sample in a data stream is examined at most once and cannot be backtracked.

—*Real-time response:* Many data stream applications such as stock market prediction require real-time response. The amount of time for processing the data and providing decision must be fast.

—*Bounded memory :* The amount of arriving data is extremely large or potentially infinite. As we may only compute and store a small summary of the data streams and possibly throw away the rest of the data; approximate results are acceptable.

—*Concept-drift detection:* In data streams, concept drifts refer to the situation when the discovered patterns (or the underlying data distribution) change over time. A formal definition of concept-drifts was introduced by Kelly *et al.* [Kelly et al. 1999]. Here, a concept at time instant t is defined as a set of probabilities of the classes and class-conditionals:

$$\mathcal{CD} = \{(P(c_1), P(x_t|c_1)); (P(c_2), P(x_t|c_2)); \ldots; (P(c_m), P(x_t|c_m))\}.$$

By monitoring changes in this set of probabilities $\mathcal{CD}$, a data stream model is able to detect and adapt itself according to concept drifts .

To help readers understand an overview of data stream mining, we propose a general model of data stream algorithms in Figure 1. When a data stream comes, a buffer is used to store the most recent data. The stream mining engine reads the buffer to create a synopsis of the data in memory. In order to maintain the synopsis, the system may apply different time-window and computational approaches. When certain criteria are triggered; for example, a users request or after a certain time

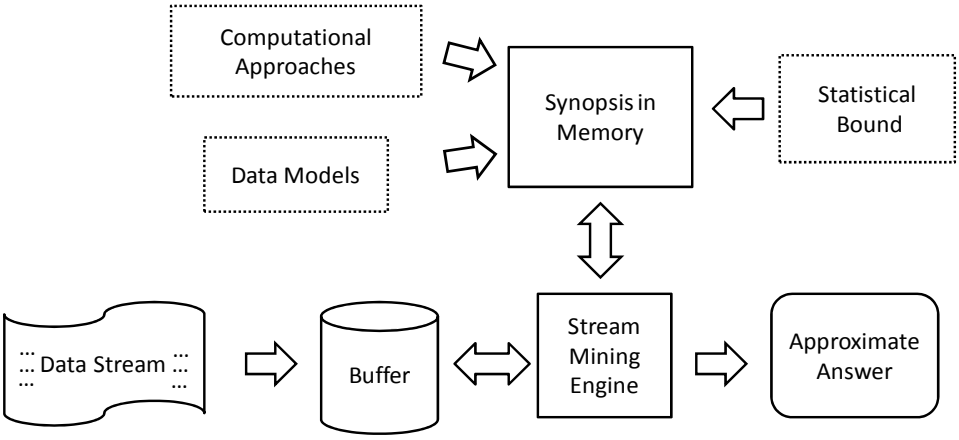

Fig. 1. A general model for data stream mining.

lapse; the stream mining engine will process the synopsis and output approximate results. In general, most data stream algorithms are derived and adapted from traditional mining algorithms.

2.2 Time Window

As data streams are potentially infinite, it is possible to only able to process a portion of the entire data streams. This interesting portion is defined as a time-window of data objects. $W[i,j] = (x_i, x_{i+1}, ..., x_j)$, where $i < j$. There are different types of time-windows: landmark window, sliding window, fading window and tilted-time window.

Landmark window

In the landmark window, we are interested in the entire data stream from starting time instant 1 to the current time instant t_c; the window is $W[1, t_c]$. Using the landmark window, all transactions in the window are equally important; there is no difference between past and present data. However, as data stream evolves continuously, the model built with old data objects may become inconsistent with the new ones. In order to emphasize recent data, one may apply the sliding window, tilted window, or fading window variants.

Sliding window

In the sliding window variant $W[t_c - w + 1, t_c]$, we are only interested in the w most recent transactions; the others are eliminated. The mining result is dependent on the size of the window w. If w is too large and there is a concept drift, the window possibly contains outdated information and the accuracy of the model decreases. If w is small, the window may have deficient data, the model over-fits and suffers from large variances. Previous work considers a fixed value for the size of the sliding window specified by users or an experimental value. Recently, there are proposals for flexible sliding windows where the size of the window changes according to the accuracy of the model [Bifet 2010; Last 2002]. When the accuracy is high, the window extends; and when the accuracy is low, the window shrinks.

Fading window

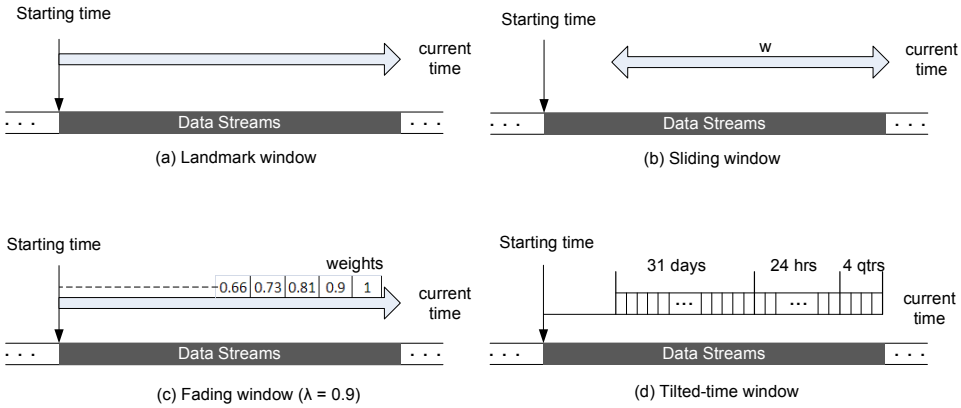

Fig. 2. Examples of time-windows.

In the fading window variant, each data object is assigned a different weight according to its arrival time so that new transactions recieve higher weights than old ones [Cao et al. 2006; Chen and Tu 2007; Han 2005]. Using the fading window, we reduce the effect (importance) of old and out-dated transactions on the mining results. A decreasing exponential function $f(\triangle t) = \lambda^{\triangle t}$ $(0 < \lambda < 1)$ is usually used in the fading model. In this function, $\triangle t$ is the age of a data object that is equal to time difference between the current time and its arrival time. The fading window needs to choose a suitable fading parameter λ, which is typically set in the range [0.99, 1] in real applications.

Tilted-time window

The tilted-time window variant is somewhere between the fading window and sliding window variants [Aggarwal et al. 2003; Han 2005]. It applies different levels of granularity with regard to the recency of data. One is more interested in recent data at fine scale than long-term data from the past at coarse scale. Tilted-time window provides a nice tradeoff between storage requirements and accuracy. It approximately stores the entire dataset and considers all the transactions at the same. However, the model may become unstable after running for a long time. For example, the tree structure in FP-Stream [Chris et al. 2003] will become very large over time, and the process of updating and scanning over the tree may degrade its performance. Similarly, the micro-structures in On-Demand Classification [Aggarwal et al. 2006] will become larger and larger that may give rise to the problem of low-purity clustering with large micro-clusters [Zhou et al. 2008].

Figure 2 shows examples of four different time-windows. For fading window, λ is set to 0.9; the weights of data objects decreases. For tilted-time window, we store the 4 most recent quarters of an hour, then the last 24 hours, and last 31 days.

2.3 Computational Approaches

Besides various time-window variants, there are computational approaches to process the data streams. **Incremental Learning**

Incremental learning is another computational approach for data streams [Dang et al. 2009; Guha et al. 2004; Guha et al. 2000; Hulten et al. 2001; Leite et al.

2009; Lhr and Lazarescu 2009; Sattar et al. 2009; Smith and Alahakoon 2009; Street and Kim 2001]. In this approach, the model incrementally evolves to adapt to changes in incoming data. There are two ways to update the model: window and data instance. Street *et al.* deployed an ensemble of classifiers for data stream [Street and Kim 2001]. It evaluated a window of incoming data and adapted the model by adjusting the weight of each classifier or replacing an old classifier with an updated one. The incremental approach has the advantage of providing mining results instantly, but it requires much computational resources.

Two-phase Learning

Two-phase learning, also known as online-offline learning, is a common computational approach in data streams [Aggarwal et al. 2003; 2006; Cao et al. 2006; Chen and Tu 2007; Kranen et al. 2010; Park and Lee 2004; 2007; Rai et al. 2009; Seidl et al. 2009; Wan et al. 2009]. The basic idea is to divide the mining process into two phases. In the first phase (online phase), a synopsis of data is updated in a real-time manner. In the second phase (offline phase), the mining process is performed on the stored synopsis whenever a user sends a request. The two-phase learning approach is able to process data streams at very high speed. However, its limitation is that users must wait until the mining results are available.

3. CLUSTERING

Clustering or data segmentation is the process of grouping objects into different sets called clusters. The goal is that data objects in the same cluster are similar and are dissimilar to data objects in other clusters. Clustering is a well-studied problem, and many clustering methods have been proposed in the literature.

As mentioned in the general model (Figure 1), data stream algorithms typically maintain synopses of data streams using different time-window and computational approaches. Furthermore, data stream algorithms generally extend traditional algorithms to work for data streams with the goal to satisfy constraints, such as bounded memory, single-pass, real-time processing, and concept drifts. Both traditional and data stream clustering methods can be classified into the following same categories: *partitioning methods, hierarchical methods, density-based methods, grid-based methods,* and *model-based methods*.

To give readers a broad overview of data stream clustering, we start with a quick review of traditional clustering, then discuss the most representative data stream clustering algorithms. We also identify the connection between traditional clustering and data stream clustering, and discuss their advantages as well as limitations.

3.1 Traditional Data Clustering

Partitioning methods: A partitioning method constructs k partitions where each partition represents a cluster. Using a pre-defined parameter k, a partition clustering method usually applies some heuristic to initialize partitions. Then, it iteratively reassigns objects from one group to another group in order to minimize its objective function. The most popular partitioning methods are *k-means* and *k-medoids*.

- *k-means*: In the k-means method, each cluster is represented by a mean or a center. First, it randomly selects k objects for the k means from the dataset. Next, every object is assigned to the cluster whose center is closest. Then, the mean of

each cluster is recomputed for the next iteration. The process stops whenever its criterion function converges. Typically, the square-error function is used to measure the compactness of the clusters.

$$f = \sum_{i=1}^{k} \sum_{x \in C_i} \mid x - m_i \mid,$$

where x is the data object and m_i is the mean of cluster C_i. The k-means method is quite simple, easy to implement, relatively scalable and efficient. However, it is only suitable for finding spherical-shaped clusters and is sensitive to noise and outliers as such kind of data greatly distort the mean value.

- *k-medoids*: To overcome the outlier sensitivity problem of the k-means method, the $k-medoids$ method chooses a data object, called *medoid*, to represent a cluster. The basic idea of k-medoids algorithm is similar to the k-means algorithm; data objects are reassigned to another cluster that has the nearest medoid, such as in PAM [Kaufman and Rousseeuw 1990]. The objective function becomes the sum square-error of all the objects to their nearest medoids. This process is performed repeatedly by a swapping method that tries to improve the clustering result by replacing the medoid with a non-representative object. CLARAN (Clustering LARge Application) [Ng and Han 2002] is an extension of PAM to deal with large datasets. Instead of processing the entire dataset, it uses a sampling-based method to select a small portion of the dataset and then performs the PAM method on this selected sample.

Hierarchical methods:

A hierarchical method aims to group data objects into a hierarchical tree of clusters, which can be presented by a dendrogram. A dendrogram is a visual tool to show the way clusters are created. There are two main approaches in hierarchical clustering: agglomerative (bottom-up) and divisive (top-down) approaches.

