Advanced Review

Data stream mining in ubiquitous environments: state-of-the-art and current directions

Mohamed Medhat Gaber,[1] João Gama,[2*] Shonali Krishnaswamy,[3] João Bártolo Gomes[4] and Frederic Stahl[5]

In this article, we review the state-of-the-art techniques in mining data streams for mobile and ubiquitous environments. We start the review with a concise background of data stream processing, presenting the building blocks for mining data streams. In a wide range of applications, data streams are required to be processed on small ubiquitous devices like smartphones and sensor devices. Mobile and ubiquitous data mining target these applications with tailored techniques and approaches addressing scarcity of resources and mobility issues. Two categories can be identified for mobile and ubiquitous mining of streaming data: single-node and distributed. This survey will cover both categories. Mining mobile and ubiquitous data require algorithms with the ability to monitor and adapt the working conditions to the available computational resources. We identify the key characteristics of these algorithms and present illustrative applications. Distributed data stream mining in the mobile environment is then discussed, presenting the *Pocket Data Mining* framework. Mobility of users stimulates the adoption of context-awareness in this area of research. Context-awareness and collaboration are discussed in the *Collaborative Data Stream Mining*, where agents share knowledge to learn adaptive accurate models.

How to cite this article:
WIREs Data Mining Knowl Discov 2014, 4:116–138. doi: 10.1002/widm.1115

INTRODUCTION

The phenomenal growth of mobile devices coupled with their ever-increasing computational capacity presents an exciting new opportunity for real time, intelligent data analysis in mobile and ubiquitous environments. Ubiquitous or mobile data mining refers to the process of performing data stream mining using mobile and/or embedded devices (e.g., sensors) to support critical applications such as mobile healthcare, intelligent transportation systems, mobile activity recognition, smart homes, and emergency or disaster management like bush fires.[1,2] Thus, the key focus is on developing data stream mining algorithms that are highly scalable or computationally efficient, energy-efficient, and context or resource-aware. These features enable the continued operation of data stream mining algorithms in a highly dynamic mobile or ubiquitous environment. The typical constraints that have to be addressed in performing mobile or ubiquitous data stream mining are: (1) data streams are generated and sent in real time in a stream format with little or no potential for persistent storage, (2) resource constraints include limited computational resources such as memory, processor speed, network bandwidth, battery power, and screen real-estate, (3) temporal constraints refer to real-time information and decision-making needs that, in turn, necessitate the analysis to be online, incremental, and continuous, (4) mobility of users and devices and the connectivity issues thereof, and

*Correspondence to: jgama@fep.up.pt
[1]Robert Gordon University, Aberdeen, AB10 1FR, UK
[2]University of Porto, Porto, 4050-190, Portugal
[3]Monash University, Melbourne, Victoria 3800, Australia
[4]Institute for Infocomm Research, Singapore 138632, Singapore
[5]University of Reading, Reading RG6 6AY, UK
Conflict of interest: The authors have declared no conflicts of interest for this article.

(5) adaptation and context-awareness of the analysis process to varying or dynamically changing resource-levels and user needs.

In the past few years, rapid strides have been made in accurately and efficiently mining high-speed data streams in mobile devices such as smartphones and there is a growing focus on 'in-network' processing using embedded devices such as sensor nodes. These techniques leverage the body of work that exists in mining data streams and aim to enable the operation of these algorithms in resource-constrained environments. There is also an emerging focus on context-aware data stream mining which targets the learning process to be aware of its operational and application constraints and self-adapt according to changing situations or needs. The body of work in mobile or ubiquitous data stream mining ranges from algorithms, adaptation strategies for coping with mobile or ubiquitous environments, systems or toolkits for mobile data mining, and innovative or new applications.

This article is organized as follows. We start the review with a concise background of fundamental streaming techniques used in data mining. Section *Ubiquitous Data Mining: From Algorithms to Applications* discusses the key characteristics of mining mobile and ubiquitous data spanning state-of-the-art techniques and application areas. Section *PDM: A General Framework for Ad hoc Mobile and Distributed Data Mining* describes the *Pocket Data Mining* (PDM), a generic framework for distributed *ad hoc* data stream mining. Section *Collaborative Data Stream Mining in Ubiquitous Environments* presents a framework for collaborative mining between peer agents that share knowledge to learn adaptive accurate models. Finally, we conclude the article with a summary and pointers to future directions. To increase the readability of the article, Figure 1 gives a roadmap of how the article flows through its sections.

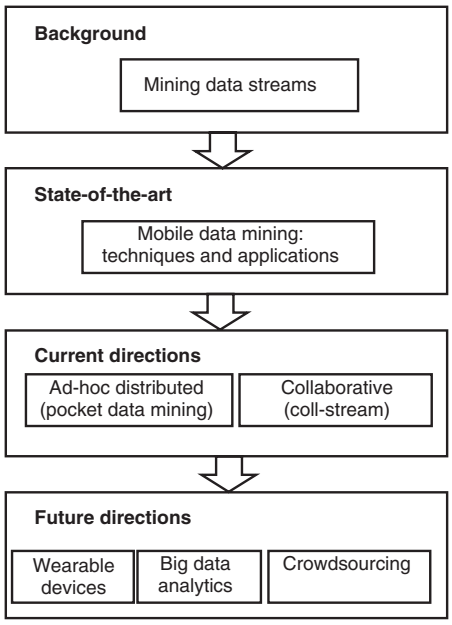

FIGURE 1 | A roadmap of the survey.

STREAMING ALGORITHMS

The developments of information and communication technologies dramatically change the data collection and processing methods. What distinguish current datasets from earlier ones are *automatic data feeds*. We do not just have people entering information into a computer. We have computers entering data into each other.[3,4] Moreover, advances in miniaturization and sensor technology lead to sensor networks, collecting high detailed spatiotemporal data about the environment.

Data mining in this context requires continuous processing of the incoming data monitoring trends, and detecting changes. Traditional one-shot systems—memory based, trained from fixed training sets and generating static models are not prepared to process the high detailed data available—are also not able to continuously maintain a predictive model consistent with the actual state of the nature, or to quickly react to changes.

In the streaming model the input elements $a_1, a_2, \ldots, a_j, \ldots$ arrive sequentially, item by item.[3] We can distinguish between two different models:

1. Insert Only Model: once an element a_i is seen, it cannot be changed;
2. Insert–Delete Model: element a_i can be deleted or updated.

From the view point of a Data Streams Management Systems (DSMS),[5] several research issues emerge that require approximate query processing techniques to evaluate continuous queries that require unbounded amount of memory.[6] A relevant issue is the definition of the semantics (and implementation) of blocking operators (operators that only return an output tuple after processing all input tuples, like join, aggregation, and sorting) in the presence of unending streams.

Algorithms that process data streams deliver approximate solutions, providing a fast answer using few memory resources; they relax the requirement of an exact answer to an approximate answer within a small error range with high probability. In general, as the range of the error decreases the space of computational resources goes up.

TABLE 1 | Summary of the Main Differences Between Standard Database Processing and Data Stream Processing

	Databases	Data Streams
Data access	Random	Sequential
Number of passes	Multiple	Single
Processing time	Unlimited	Restricted
Available memory	Unlimited	Fixed
Result	Accurate	Approximate
Distributed	