In the agglomerative approach, each data object is initially set as a single cluster and then these clusters are merged together to form larger and larger ones. This process repeats until a single cluster or certain termination conditions are satisfied. The division approach starts by grouping all data objects in one single cluster. It then divides the cluster into smaller ones until each data object forms its own cluster or until certain termination conditions are satisfied. Most hierarchical methods are agglomerative; they differ only in their definition of the similarity between clusters, such as minimum distance, maximum distance, mean distance, or average distance. These distances have different effects on the performance of the algorithms. For example, maximum distance will produce high-quality clusters, but it is sensitive to outliers and requires more computations. Some representative algorithms are BIRCH [Zhang et al. 1996], CURE [Guha et al. 1998], ROCK [Guha et al. 1999], and CHAMELEON [Karypis et al. 1999].

Density-based methods: Discovering arbitrary-shaped clusters is quite challenging, especially using the partitioning method. Density-based clustering methods are proposed to resolve such problems. In this approach, clusters are considered as dense regions of objects and are separated by sparse regions with low density in the data space.

- DBSCAN [Martin et al. 1996]: DBSCAN is a popular density-based cluster-

ing method that defines a cluster as a maximal set of density-connected points. An object p is called a core object or a dense object if its ϵ-neighborhood has at least $MinPts$ objects. All objects in the ϵ-neighborhood of p are *density-reachable* by p. DBSCAN first creates a list of core objects, and a new cluster with each core object. Then, these clusters are iteratively extended and merged using the density-reachability relation. The process stops whenever clusters cannot be extended anymore. Hence, final clusters may have arbitrary shapes.

- OPTICS [Ankerst et al. 1999]: OPTICS is an extension of DBSCAN that is less sensitive to user-defined parameters. A core point p in the OPTICS algorithm is defined together with its core-distance ϵ' that defines the smallest neighborhood with $Minpts$ points. The reachability distance between an object q and a core object p is the larger value of the core-distance of p and their distance. OPTICS records the reachability distance of every core point and orders data points in such a way that the $(i+1)^{th}$ point has the smallest reachability distance to the i^{th} one. By plotting the order of data points with regard to the reachability distance, a reachability plot (a special kind of dendrogram) can be created to visualize the clustering structures at several levels of details.

- PreDeCon [Bohm et al. 2004]: Bohm *et al.* extended the DBSCAN algorithm using a preference distance measure to capture the subspace of each cluster. In this way, PreDeCon works with high-dimensional data. The preference distance is simply a weighted Euclidean distance that ignores attributes having variant values greater than a threshold value. Moreover, definitions in DBSCAN (such as ϵ-neighborhood, core point, density-reachable, etc) are adapted using the preference distance. Similar to DBSCAN, PreDeCon constructs clusters by extending and merging its core points. PreDeCon shows better performance than DBSCAN in high-dimensional datasets such as biological datasets.

- DENCLUE [Alexander et al. 1998] : DENCLUE is a combination of density-based and grid-based approaches. It divides the high-dimensional data space into grid cells, each of which contains information of data points within the cell. DENCLUE generalizes the definition of density by using a mathematical influence function, such as a square wave influence function or a Gaussian influence function. The density of an object is defined as the sum of influence functions from surrounding data objects. The local maxima of density function is called density attractor; arbitrary shape clusters are constructed by locating density attractors and their attracted neighbors.

Grid-based methods:

Grid-based clustering methods quantize data space into a multi-resolution grid structure. The grid structure contains many cells, each of which has a subspace and stores summary information of data objects within the subspace. By processing this data structure, clustering methods are generally fast and are independent of the size of the dataset.

- STING [Wang et al. 1997]: STING is an example of a grid-based clustering method for mining spatial data. STING divides the data space into rectangular cells and creates a hierarchical structure having different levels of granularity. Statistical information is stored in each cell, such as the mean, standard deviation, minimum, maximum, and the type of distribution. Statistical information of higher level cells

can be easily computed from lower level cells. STING only needs a single scan of the dataset to generate the hierarchical grid structure with complexity $O(N)$, where N is the total number of data objects. Moreover, the query processing time is $O(G)$, where G is the total number of finest grid cells.

- WaveCluster [Sheikholeslami et al. 2000]: WaveCluster is another grid-based clustering method for detecting arbitrarily-shaped clusters. The key idea of WaveCluster is to apply wavelet transformation to make the clusters more distinct and salient in the transformed space. WaveCluster is computationally efficient and is able to detect the clusters at different scales due to the multi-resolution property of wavelet transformation. It is robust to outliers and the order of input data. However, WaveCluster is unsuitable for high-dimensional data since wavelet transform is applied to each dimension of the feature space.

Model-based methods: In the above clustering algorithms, clusters are formed by optimizing certain criteria that rely on the distance among data objects. In the model-based approach, data are assumed to be generated by a mixture of probability distributions and each component of the mixture represents a cluster. Model-based clustering methods use a generative statistical model and then try to optimize the likelihood between data and this model.

- Expectation-Maximization (EM) [Dempster et al. 1977]: EM-based algorithm is a soft clustering method; it is robust to noise and is able to handle missing data. In the initial step, the EM-based algorithm guesses the parameters of the model, such as the maximum number of clusters and their centers. Then, it iteratively performs two alternating steps: the Expectation (E) step and the Maximization (M) step. In the E step, for each data object, we use current parameters of the model to calculate its membership for each cluster. In the M step, we re-estimate the model parameters; for example, re-calculate the new centers to maximize the likelihood between the model and the assumed model.

- Self-Organizing Map (SOM): Introduced by Kohonen in 1990, SOM is one of the most popular neural network methods for clustering. SOM can be viewed as a nonlinear projection from high-dimensional data space onto a lower-dimensional regularized grid. SOM first randomizes the weight vectors of the grid. Then, for each incoming data object, it finds the best-matching unit and scales its neighbor unit. SOM assumes that there are some topologies among the input data objects; and the grid will eventually take on these topologies. The organization of SOM forms a feature map that can be used to visualize the dataset and explore discovered clusters.

3.2 Data Stream Clustering

Partitioning methods:

- STREAM [O'Callaghan et al. 2002]: STREAM is one of the first data stream algorithms. It addresses the bounded memory and single-pass constraints. It uses a divide-and-conquer strategy to perform clustering incrementally and hierarchically. The STREAM algorithm works in a window mode. When an amount of data objects fits into main memory, it is clustered using LSEARCH, an advanced k-medoids algorithm. Intermediate medians with its weight representing result clusters are stored. The process is repeated for subsequent windows. The LSEARCH is recursively applied to the representative medians. Together with a sampling

method, STREAM is able to perform clustering with limited time and memory. However, it fails to detect concept drifts and discover non-spherical clusters, and is sensitive to parameter k due to the intrinsic properties of k-medoids.

Hierarchical methods:

- CluStream [Aggarwal et al. 2003]: CluStream extends the traditional clustering method BIRCH [Zhang et al. 1996] for data streams. CluStream uses micro-clusters to store the summary of data streams. A micro-cluster is an extension of the Clustering Feature in BIRCH with two more dimensions: time and square of time. It applies the tilted time window to optimize the number of stored snapshots (the status of micro-clusters in the data stream). CluStream follows the online-offline approach, which is similar to the multi-phase clustering technique in BIRCH. In the online phase, it continually maintains a set of q micro-clusters in the data stream. In the offline phase, it performs k-means to cluster the stored q micro-clusters. CluStream analyzes the evolution of clusters by using additional property to extract information of micro-clusters during a specific time range.

Based on CluStream's framework, many extensions have been proposed. HPStream [Aggarwal et al. 2004] addresses the problem of high dimensional data streams by deploying a projection technique to select the best attribute set for each cluster (subspace clustering). Similar to CluStream, HPStream maintains micro-cluster structures as synopses of data streams. Furthermore, each micro-cluster consists of a set of relevant attributes, which can be considered its subspace. When a new data instance arrives, the average Manhattan distance between the new instance and each cluster is computed. Only relevant attributes of the clusters are utilized in the distance computation. Then, the new instance is assigned to the closest cluster if their distance does not exceed a limiting range, a multiple of the cluster's radius. Moreover, the statistical properties of the closest cluster are also updated. HPStream only maintains a fix number of micro-clusters. When the number of clusters reaches a threshold value, it removes the oldest cluster to give space for a new one.

SWClustering [Zhou et al. 2008] indentifies a problem that the clustering results of CluStream may degrade after running for a long time. For example, when the center of a micro-cluster gradually shifts, CluStream maintains the micro-cluster with growing radius, instead of splitting it into many micro-clusters. To summarize data streams, SWClustering creates a Temporal Cluster Feature (TCF) for a sliding window. The TCF is similar to a micro-cluster; the only difference is that TCF stores the latest timestamp, while micro-cluster stores the sum of timestamps. An Exponential Histogram of Cluster Features (EHCF), a collection of TCFs, is used to capture the evolution of individual clusters. SWClustering not only produces more qualified clustering due to the fine granularity of EHCFs, it also has better performance than CluStream in terms of running time and memory usage. E-Stream [Udommanetanakit et al. 2007] classifies cluster evolution into five categories: appearance, disappearance, self evolution, merging, and splitting. It utilizes the fading model and cluster histograms to identify the type of cluster evolution.

- REPSTREAM [Lhr and Lazarescu 2009]: Inspired by CHAMELEON [Karypis et al. 1999], REPSTREAM is another graph-based hierarchical clustering approach for data streams. To identify clusters, REPSTREAM updates two sparse graphs

that are formed by connecting each vertex to its k-nearest vertices. The first graph captures the connectivity relationship among coming data points and is used to select a set of representative vertices. The second graph of representative vertices helps to make clustering decisions at a higher level. REPSTREAM applies the fading window to diminish the effect of old data. REPSTREAM keeps track of the connectivity between the representative vertices and performs merging or splitting according to their connectivity.

Density-based methods:

- DenStream [Cao et al. 2006]: DenStream is a density-based stream clustering algorithm that extends the DBSCAN algorithm. Similar to CluStream, DenStream uses micro-clusters to capture synopsis information of data streams; its online component continually updates the micro-clusters collection. Each micro-cluster has a center and a radius that are derived from its clustering feature vector. DenStream applies the fading model where elements of its feature vector decrease over time. Given threshold values for the weight and radius, there are three types of micro-clusters: a core micro-cluster, a potential core micro-cluster, and an outlier micro-cluster. For the offline components, it applies DBSCAN on these kinds of micro-clusters; a cluster is created as a group of micro-clusters that are dense and close to another.

- OPTICS-Stream [Tasoulis et al. 2007]: OPTICS-Stream is an extension of the OPTICS algorithm for data streams. Similarly to DenStream, it uses micro-clusters and the fading model to construct the synopsis. The offline component performs clustering by using the definitions of core-distance and reachability distance in OPTICS [Ankerst et al. 1999]. The reachability plot, which becomes a 3-D plot given the time dimension, can be used to visualize the changes of the cluster structures in the data stream over time.

- incPreDecon [Kriegel et al. 2011]: incPreDecon is an incremental version of the PreDeCon [Bohm et al. 2004] algorithm, which is designed to work for dynamic data. The algorithm supports two types of updates, a single instance update and batch update. Both updating methods share the same strategy. They first identify a group of affected objects, whose properties may be changed into one of the following cases: (i) core $\rightarrow$ non-core, (ii) non-core $\rightarrow$ core, and (iii) core $\rightarrow$ core but under different preferences. Given new arriving data p, the group of affected objects consists of reachable objects from p. Properties of these objects, as well as their clusters, are updated according. The ability of incPreDecon to work for data streams is still unclear as only experiments with relatively small datasets have been performed.

- D-Stream [Chen and Tu 2007]: D-Stream is a density-based clustering method for data streams. It can be considered as an extension of the DENCLUE algorithm [Alexander et al. 1998]. In D-Stream, each dimension is divided into p segments. With this, a grid of equal size hyper-rectangular cells is created. Similar to a micro-cluster, a grid cell in D-Stream is used to store synopsis information of data objects falling into it. As D-Stream uses the fading model to decrease the weight of cell over time, it periodically removes sparse grid cells to save memory and accelerate the mining process. D-Stream performs clustering upon a user request. A cluster is defined as a group of adjacent dense grid cells.

Grid-based methods:

- MR-Stream [Wan et al. 2009]: MR-Stream is a multi-resolution density-based clustering method for data stream. MR-Stream takes advantages of both D-Stream [Chen and Tu 2007] and STING [Wang et al. 1997]. MR-Stream proposes a tree of grid cells to capture the hierarchical structure of the data space. A deeper level tree node has higher granularity level. Similar to D-Stream, MR-Stream applies the fading model and periodically prunes sparse grid cells to save memory. Additionally, MR-Stream significantly reduces the number of tree nodes by merging sibling nodes of a parent node if they are all dense or sparse. Thus, MR-Stream preserves more memory than D-Stream and accelerates the clustering process. Moreover, MR-Stream provides a better cluster result by extending the neighborhood range concept and supports a memory sampling method that helps users to detect when concept drifts occur.

- CellTree [Park and Lee 2004] : CellTree is another grid-based algorithm for data streams. It starts by partitioning the data space into a set of mutually exclusive equal size cells. When a weight of a cell is greater than a threshold value, the cell is dynamically divided into two intermediate cells using a *hybrid-partition* method that selects a better method between μ-partition and σ-partition methods. The μ-partition divides a dimension with the largest standard deviation, while the σ-partition method choose to split a dimension with the smallest standard deviation. To save memory, CellTree prunes sparse cells with density less than the threshold value. CellTree has been extended to a better version Cell*Tree [Park and Lee 2007] that uses a *B+Tree* to store the synopses of data streams. The hybrid-partition method has a drawback that CellTree is not able to employ any indexing structure to access a specific grid cell immediately. In Cell*Tree, a dense grid cell is divided into a fixed number of equal size grid cells, which are indexed easily based on its order. Moreover, Cell*Tree applies the fading model to emphasize the latest change of information in a data stream on the clusters.

Model-based methods:

- SWEM [Dang et al. 2009]: SWEM is an EM-based clustering algorithm for data streams using a sliding window. In SWEM, each micro-component is represented by a tuple consisting of a weight, a mean, and a covariance matrix. For the first data window, SWEM applies the EM algorithm to obtain the converged parameters. Then, in the incremental phase, SWEM utilizes the converged parameters in the previous window of data object as the initial values for the mixture models' parameters. If the two sets of parameters are significantly different; SWEM re-distributes components in the entire data space by splitting those micro-components with large variance and merging neighbor micro-components. SWEM also deploys the fading model to expire the statistic summarization of the micro components. In short, SWEM may be considered as an EM clustering using Mahalanobis distance with fading window.

- GCPSOM [Smith and Alahakoon 2009]: There are two important extensions of SOM: Growing self-organizing map (GSOM)[Alahakoon et al. 2000] and cellular probabilistic self-organizing map (CPSOM) [Chow and Sitao 2004]. In GSOM, there is no need to pre-specify the size of the output map; it dynamically grows nodes at the boundary of the map whenever its accumulated error exceeds a threshold.

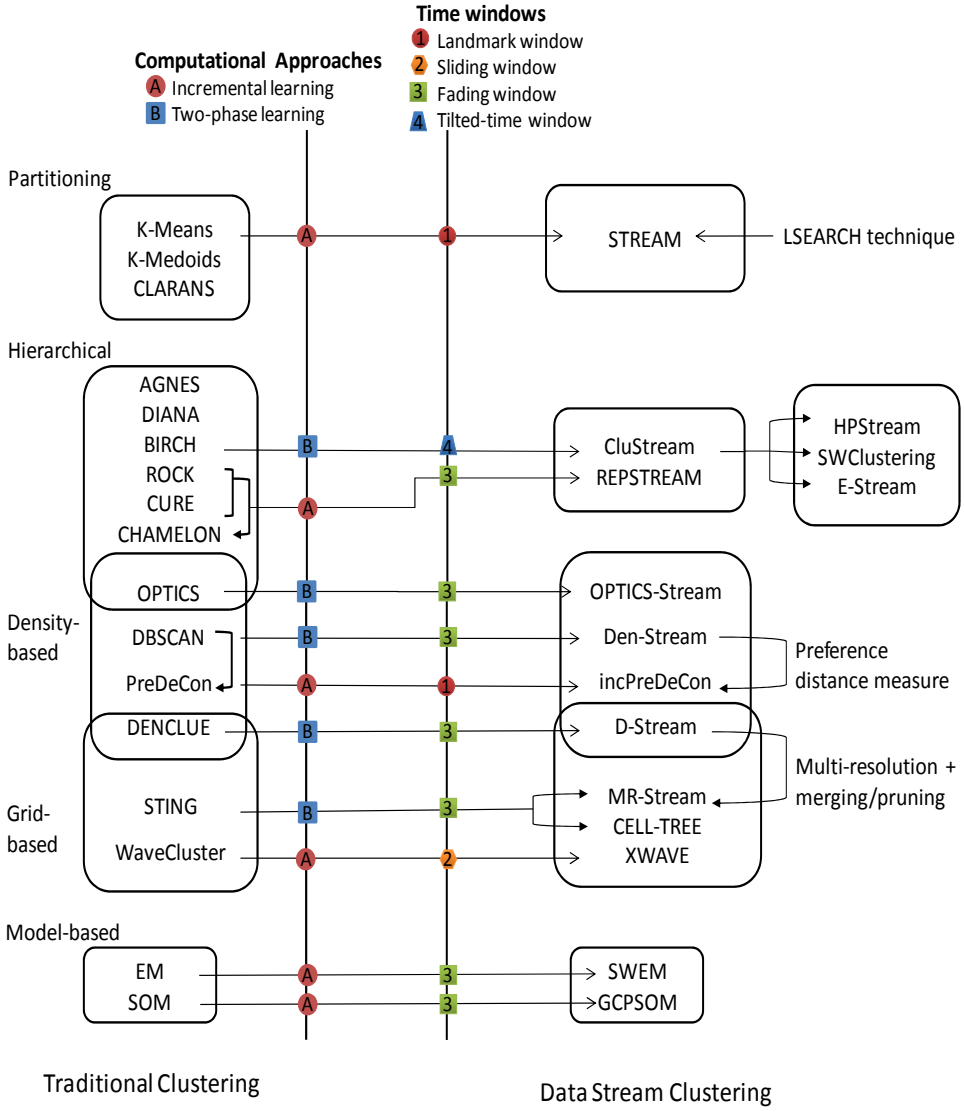

Fig. 3. The relationships between traditional clustering methods and stream clustering methods.

CPSOM is an online algorithm and is suitable for large datasets. CPSOM uses a fading window to reduce the weight of the neuron state. Thuss, CPSOM may forget old patterns and adapt to new patterns as they appear. GCPSOM is a hybrid algorithm that aggregates the advantages of both the GSOM and CPSOM. Therefore, GCPSOM dynamically grows the feature map for clustering data streams and keeps track clusters as they evolve.

3.3 Overall analysis

Previous sections have introduced many aspects of data stream mining, including constraints in data stream mining, time windows, and computational approaches. Many traditional and data stream clustering algorithms are also presented. Furthermore, we perform an overall analysis to summarize the above ideas and give readers a coherent view of data stream clustering. In Figure 3, we plot an intuitive diagram to outline the relationship between traditional clustering algorithms and data stream clustering algorithms. Traditional clustering methods are shown on the left if the diagram, and data stream clustering methods are on the right. Two schemata for categorizing data stream clustering methods are in the middle with two computational approaches and four time-windows.

Based on this diagram, we observe that most data stream clustering methods are adapted from traditional clustering methods but apply different computational approaches and time windows. For example, STREAM extends the k-means algorithms with incremental computational approach and landmark window. The key idea of STREAM is to apply the LSEARCH technique to perform k-means incrementally and hierarchically. CluStream extends the BIRCH algorithm and applies the two-phase learning approach and tilted-time window. HPStream improves CluStream to work for high dimensional data streams; SWClustering enhances CluStream on long-term running, and E-Stream extends CluStream to classify different types of concept drifts. REPSTREAM extends CHAMELEON algorithm with the incremental learning approach and fading window. Similarly, DenStream is an extension of DB-SCAN with the two-phase learning approach and fading window. incPreDeCon combines it with the preference distance measure to work with data streams. D-Stream inherits from DENCLUE; it has been extended into a multi-resolution approach with merging and splitting operations in MR-Stream. Moreover, CELL-TREE, XWAVE, SWEM, and GCPSOM are extensions of STING, WaveCluster, EM, and SOM respectively.

Although many clustering methods have been proposed, they are often unable to address all data stream mining constraints simultaneously, which are discussed in Section 2. For the concept-drift constraint, we establish two levels of how the algorithms respond to concept-drifts. The first level, *concept-drift adaptation*, means that an algorithm may update with new concepts and remove outdated concepts. Most algorithms deploy the time windows to update fresh information. Therefore, they satisfy this criterion. However, those applying the landmark window do not. The second level, *concept-drift classification*, indicates that an algorithm may detect and adapt properly to different types of concept-drifts. For example, with cluster shifting or recurrence, information about the cluster be collected as much as possible; thus, we should extend the sliding window or retrieve historical information of recurrence clusters. Another example with cluster appearance/disapperance, a clustering algorithm should remove the least informative clusters (the least weight cluster or the farthest cluster from the new one), not the oldest one. Furthermore, we evaluate whether these algorithms can work for high-dimensional data streams. Table II summarizes the capabilities of previously reviewed data stream clustering techniques. We observe that only HPStream and incPreDecon can work with high-dimensional data streams. E-Stream partly satisfies the concept-drift classification

Algorithm	Bounded Memory	Single-pass	Real-time Response	Concept-drift Adaptation	Concept-drift Classification	High-dimensional Data
STREAM [O'Callaghan et al. 2002]		√	√			
CluStream [Aggarwal et al. 2003]	√	√	√	√		
HPStream [Aggarwal et al. 2004]	√	√	√	√		√
SWClustering [Zhou et al. 2008]	√	√	√	√		
E-Stream [Udommanetanakit et al. 2007]	√	√	√	√	√	
RepStream [Lhr and Lazarescu 2009]		√		√		
OpticsStream [Tasoulis et al. 2007]	√	√	√	√		
Den-Stream [Cao et al. 2006]	√	√	√	√		
incPreDeCon [Kriegel et al. 2011]	√	√				√
D-Stream [Chen and Tu 2007]	√	√	√	√		
MR-Stream [Wan et al. 2009]	√	√	√	√		
Cell-Tree [Park and Lee 2004]	√	√		√		
Cell*Tree [Park and Lee 2007]	√	√	√	√		
XWAVE [Guha et al. 2004]	√	√	√	√		
SWEM [Dang et al. 2009]	√	√	√	√		
GCPSOM [Smith and Alahakoon 2009]	√	√		√		

Table II. Capabilities of data stream clustering algorithms.

constraint as it can distinguish different types of cluster evolutions; however, it does not consider high dimensionality.

Besides the tradeoffs of time-windows and computational approaches in Section 2, it is worthy to note that traditional and data stream clustering algorithms share similar advantages and limitations. Table III illustrates the trade-offs of each category of clustering techniques. For example, partitioning clustering algorithms are simple and relatively efficient; however, they need to specify the number of clusters, and are unable to discover non-spherical clusters. Grid-based clustering methods are quite fast; they are able to discover arbitrary-shape clusters. Nevertheless, their clustering quality depends on grid granularity, and they are unsuitable for high-dimensional data streams.

4. CLASSIFICATION

Classification is the process of finding a general model from past data to apply to new data. Classification is performed in two steps: Learning step and testing step. In the learning step, the system tries to learn a model from a collection of data objects, called the training set. In the testing step, the model is used to assign a class label for unlabeled data objects in the testing set. There are many classification techniques in the literature, such as the decision tree, Bayesian classification, rule-based classification, neural networks, support vector machines, k-nearest-neighbor and ensemble classifiers. In this section, we review several methods in traditional

Algorithm	Advantages	Limitations
Partitioning	Simple and relatively efficient. Terminate at a local optimum.	Need to specify the number of clusters. Unable to discover non-spherical clusters.
Hierarchical	Derive more meaningful cluster structures.	High complexity. Sensitive to the order of the data records.
Density-based	Can find arbitrary-shape clusters. Robust to noises.	Need many parameters: density and noise thresholds. Difficult to detect clusters with different densities.
Grid-based	Fast and can discover arbitrary-shape clusters. Robust to noises.	The clustering quality depends on the grid granularity. Unsuitable to high-dimensional data.
Model-based	Simple and can include domain knowledge.	Depends strongly on the assumed models.

Table III. Advantages and limitations of clustering approaches

and data stream classification. Then, we outline their mutual relationships and present an overall analysis of these algorithms.

4.1 Traditional Classification

Decision Tree:

The decision tree classifier is a popular data classification method. It aims to construct a decision tree where each interior node represents one attribute, and an edge to its child node corresponds to the possible values of its attribute. Each leaf stands for a group of data objects whose attribute values satisfy the specified ranges represented by the path from the root to the leaf.

Most decision trees are recursively constructed in a top-down manner with the greedy approach [Breiman et al. 1984; Quinlan 1986; 1993]. There are usually three basic steps to construct a decision tree: (i) select the best attribute to split, (ii) create a node and label it with the splitting attribute, and (iii) loop until certain conditions are met, such as all the training data belong to the same class or entropy threshold is exceeded.

Decision tree algorithms usually differ in how they select the splitting attributes and prune the tree when it becomes too large. There are heuristics to find the "best" splitting attributes, such as information gain in ID3 [Quinlan 1986], gain ratio in C4.5 [Quinlan 1993] (ID3's successor), and the Gini index [Breiman et al. 1984]. Tree pruning methods are used to prevent data over-fitting by removing redundant, least reliable branches. Thus, the pruned tree becomes smaller and simpler, and may classify testing data more quickly and accurately. Punning techniques can be applied; for example, pre-pruning to halt the tree's construction early; post-pruning to remove sub-trees from a "full grown" tree; and pessimistic pruning that uses error rate to decide when to stop the sub-tree pruning.

Decision trees are simple to understand and interpret. They are able to handle both numerical and categorical data, and perform well with large data. However, decision trees suffer from local optimal problems due to the use of heuristics; it may

create over-complex trees that do not generalize the data properly.

Bayesian Classification:

Bayesian classifiers are statistical classifiers that are based on Bayes' Theorem to predict class membership probabilities. Naive Bayes, an instance of Bayesian classifiers, is well-known for its simplicity and low computational cost. It assumes the class conditional independence described as follows:

Given a data object x with d attributes $x = (a_1, a_2, \ldots, a_d)$, x is assigned to the class having the highest posterior probability $P(c_i|x)$ computed as follows:

$$P(c_i|x) = \frac{P(x|c_i)P(c_i)}{P(x)} = \frac{P(c_i)}{P(x)}P(x|c_i) \quad (1)$$

$$= \frac{P(c_i)}{P(x)}\prod_{k=1}^{d} P(a_k|c_i),\ c_i \in \{c_1, c_2, c_3, \ldots, c_m\}, \quad (2)$$

where the values $P(a_k|c_i)$ and $P(c_i)$ are estimated from the training data. Naive Bayes works in an incremental manner in that it simply increases the relevant counts, and makes predictions based on posterior probabilities whenever it receives a new data object.

Neural Networks:

Neural networks consist of a set of input/output units where each connection has a modifiable weight. These weights are adjusted during the learning phase to predict the correct label of testing data. The central idea of neural network is to extract a nonlinear function by using linear combinations of the inputs as derived features. Therefore, the crucial problem of neural networks is to set the weights based on training data and desired output. Backpropagation algorithm is a common method of training neural networks that minimizes the mean squared error between the network's prediction and the actual value.

A *multilayer feed-forward neural network* (FFNN) that is based on the backpropagation algorithm has an input layer, one or more hidden layers and an output layer. All weights and biases in the feed-forward neural network are initialized with small random numbers. Then, each training instance passes through the network and produces a predicted value. The error between the predicted value and its actual value is propagated backward by updating the weights and biases accordingly. Neural networks are robust to noise, produce high accurate results and can be parallelized easily. However, they suffer long training times and require a number of parameters that are best determined empirically, such as the initialized values of weights and biases, the number of hidden layers, and learning rate. Poor result interpretation is another disadvantage of neural networks; this is typically mitigated by network pruning techniques.

Support Vector Machine (SVM):

Support Vector Machines was first introduced by Vapnik *et al.* in 1992 [Vapnik 1999]. It has become a principled and powerful classifier outperforming most other systems in many applications. SVM finds the maximum marginal hyperplane (MMH) to best separate training data of different classes. Training objects falling on the MMH are called *support vectors*. The problem of finding the MMH is known as a convex optimization and is solved by using Lagrangian formulation.

When data are linearly inseparable, SVM transforms data into a higher dimension, and finds a feasible linear separation. To reduce computational cost in the transformed space, *kernel tricks* are used to compute the equivalent dot products in the original input space without caring about the transforming function. SVM usually requires a long training time; however, it is highly accurate and less prone to over-fitting.

k-Nearest-Neighbor Classifier (k-NN):

k-NN classifier is a lazy learner that does not construct a general model. The k-NN classifier does less work during the training step and does more work during the testing step. It indexes training data and simply assigns testing data to the dominant class of their k nearest neighbors. The distance measure between two data instances can be diverse: Euclidean distance, Manhattan distance, Jaccard coefficient, etc. Although this algorithm is simple and robust to noises, it requires much space to store all training data and is computationally expensive, as it needs to compute and compare distances from testing data to all others. Hence, it should be speeded up by applying efficient storage structures and pre-sorting techniques. Liangxiao *et al.* introduced a good survey of improving k-NN [Liangxiao et al. 2007].

Ensemble Classifiers: Many classifiers have been proposed for inducing models from data. Each classifier has its own strengths and weaknesses. In theory, there is no specific algorithm that works well for different kinds of datasets. Developing a new algorithm that outperforms those existing classifiers in most kind of datasets is really hard or even infeasible. However, an easier way is integrating existing classifiers to create a more powerful classifier; this is called ensemble classifiers. Typically, building an ensemble consists of two steps: (1) construct a set of diverse classifiers and (2) combine their classification results. Various ensemble classifiers methods have been proposed, and they are principally different from the above two steps. Here, we only present three basic ensemble methods: Bagging, Boosting and AdaBoost.

—*Bagging*: Bagging (bootstrap aggregating) was introduced by Breiman. In Bagging, each classifier is trained on a different bootstrap replicate of the training set. A bootstrap replication is generated by random sampling with replacement from the original dataset. Bagging uses voting method to combine its members' classification results.

—*Boosting*: Boosting aggregates many weak learners into a strong ensemble by filtering the training data. Data objects that are misclassified or disagree on previous classifiers are treated as training objects for the next classifier. The original boosting works as follows:
  —Select a random sample without replacement and train a learner C_1.
  —Train weak learner C_2 with filtered examples, half of which are misclassified by C_1.
  —Train weak learner C_3 only with examples that C_1 and C_2 disagree on
  —Combine the predictions of C_1, C_2 and C_3 by majority voting.

—*AdaBoost*: AdaBoost, short for Adaptive Boosting, is the most well-known and successful boosting algorithm in practice. It adapts to the errors of weak classifiers and set the weight for each classifier instead of majority voting. Adaboost

re-weighs the samples in a manner that wrongly classified samples retain their weights while correctly classified samples relatively decrease their weights by $\frac{\epsilon}{1-\epsilon}$, where ϵ is the error rate of the previous classifier. To combine classification results, Adaboost sets weights to classifier members according to their error rates, $\alpha = log\frac{\epsilon}{1-\epsilon}$. AdaBoost algorithm reduces error on the training set to zero with a number of iterations. Currently, it is still unclear about AdaBoost's capability to generalize testing data, which is not included in the training data.

4.2 Data Stream Classification

Hoeffding Tree: Hoeffding tree is a decision tree classifier for data streams [Domingos and Hulten 2000]. Traditional decision trees need to scan the training data many times to select the splitting attribute. However, this requirement is infeasible in the data stream environment. To overcome this limitation, the Hoeffding bound is used to choose an optimal splitting attribute within a sufficient amount of receiving data objects. Given N independent observations of a random variable r with range R and computed mean $\overline{r}$, the Hoeffding bound guarantees that the true mean of r is at least $\overline{r} - \epsilon$ with probability $1 - \delta$, where δ is a user-specified parameter.

$$P(E[r] \geq (\overline{r} - \epsilon)) \geq 1 - \delta,\ \epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}}$$

Let $G(X_i)$ be a heuristic measure to select a splitting attribute. After receiving N observations, X_a and X_b are the best and the second best splitting attribute. In case of Hoeffding tree, the variable r is now considered as $r = \Delta G = G(X_a) - G(X_b)$. If $\overline{r} = \Delta\overline{G} = \overline{G}(X_a) - \overline{G}(X_b) > \epsilon$, where ϵ is computed from the above equation, we say that this difference is larger than $(\overline{r} - \epsilon) > 0$ with confidence $1 - \delta$. Then, the attribute X_a is selected to build the tree.

Hoeffding tree algorithm is an incremental algorithm, which satisfies the single-pass constraint of data stream mining. For each new arriving data, Hoeffding tree algorithm use Hoeffding bounds to check whether the best splitting attribute is confident enough to create the next level tree node.

Hoeffding tree algorithm has high accuracy and works well with large datasets. However, it is not able to handle concept drifts in data streams as no node can be changed once created. CVFDT [Hulten et al. 2001] is an extension of the Hoeffding tree to address concept drifts in data streams. CVFDT maintains sufficient statistics at every tree node to monitor the validity of its previous decisions. When data come, it continually updates the statistics stored in tree nodes. Using the sliding window, CVFDT removes the effect of outdated data by decreasing the corresponding statistics at the tree nodes. It periodically scans the tree nodes to detect concept drifts. If concept drift appears, CVFDT concurrently grows alternative branches with the new best attribute and removes the old branches with alternative branches if it becomes less accurate.

Bayesian Belief Network: Seidl *et al.* proposed a novel index-based classifier called Bayes tree [Seidl et al. 2009]. Adapted from the R*-tree, Bayes tree generates a hierarchical Gaussian-mixture tree to represent the entire dataset. Each tree node contains statistics of the data objects within, including a minimum bounding rectangle, the number of data objects, linear sum, and quadratic sum of all data

objects.

To solve a multi-labeled classification problem, a single Bayes tree is constructed for each class. For each testing data object $\mathbf{x}$, the algorithm tries to find a set of prefix-closed nodes E_i, called frontier, in every Bayes tree. The probability that $\mathbf{x}$ belongs to class c_i is computed as follows:

$$P(c_i|x) = (\sum_{e_s \in E_i} \frac{n_{e_s}}{n} g(x, \mu_{e_s}, \sigma_{e_s})) * P(c_i)/P(x),$$

where e_s is a tree node in the frontier set E_i; n_{e_s}, μ_{e_s} and σ_{e_s} are respectively the number of data object, the center and the deviation of tree node e_s. Testing object x is labeled with the class having the maximum probability. Bayes tree is an anytime classifier that can provide a decision after a very short initialization, and later provide more accurate decisions when more time is available by selecting the more precise frontier.

Bayes tree is later extended to the MC-tree [Kranen et al. 2010] by Kranen *et al.*. The MC-tree combines all classes in a multi-Gaussian tree and needs only one step to refine all class models simultaneously rather than $|C|$ steps in Bayes tree, where $|C|$ is the number of classes. Moreover, MC-Tree optimizes tree construction by using multi-dimensional scaling (MDS) [de Leeuw 2005] to transform the data space. It is claimed to outperform Bayes tree with higher accuracy of up to 15%.

Neural Network: Leite *et al.* propose an evolving granular neural network (eGNN) supported by granule-based learning algorithms to classify data streams. There are two phases in eGNN. In the first phase, eGNN use T-S neurons to construct information granules of incoming data. Then, the neural network is built on the information granules rather than the original data in the second phase. A granule associated with a class label is defined by triangular membership functions, which are later evolved to accommodate new data. The weights are decreased by a decay constant; this process helps to reduce the degree of importance of outdated granules. Basically, eGNN uses class exemplars in the form of information granules to perform classification tasks. When testing data comes, a max-neuron selects the best-fit granules and assigns their labels as the prediction labels of the testing data. eGNN is able to tackle classification problems in continuously changing environments. However, the requirement of long training time is still a cost that limits the ability of eGNN to work with a massive dataset. Therefore, only small datasets are used to evaluate the performance of eGNN in the experiment section of this paper. The authors later extend this work to a more general and efficient semi-supervised approaches [Leite et al. 2010] to work with partially labeled dataset.

Support Vector Machines (SVMs) have shown its prominent performance in many machine learning problems with static datasets. However, it is very expensive to use SMVs in large scale application due to its time complexity $O(N^3)$ and memory complexity $O(N^2)$, where N is the number of data objects. To work with a very large dataset, Tsang *et al.* proposed the Core Vector Machine (CVM) algorithm that uses Minimum Enclosing Ball (MEB) to reduce its complexity [Tsang et al. 2007]. A MEB is a hyper-sphere that represents the set of data objects inside it. The algorithm first finds a representative MEB set that is a good approximation of the original dataset. Then, the optimization problem of finding the maximum margin is directly performed on this MEB set.

Rai *et al.* propose StreamSVM [Rai et al. 2009], an extension of CVM with a single scan, to work with data streams. In StreamSVM, a MEB has a flexible radius that is increased whenever a new training is added. The algorithm is competitive by providing an approximation result to the optimal one. However, StreamSVM is still unable to detect concept drifts in data streams.

k-Nearest-Neighbor Classifier: On-Demand-Stream is a k-NN data stream classifier that extends the CluStream method [Aggarwal et al. 2006]. It inherits most of the good features in CluStream such as the micro-cluster structure, the tilted-time window, and the online-offline approach. A micro-cluster in On-Demand-Stream is extended with a class label, and it only takes data objects with the same class label. Its offline classification process starts to find the best window of data objects, called the best time horizon. These micro-clusters in the best time horizon are extracted using the addition property of micro-clusters. On-Demand-Stream perform 1-NN classification by assigning a testing data object to the label of the closest micro-clusters.

Inspired by M-tree [Ciaccia et al. 1997], Zhang *et al.* [Zhang et al. 2011] propose a Lazy-tree structure to index micro clusters (which are called exemplars in the paper). This helps to significantly reduce k-NN's classification time from $O(N)$ to $O(log(N))$, where N is the total number of exemplars in the Lazy-tree. The tree consists of three main operations: search, insertion and deletion operations. The insertion and deletion operations add new nodes and remove outdated nodes in a way that guarantees that the tree is balanced. The search operation is used to classify testing data. A *branch-and-bound* technique based on the triangle inequality filters unnecessary checking exemplars; therefore, it minimizes the number of comparison and accelerates the searching operation.

Ensemble Classifiers: Bagging and Boosting have shown their superior through extensive experiments on traditional dataset. Therefore, many researchers have tried to adapt the methods to work on data streams.

- *Online Bagging & Boosting*: Oza *et al.* proposed the *Online Bagging & Boosting*, which is one of the first work of adapting traditional bagging and boosting [Oza 2005]. From a statistical view, each training data object appears k times in training datasets of classifier members with the probability $P(k) = \binom{N}{k}\left(\frac{1}{N}\right)^k\left(1-\frac{1}{N}\right)^{N-k}$, where k is the size of training set and N is the size of dataset. On data streams, we can assume that the number of data objects is unlimited, $N \to \infty$. Therefore, the probability $P(k)$ tends to a $Poisson(1)$ distribution, where $Poisson(1) = exp(-1)/k!$. Within this observation, Oza *et al.* proposed Online Bagging to assign each data object a weight according to $Poisson(1)$ distribution, which is considered as a replacement sampling method. In Online Boosting, the weights of coming data objects and classifier members are adjusted according to the error rates of classifier members at the current time.

- *Weighted Ensemble Classifiers*: With the observation that the expiration of old data should rely on data distribution instead of their arrival time, Wang *et al.* proposed an ensemble classifier with weighting method for mining concept-drifting in data streams [Wang et al. 2003]. The algorithm constructs and maintains a fix k number of classifiers, which can be C4.5, RIPPER, or naive Bayesian. Processing

in a batch-mode manner, it use each new chunk of coming data objects to train a new classifier. Then, the ensemble is formed by selecting the k most accurate classifiers, and the weight of each classifier is set according to its accuracy. This method is stated to be better than a single data stream classifier, such as VFDT, CVFDT. However, it is quite sensitive to the chunk size and the number of classifier members k.

In real applications, data stream may contain noise where data instance may be mislabeled or have erroneous values. Zhang *et al.* proposed an *aggregated ensemble* algorithm to tackle the problem of learning from noisy data stream [Zhang et al. 2009; Zhang et al. 2011]. This approach is a combination of horizontal and vertical ensemble frameworks. The horizontal framework builds a different classifier on each data chunk, while the vertical framework builds different classifiers on the up-to-date data chunk with different learning algorithms. The horizontal framework is robust to noise and can reuse historical information; however, it is unsuitable with sudden drifts, where the concepts of data stream change dramatically. On the other hand, the vertical frame work can produce good results even in case of sudden drifts; nevertheless, it is sensitive to noise. Building classifiers on different data chunks using different learning algorithms, the aggregated ensemble constitutes a Classifier Matrix. The average weighting method is used on this matrix to predict the label for testing data. The author theoretically proved that the aggregate ensemble has less or equal mean squared error of the vertical and horizontal frameworks in average. However, this framework suffers high time complexity.

Nguyen *et al.* [Nguyen et al. 2012] addressed the problem of learning from high-dimensional data streams, where only a small subset of data features are important for learning process. Moreover, in this context, the definition of relevant features is temporary and limited to a certain period of time. Informative features may become irrelevant afterward, and previously insignificant features may become important features. Proposing a definition of feature drifts, i.e., a change in the set of important features, the authors combined this concept with a weighted ensemble classifier to tackle this problem. A multivariate feature selection method [Lei and Huan 2003] is adapted with a sliding window technique to detect feature drifts. Then, an ensemble learner consists of selected online learners is constructed with an optimal weighting method. When a gradual drift occurs, classifier members of the ensemble are updated and theirs weights are also adjusted according to their error rates. When a feature drift occurs, the ensemble replaces an old-fashioned learner with a updated classifier, which is trained with a new set of important features. Experiment results show that the algorithm is effective and efficient with high-dimensional data streams.

- *Adapted One-vs-All Decision Trees (OVA)* [Sattar et al. 2009]: OVA is a recent ensemble method for data streams. It learns k binary CVFDT classifiers, and each classifier is trained to classify instances between a specified class and all remaining classes. To classify a new data object, each classifier is run and the one with the highest confidence is returned. The confident of the classification is set as the proportion of the dominant class at the leaf where the testing object has reached. To archive a high accuracy, OVA aims to construct an ensemble of CVFDT classifiers with a low error correlation and a high diversity. Moreover, OVA quickly adapts to

concept drifts as it only needs to update two component classifiers related to the evolving class, and can work well with imbalance data streams.

- *Meta-knowledge Ensemble* [Zhang et al. 2011]: The high complexity of ensemble learning limits it to be applicable to many time-critical data stream applications in the real world. Zhang *et al.* proposed a meta-knowledge ensemble algorithm that selects the best suit classifier for testing data. An Ensemble-tree (E-tree) is constructed to organize base classifiers, each of which occupies a weight and a closed space in the whole data space. Similar to R-tree [Guttman 1984], the E-tree has three key operations, including search, insertion and deletion operation. Hence, it is height-balanced and guarantees a logarithmic time complexity for prediction. The insert operation is used to integrate a new classifier into the ensemble. When the number of classifiers in a node exceeds a predefined value, the node is split into two nodes with a principle that the covering area of the two nodes should be minimized. The deletion operation removes outdated classifier when E-tree reaches its capacity, the tree may need to be re-organize to guarantee its balance. The search operation is used to classify a testing instance x. Classifiers whose close space contains x are invoked, a weighted voting method is applied to decide a class label for x.

4.3 Overall analysis

After presenting many data stream classifiers, we introduce an overall analysis to summarize all related issues and give readers a broader view of data stream classification. Figure 4 shows the relationship between traditional and data stream classifiers. Traditional classifiers are on the left; data stream classifiers are on the right. To categorize data stream classifier, the two schemata, computational approaches and time windows, are in the middle.

Similarly, we observe that data stream classifiers are inherited from traditional classifiers and apply different computational approaches and time windows. For example, VFDT is an extension of the decision trees for data streams. VFDT uses Hoeffding bound to create a tree node when having a sufficient amount of data. It follows the incremental learning approach and land-mark window. CVFDT is an enhanced version of VFDT that can adapt to concept-drift by constructing alternative trees. Bayes tree is an extension of Bayesian classifier with the two-phase learning approach and land-mark window. MC-Tree improves the Bayes tree with multi-label nodes. eGNN is a neural network algorithm that designed to work for data streams. CVM, derived from SVM classifier, follows the two-phase learning approach and fading window. StreamSVM extends CVM by making the radius of minimum enclosing balls flexible. On-Demand, an extension of k-NN classifier, applies the two-phase learning approach and tilted-time window. Lazy-Tree is another improved version of k-NN classifier with the two-phase learning approach and sliding window. Many ensemble classifiers are designed to work for data streams. Online Bagging & Boosting are extensions of traditional Bagging & Boosting with incremental learning approach and sliding window. There are many weighted ensemble classifiers with different weighting strategies. For example, aggregated ensemble uses average weighting method (voting); while HEFT-Stream, WCE, OVA, and Ensemble-tree apply weighting method based on the accuracy of classifier members. They follow the incremental learning approach, and can be loosely considered of applying sliding window as they typically remove the least

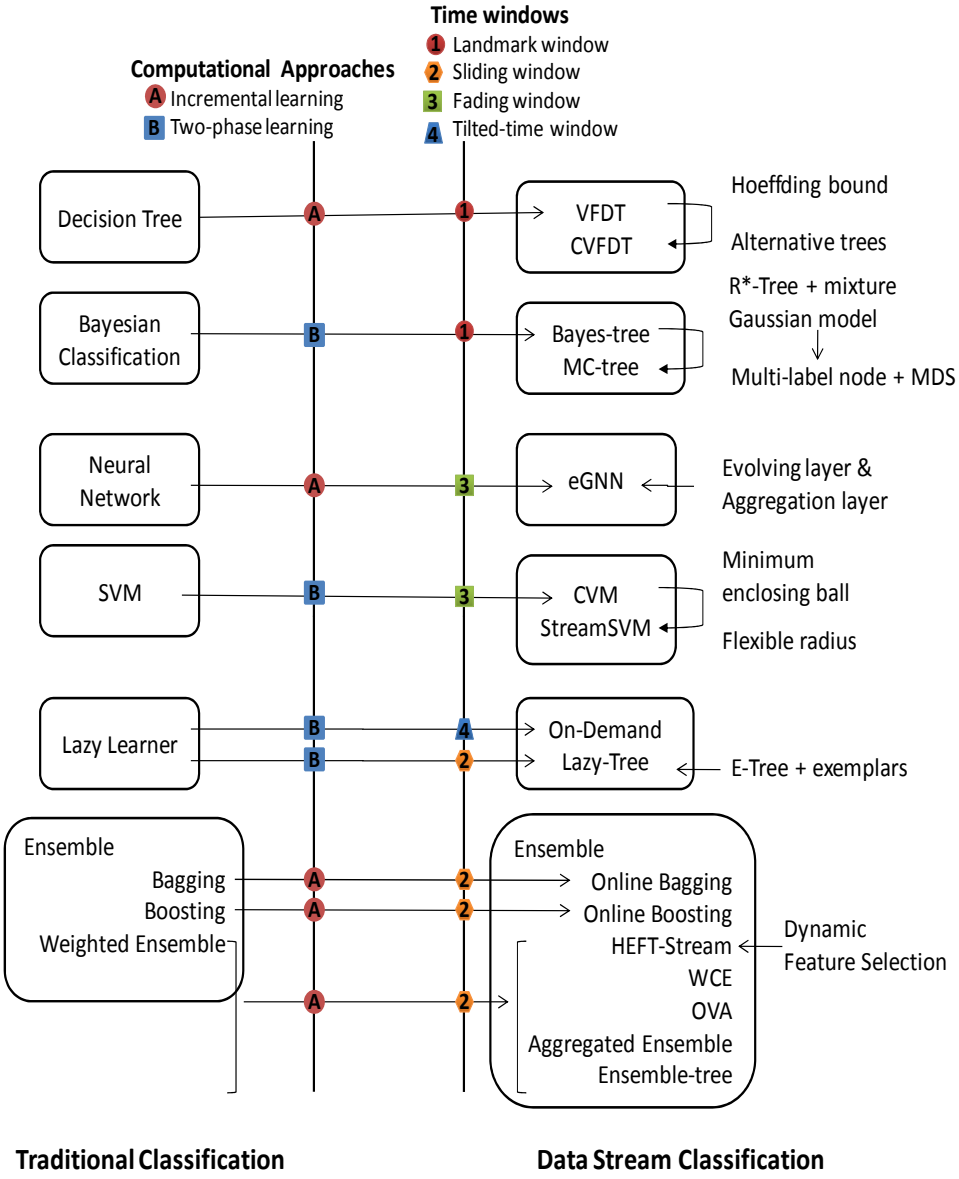

Fig. 4. The relationships between traditional classification methods and stream classification methods.

accurate clarifier member.

Moreover, we evaluate capabilities of data stream classifiers into the Table IV. Although many classifiers have been proposed, they are typically unable to satisfy all constraints of data stream mining at the same time, which are discussed in Section 2. Similarly, we propose two levels of concept-drift constraint, concept-drift adaptation and concept-drift classification. We also assess whether these classifiers

Algorithm	Bounded Memory	Single-pass	Real-time Response	Concept-drift Adaptation	Concept-drift Classification	High-dimensional Data
VFDT [Domingos and Hulten 2000]	√	√	√			√
CVFDT [Hulten et al. 2001]	√	√	√	√		√
Bayes tree [Seidl et al. 2009]	√	√	√	√		
MC-tree [Kranen et al. 2010]	√	√	√	√		
eGNN [Leite et al. 2009]	√	√		√		
CVM [Tsang et al. 2007]	√	√		√		
StreamSVM [Rai et al. 2009]	√	√		√		
On-Demand [Aggarwal et al. 2006]	√	√	√	√		
Lazy-Tree [Zhang et al. 2011]	√	√	√	√		
Online Bagging & Boosting [Oza 2005]	√	√	√			
WCE [Wang et al. 2003]	√	√	√	√		√
OVA [Sattar et al. 2009]	√	√	√	√		√
HEFT-Stream [Nguyen et al. 2012]	√	√	√	√	√	√
Aggregated Ensemble [Zhang et al. 2011]	√	√	√	√		
Ensemble-tree [Zhang et al. 2011]	√	√	√	√		√

Table IV. Capabilities of data stream classification algorithms.

can work for high-dimensional data streams. We observe that all classifiers satisfy the bounded-memory and single pass constraints. eGNN, CVM, and StreamSVM cannot response in a real-time manner, as they require much time for training process. HEFT-Stream can distinguish two kinds of concept-drifts (gradual drifts and feature drifts), and adapts properly to them. VFDT, CVFDT, WCE, OVA, and Ensemble-tree can work for high-dimensional data streams, as they deploy the decision tree as their primitive classifiers.

When a classifier follows a computational approach and time-window, it will have some specific tradeoffs, which is discussed in Section 2. Moreover, traditional and data stream classifiers also have common advantages and limitations in Table V. For example, decision trees are easy to understand, robust to noise, efficient, and can remove redundant attributes; however, it suffers an over-fitting problem when the tree has many levels and the classification rules become difficult to interpret. The lazy-learner can be implemented easily, but it consumes much memory and is susceptible to high-dimensional data streams. Ensemble is highly accurate and easy to implement; however, it is mostly based on heuristics and lacks of solid theory.

5. FUTURE RESEARCH

Although significant data stream research has been performed for more than a decade, there are still many research issues that need to be further explored. The

Algorithm	Advantages	Limitations
Decision Tree	Easy to understand Robust to noises Low computational cost, even for very large training datasets Can deal with redundant attributes.	Suffer an over-fitting problem when the tree has many levels and the rules become difficult to comprehend.
Bayesian Classifier	Simple, efficient, effective and robust to noisy data	Independence assumption is often violated in the real world.
Neural Network	Well-generalizing capability. Can solve dynamic or non-linear problem.	Inability to interpret the learned model (black-box). High complexity.
SVM	Provide a unique solution. Can solve non-linear problem with implicit kernel. Work well even when training sample are bias.	Depend on the choice of the kernel. High complexity. Difficult to design multi-class SVM classifiers.
Lazy Learner: k-NN	Easy to implement.	Large storage requirements. High complexity. Highly susceptible to high-dimensional data.
Ensemble	High accuracy. Easy to implement.	Based on heuristics and lack of solid theory.

Table V. Advantages and limitations of classification algorithms

most important topics are described follow:

5.1 Dynamic Feature Selection

In high-dimensional dataset, not all data attributes (features) are important to the learning process. Figure 5(a) shows three common types of features: (i) irrelevant features, (ii) relevant but redundant features, and (iii) relevant and non-redundant features. The critical task of dimension reduction techniques is to extract the set of relevant and non-redundant features so that the learning process is more meaningful and faster. Moreover, to reduce the number of features, an algorithm may select a subset of the full feature space (called feature selection), or derive a new feature by combining existing features (called feature extraction). Feature selection is preferable to feature extraction, as it preserves the meaning of selected features while the latter usually creates hard-to-understand features. Moreover, feature selection has a limited feature search space $|2^d|$, where d is the number of features. Feature extraction has unlimited search space; however, its performance is usually better than feature selection. Figure 5(b) illustrates some approaches for each type of dimension reduction algorithms. For example, feature selection includes projected, subspace, and hybrid approaches. Feature extraction consists of PCA-based, fractal-based, and random sampling approaches [Kriegel et al. 2009].

In the literature, feature selection techniques can be classified into three categories: filter, wrapper and embedded models [Liu and Yu 2005]. The filter model applies an independent measure to evaluate a feature subset; thus, it only relies on the general characteristics of data. The wrapper model runs together with a learning algorithm and uses its performance to evaluate a feature subset. A hybrid

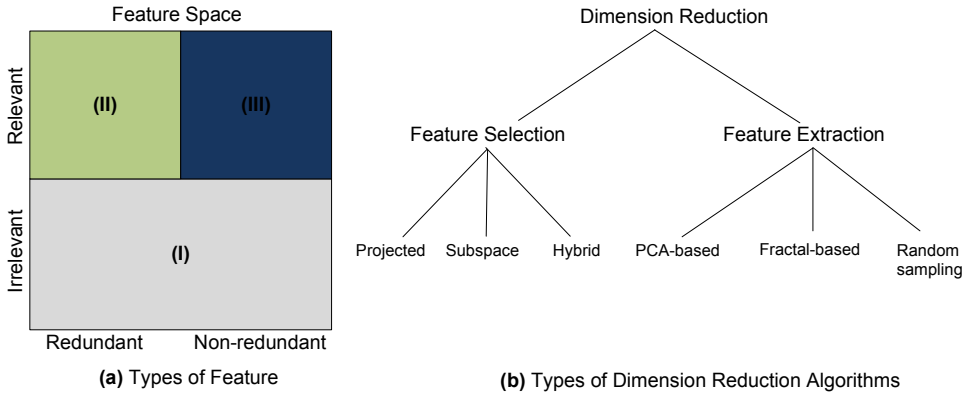

Fig. 5. Overview of Feature Selection

Approach	Advantages	Limitations
Filter	Fast Scalable Independce of the classifier	Ignores interaction with the classifier
Wrapper	Interact with the classifier Can find the most "useful" features toward the classifier	High computational cost Risk of overfitting Classifier dependent selection
Hybrid (Filter+Wrapper)	Interact with the classifier Better computational complexity than wrapper methods Are prone to over-fitting	Classifier dependent selection

Table VI. Advantages and limitations of feature selection models.

model takes advantage of the above two models. Advantages and limitations of the three models are elaborated in Table VI. Furthermore, the importance of a feature evolves in data streams and is restricted to a certain period of time. Features that are previously considered as informative may become irrelevant, and vice-versa, those rejected features may become important features in the future. Thus, *dynamic feature selection* techniques are required to monitor the evolution of features.

After investigating many clustering and classification algorithms, we observe that only a few algorithms work with high-dimensional data streams, and, they have many limitations. For clustering, HPStream [Aggarwal et al. 2004] applies the projected approach, and only removes irrelevant features but no redundant features. It requires a predefined value of the average dimensional degree, and is just suitable for spherical-shape cluster. Similarly, IncPreDeCon [Kriegel et al. 2011] only takes out irrelevant features and suffers from high complexity. For classification, some classifiers work with high-dimensional data streams as they are simply derived from or deployed decision tree as primitive classifiers [Domingos and Hulten 2000; Hulten et al. 2001; Wang et al. 2003; Sattar et al. 2009; Zhang et al. 2011]. In this scenario, they are considered to be deploying a filter model that uses decision tree's

informative measures (e.g., information gain) to select relevant features. Hence, these algorithms strictly use decision tree as primitive classifiers, and are not able to remove redundant features. HEFT-Stream [Nguyen et al. 2012] is the only ensemble classifier that removes both irrelevant and redundant features, and that works with any type of classifier. However, HEFT-Stream still uses the filter model, which is independent of classifier members. Therefore, the problem of dynamic feature selection in data streams is open and requires further exploration. A dimension reduction technique that follows the hybrid model, captures features' evolution dynamically, and removes both irrelevant and redundant features is much expected.

5.2 Incorporating Temporal Order

Spiliopoulou *et al.* [Spiliopoulou et al. 2006] proposed the MONIC algorithm to model and detect changes in clusters. Matching clusters by checking their overlaps, they defined two types of cluster transitions: internal transitions (e.g., cluster shrinks, cluster expands, cluster becomes compacter or diffuser, cluster shifts) and external transitions (e.g., cluster survives, cluster is split into many clusters, cluster is merged with other cluster, cluster disappears, and cluster emerges). MONIC has the advantage of independence from any clustering algorithm; however, it is unsuitable for data streams. It requires time to post-process clustering results; thus, cannot show relationships among clusters in a real-time manner.

C-TREND [Adomavicius and Bockstedt 2008] constructs a temporal cluster graph to capture temporal order among clusters. In this graph, each node represents for a cluster and is labeled with the size of the cluster. Edges connect adjacent node and are labeled with a distance value (similarity) between two nodes.Althought C-TREND is able to visualize and detect trends in multi-attribute temporal data, it suffers from many limitations. First, C-TREND is not suitable for data streams as it requires a predefined number of clusters. C-TREND is also computationally expensive as it deploys hierarchical clustering and constructs a dendrogram as a preprocessing step. Moreover, it only considers transitions among clusters while other internal cluster transition are ignored.

TRACDS [Hahsler and Dunham 2011] takes advantage of both MONIC and C-TREND. It presents a technique that is independent of clustering algorithms and that records transitions among clusters into a transition-count matrix that represents a Markov Chain directed graph. Unlike MONIC that performs post-process clustering results to detect cluster transitions, TRACDS constructs the transition-count matrix incrementally whenever each data instance arrives and is assigned to a specified cluster. Therefore, TRACDS only needs a very light-weight interface, and is able to work for data streams.

The above research work tracks evolution of clusters; however, they typically ignore the historical clustering that is important to producing better clustering results. *Evolutionary clustering* that considers this research issue has attracted much research work recently. It focuses on processing time-stamp data to produce a sequence of clustering, each of which is a time-step of the system. Evolutionary clustering should optimize the trade-off between preserving the faithfulness of current data as much as possible, while preventing dramatic shifts from historical clustering to the next step. Chakrabarti *et al.* [Chakrabarti et al. 2006] first introduced the problem and proposed a framework for evolutionary clustering by

embedding a smoothing parameter to penalize the important of historical clusters, as follows:

$$\sum_{t=1}^{T} sq(C_t, M_t) - cp. \sum_{t=2}^{T} hc(C_{t-1}, C_t),$$

where $sq(C_t, M_t)$ is a snapshot quality of the clustering C_t at time t with respect to a relationship matrix M_t, $hc(C_{t-1}, C_t)$ is a history cost between C_t and C_{t-1}, and cp is the smoothing parameter. Most evolutionary clusterings differ from snapshot quality and history cost functions. The authors deployed agglomerative hierarchical clustering and k-means as examples of the framework. In hierarchical clustering, snapshot quality is the quality of all merges performed to create C_t; history cost is the distance between two trees. In k-means clustering, snapshot quality is the sum of distance between each data instance and its center; history cost is the sum of distance between each pairing clusters between time t and $t-1$. Extending this idea, Chi *et al.* proposed two frameworks for evolutionary spectral clustering, called Preserving Cluster Quality (PCQ) and Preserving Cluster Membership (PCM) [Chi et al. 2007; 2009]. PCQ penalizes clustering results that disagree with past similarities, while PCM penalizes clustering results that disagree with past clustering results.

Visualizing clusters over time has also attracted much research in both data stream mining and evolutionary clustering. OPTICS-Stream [Tasoulis et al. 2007] is one of the first work that attempts to visualize the temporal order of clusters. Inheriting the OPTICS algorithm [Ankerst et al. 1999], OPTICS-Stream uses micro-cluster structure to build a synopsis of data streams, and further orders reachability-distance among micro-clusters. It produces a 3-D reachability plot that can be used to visualize the clustering structure and structure changes in data streams. However, this plot fails to provide a clear view of borders among clusters and cannot extract external transitions clusters, such as splitting or merging clusters.

Lin *et al.* [Lin et al. 2010] proposed ContexTour framework for visualizing and exploring multiple dimensions of community activities in social networks. It includes users, community-generated content, relevant topic, as well as their evolutions. The framework has two components, Dynamic Relational Clustering (DRC) and Dynamic Network Contour-map (DNC). The DRC is an evolutionary clustering method that efficiently tracks and finds smoothly evolving soft clusters that best fit the community evolution. The DNC visualizes the results of DRC. It first selects the most important entities that have high conditional probability given a cluster; then the cosine similarity between those entities is computed. The contour-map layout is constructed to depict the density distribution of all entities. The DBLP dataset is examined to evaluate the performance of ContexTour.

To better understand evolving topics in text data, Cui *et al.* [Cui et al. 2011] proposed TextFlow, an interactive visual analysis tool for analyzing evolution patterns from multiple topics. The core component of TextFlow is a three-level Directed Acyclic Graph (DAG) whose first-level corresponds to the topic flow, the second layer encodes keyword bundles (critical events), and the third level represents keyword thread. First, text documents are clustered so that topics represented by document clusters and their connections are extracted. The Hierarchical Dirichlet Processes [Teh et al. 2006] is used to trace spitting/merging events between topics. These critical events are likely to involve intense keyword changes as a topic is

summarized by keywords. Acknowledging the limitation of stack-based graph [Dork et al. 2010] that cannot well present topic merging and splitting, TextFlow applies a river-flow-based visualization that help users understand and analyze the evolution topics easily. The publication and Bing news datasets are used to demonstrate this framework.

Evolutionary clustering and data stream clustering have a close relationship. Working with large amounts of high rate data, data stream clustering much focuses on single-pass and scalability issues. While evolutionary clustering aims to obtain cluster that evolve smoothly over time, most evolutionary clusterings are spectral clustering algorithms that require matrix and graph processing operations; thus, they are usually computationally expensive. Therefore, developing an evolutionary clustering for data stream remains a challenges. Moreover, any clustering algorithm should support user-friendly visualization so that users may easily explore transitions among clusters and understand the major reasons that trigger these evolutionary transitions.

5.3 Other Research Directions

—*Concept drift* is one of the basic challenges in data stream mining. As mentioned in Section 2.1, a concept drift can be defined as a tuple of prior probabilities $P(c_i)$ and class-conditional probabilities $P(x_t|c_i)$. By monitoring these probabilities, researchers try to categorize different types of concept drifts, such as, gradual drifts, sudden drifts, incremental drifts, reoccurring drifts [Minku et al. 2010; Zhang et al. 2008; Zliobaite 2010]. Although many adaptive learners have been proposed to deal with concept drifts, none of them is able to detect all types of concept drifts. This limitation leads us to the following research questions: How can we measure the degree of concept drifts so that it is possible to detect all kinds of concept drifts? Also, how can a classifier properly adapt to different types of concept-drifts?

—*Cloud computing* has recently become a highly disruptive technology. It aims to provide computation, data access, software and storage services that are available on demand. Thus, cloud computing is expected to facilitate data stream mining soon. However, the privacy issue is one of the key concerns of cloud computing. Although cryptography may solve the privacy preservation problem, its high complexity is still a cost. Thus, deploying data stream mining on cloud computing is still a challenge and requires more research efforts.

—*Multiple Data Streams (MDS)* has recently been used in many applications where many local data sources are located at different locations and have different protocols such as sampling rates, data formats, data processing models, distribution time and so on. Therefore, synchronizing and merging data is more significant in MDS mining. Also, the relationships among data sources is still unclear. Currently, MDS mining is still in its infancy stage with lack of formal definitions and thorough surveys.

6. CONCLUSIONS

Recently, data streams have become popular in many real-life applications where intensive data can be generated frequently. Research in data stream mining has

led to a large variety of clustering and classification algorithms in the past decade. However, each algorithm has its own advantages and limitations, and is not suitable for all kinds of problems and arbitrary datasets. Given a research problem with a specified domain, choosing an appropriate algorithm becomes a critical task as this requires background knowledge of the basic principles and systematic approaches in the research field. This survey aims to provide a coherent analysis of the state-of-the-art data stream clustering and classification algorithms.

First, we outline the difference between traditional data mining and data stream mining. We identify some constraints and propose a general model for data stream mining. Various computational approaches and time windows together with their tradeoffs are presented.

Then, we discuss in detail various state-of-the-art algorithms in both traditional and data stream clustering in Section 3. Their relationships are visualized in Figure 3, which gives readers an intuitive understanding on how data stream clustering is inherited from traditional clustering. Furthermore, we examine their capabilities on satisfying constraints of data stream mining in Table II. Advantages and limitations of clustering categories are also investigated in Table III. Similarly, critical analysis is applied to data stream classification in Section 4. The relationship between traditional and data stream classifier is illustrated in Figure 4. Their capabilities are evaluated in Table IV, and the tradeoffs of classifier categories are presented in Table V.

Finally, we explore future research directions. Dynamic feature selection should attract more attention in high-dimensional data streams such as biological or text applications. Moreover, incorporating temporal order gives much help on tracking evolutions on data stream mining. It should come together with a visualization toolkit so that users can easily explore and understand the evolutions. Other research directions, e.g., measuring the degree of concept-drifts, deploying data streaming on cloud computing, and multiple data stream are interesting areas for future research.

REFERENCES

Adomavicius, G. and Bockstedt, J. 2008. C-trend: Temporal cluster graphs for identifying and visualizing trends in multiattribute transactional data. *IEEE Transactions on Knowledge and Data Engineering*, 721–735.

Aggarwal, C. C. 2009. Data streams: An overview and scientific applications. In *Scientific Data Mining and Knowledge Discovery*, M. M. Gaber, Ed. Springer Berlin Heidelberg, 377–397.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases*. Vol. 29. VLDB Endowment, 81–92.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. 2004. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*. VLDB Endowment, Toronto, Canada, 852–863.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. 2006. A framework for on-demand classification of evolving data streams. *IEEE Transactions on Knowledge and Data Engineering 18,* 5, 577–589.

Alahakoon, D., Halgamuge, S. K., and Srinivasan, B. 2000. Dynamic self-organizing maps with controlled growth for knowledge discovery. *Neural Networks, IEEE Transactions on 11,* 3, 601–614.

Alexander, H., Er, H., and Daniel, A. K. 1998. An efficient approach to clustering in large

multimedia databases with noise. In *Proceeding of the 1998 ntenational Conf. Knowledge Discovery and Data Mining (KDD98)*. New York, 58–65.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. 1999. Optics: ordering points to identify the clustering structure. In *SIGMOD '99 Proceedings of the 1999 ACM SIGMOD international conference on Management of data*. Vol. 28. ACM, 49–60.

Bifet, A. 2010. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*. Proceeding of the 2010 conference on Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams. IOS Press.

Bohm, C., Kailing, K., Kriegel, H.-P., and Kroger, P. 2004. Density connected clustering with local subspace preferences. In *Proceedings of the Fourth IEEE International Conference on Data Mining*. IEEE Computer Society, 27–34.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. A. 1984. *Classification and Regression Trees*. Chapman and Hall/CRC.

Cao, F., Ester, M., Qian, W., and Zhou, A. 2006. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining*. 328–339.

Chakrabarti, D., Kumar, R., and Tomkins, A. 2006. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Philadelphia, PA, USA, 554–560.

Chen, Y. and Tu, L. 2007. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, San Jose, California, USA, 133–142.

Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 153–162.

Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. 2009. On evolutionary spectral clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD) 3,* 4, 1–30.

Chien, S. and Immorlica, N. 2005. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th international conference on World Wide Web*. ACM, Chiba, Japan, 2–11.

Chow, T. W. S. and Sitao, W. 2004. An online cellular probabilistic self-organizing map for static and dynamic data sets. *IEEE Transactions on Circuits and Systems I: Regular Papers 51,* 4, 732–747.

Chris, G., Jiaweihan, Y., Jianpei, Z., Xifeng Yan, Y., and Philip, S. Y. 2003. Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining 212*, 191–212.

Ciaccia, P., Patella, M., and Zezula, P. 1997. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., 426–435.

Cui, W., Liu, S., Tan, L., Shi, C., Song, Y., Gao, Z., Qu, H., and Tong, X. 2011. Textflow: Towards better understanding of evolving topics in text. *Visualization and Computer Graphics, IEEE Transactions on 17,* 12, 2412–2421.

Dang, X., Lee, V., Ng, W., Ciptadi, A., and Ong, K. 2009. An em-based algorithm for clustering data streams in sliding windows. In *Database Systems for Advanced Applications*. Lecture Notes in Computer Science, vol. 5463. Springer Berlin / Heidelberg, 230–235.

de Leeuw, J. 2005. Applications of convex analysis to multidimensional scaling. In *Recent Developments in Statistics*, J. Barra, F. Brodeau, G. Romier, and B. V. Cutsem, Eds. North Holland Publishing Company, Amsterdam, 133–146.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39,* 1, 1–38.

Domingos, P. and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Boston, Massachusetts, United States, 71–80.

Dork, M., Gruen, D., Williamson, C., and Carpendale, S. 2010. A visual backchannel for large-scale events. *Visualization and Computer Graphics, IEEE Transactions on 16,* 6, 1129–1138.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. 2005. Mining data streams: a review. *ACM Sigmod Record 34,* 2, 18–26.

Gama, J. and Gaber, M. M. 2007. *Learning from Data Streams – Processing Techniques in Sensor Networks*. Springer.

Gama, J. and Rodrigues, P. 2009. An overview on mining data streams. In *Foundations of Computational, Intelligence Volume 6*, A. Abraham, A.-E. Hassanien, A. d. L. F. d. Carvalho, and V. Snel, Eds. Studies in Computational Intelligence, vol. 206. Springer Berlin / Heidelberg, 29–45.

Guha, S., Kim, C., and Shim, K. 2004. Xwave: optimal and approximate extended wavelets. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*. VLDB Endowment, Toronto, Canada, 288–299.

Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L. 2000. Clustering data streams. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*. 359–366.

Guha, S., Rastogi, R., and Shim, K. 1998. Cure: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. ACM, Seattle, Washington, United States, 73–84.

Guha, S., Rastogi, R., and Shim, K. 1999. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*. IEEE Computer Society, 512.

Guttman, A. 1984. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. 47–57.

Hahsler, M. and Dunham, M. 2011. Temporal structure learning for clustering massive data streams in real-time. In *SIAM Conference on Data Mining (SDM11)*. SIAM, Mesa, Arizona, 664–675.

Han, J. 2005. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Hulten, G., Spencer, L., and Domingos, P. 2001. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, San Francisco, California, 97–106.

Karypis, G., Eui-Hong, H., and Kumar, V. 1999. Chameleon: hierarchical clustering using dynamic modeling. *Computer 32,* 8, 68–75.

Kaufman, L. and Rousseeuw, P. 1990. *Finding groups in data: an introduction to cluster analysis*. Vol. 39. Wiley Online Library.

Kelly, M. G., Hand, D. J., and Adams, N. M. 1999. The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, San Diego, California, United States, 367–371.

Kranen, P., Gnnemann, S., Fries, S., and Seidl, T. 2010. Mc-tree: Improving bayesian anytime classification. In *Scientific and Statistical Database Management*, M. Gertz and B. Ludscher, Eds. Lecture Notes in Computer Science, vol. 6187. Springer Berlin / Heidelberg, 252–269.

Kriegel, H.-P., Krger, P., Ntoutsi, I., and Zimek, A. 2011. Density based subspace clustering over dynamic data. In *Scientific and Statistical Database Management*, J. Bayard Cushing, J. French, and S. Bowers, Eds. Lecture Notes in Computer Science, vol. 6809. Springer Berlin / Heidelberg, 387–404.

Kriegel, H.-P., Kroger, P., and Zimek, A. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data 3,* 1, 1–58.

Last, M. 2002. Online classification of nonstationary data streams. *Intelligent Data Analysis 6,* 2, 129–147.

Lei, Y. and Huan, L. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *The 20th ICML*. 856–863.

Leite, D., Costa, P., and Gomide, F. 2010. Evolving granular neural network for semi-supervised data stream classification. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

Leite, D. F., Costa, P., and Gomide, F. 2009. Evolving granular classification neural networks. In *International Joint Conference on Neural Networks, 2009. IJCNN 2009.* 1736–1743.

Lhr, S. and Lazarescu, M. 2009. Incremental clustering of dynamic data streams using connectivity based representative points. *Data and Knowledge Engineering 68,* 1, 1–27.

Liangxiao, J., Zhihua, C., Dianhong, W., and Siwei, J. 2007. Survey of improving k-nearest-neighbor for classification. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007.* Vol. 1. 679–683.

Lin, Y., Sun, J., Cao, N., and Liu, S. 2010. Contextour: Contextual contour visual analysis on dynamic multi-relational clustering. In *In Proceedings of 2010 SIAM International Conference on Data Mining (SDM 2010)*. Columbus, Ohio, USA, 418–429.

Liu, H. and Yu, L. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering 17,* 4, 491–502.

Martin, E., Hans-Peter, K., Jrg, S., and Xiaowei, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*. AAAI Press, 226–231.

Minku, L. L., White, A. P., and Xin, Y. 2010. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering 22,* 5, 730–742.

Muthukrishnan, S. M. 2003. *Data Streams: Algorithms and Applications*. Now Publishers Inc.

Ng, R. T. and Han, J. 2002. Clarans: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering 14,* 5, 1003–1016.

Nguyen, H.-L., Ng, W.-K., and Woon, Y.-K. 2012. Heterogeneous ensemble for feature drifts in data streams. In *The 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, Kuala Lumpur. Malaysia.

O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., and Motwani, R. 2002. Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*. 685–694.

Oza, N. C. 2005. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*. Vol. 3. IEEE, Waikoloa, HI, USA, 2340–2345.

Park, N. H. and Lee, W. S. 2004. Statistical grid-based clustering over data streams. *ACM SIGMOD Record 33,* 1, 32–37.

Park, N. H. and Lee, W. S. 2007. Cell trees: An adaptive synopsis structure for clustering multi-dimensional on-line data streams. *Data and Knowledge Engineering 63,* 2, 528–549.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning 1,* 1, 81–106.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann.

Rai, P., Daum, H., and Venkatasubramanian, S. 2009. Streamed learning: one-pass svms. In *Proceedings of the 21st international jont conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., Pasadena, California, USA, 1211–1216.

Sattar, H., Ying, Y., Zahra, M., and Mohammadreza, K. 2009. Adapted one-vs-all decision trees for data stream classification. In *IEEE Transactions on Knowledge and Data Engineering*. Vol. 21. 624 – 637.

Seidl, T., Assent, I., Kranen, P., Krieger, R., and Herrmann, J. 2009. Indexing density models for incremental learning and anytime classification on data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. ACM, Saint Petersburg, Russia, 311–322.

Sheikholeslami, G., Chatterjee, S., and Zhang, A. 2000. Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal 8,* 3-4, 289–304.

Smith, T. and Alahakoon, D. 2009. Growing self-organizing map for online continuous clustering. In *Foundations of Computational Intelligence Volume 4*, A. Abraham, A.-E. Hassanien,

and A. de Carvalho, Eds. Studies in Computational Intelligence, vol. 204. Springer Berlin / Heidelberg, 49–83.

Sow, D., Biem, A., Blount, M., Ebling, M., and Verscheure, O. 2010. Body sensor data processing using stream computing. In *Proceedings of the international conference on Multimedia information retrieval*. ACM, Philadelphia, Pennsylvania, USA, 449–458.

Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., and Schult, R. 2006. Monic: modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 706–711.

Street, W. N. and Kim, Y. 2001. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, San Francisco, California, 377–382.

Tasoulis, D. K., Ross, G., and Adams, N. M. 2007. Visualising the cluster structure of data streams. In *Proceedings of the 7th international conference on Intelligent data analysis*. Springer-Verlag, Ljubljana, Slovenia, 81–92.

Teh, Y., Jordan, M., Beal, M., and Blei, D. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association 101,* 476, 1566–1581.

Tsang, I. W., Kocsor, A., and Kwok, J. T. 2007. Simpler core vector machines with enclosing balls. In *Proceedings of the 24th international conference on Machine learning*. ACM, Corvalis, Oregon, 911–918.

Udommanetanakit, K., Rakthanmanon, T., and Waiyamai, K. 2007. E-stream: Evolution-based technique for stream clustering. In *Advanced Data Mining and Applications*, R. Alhajj, H. Gao, X. Li, J. Li, and O. Zaane, Eds. Lecture Notes in Computer Science, vol. 4632. Springer Berlin / Heidelberg, 605–615.

Vapnik, V. 1999. *The Nature of Statistical Learning Theory*. Springer.

Wan, L., Ng, W. K., Dang, X. H., Yu, P. S., and Zhang, K. 2009. Density-based clustering of data streams at multiple resolutions. *ACM Transactions on Knowledge Discovery from Data (TKDD) 3,* 3, 1–28.

Wang, H., Fan, W., Yu, P. S., and Han, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Washington, D.C., 226–235.

Wang, W., Yang, J., and Muntz, R. 1997. Sting: A statistical information grid approach to spatial data mining. *Proceedings of the International Conference on Very Large Data Bases*, 186–195.

Zeng, H.-J., He, Q.-C., Chen, Z., Ma, W.-Y., and Ma, J. 2004. Learning to cluster web search results. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, Sheffield, United Kingdom, 210–217.

Zhang, P., Gao, B., Zhu, X., and Guo, L. 2011. Enabling fast lazy learning for data streams. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM-11)*. IEEE, Vancouver, Canada, 932–941.

Zhang, P., Li, J., Wang, P., Gao, B. J., Zhu, X., and Guo, L. 2011. Enabling fast prediction for ensemble models on data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, San Diego, California, USA, 177–185.

Zhang, P., Zhu, X., and Shi, Y. 2008. Categorizing and mining concept drifting data streams. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Las Vegas, Nevada, USA, 812–820.

Zhang, P., Zhu, X., Shi, Y., Guo, L., and Wu, X. 2011. Robust ensemble learning for mining noisy data streams. *Decision Support Systems 50,* 2, 469–479.

Zhang, P., Zhu, X., Shi, Y., and Wu, X. 2009. An aggregate ensemble for mining concept drifting data streams with noise. In *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Lecture Notes in Computer Science, vol. 5476. Springer Berlin / Heidelberg, 1021–1029.

Zhang, T., Ramakrishnan, R., and Livny, M. 1996. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*. ACM, Montreal, Quebec, Canada, 103–114.

Zhou, A., Cao, F., Qian, W., and Jin, C. 2008. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems 15,* 2, 181–214.

Zliobaite, I. 2010. Learning under concept drift: an overview. Tech. rep., Vilnius University.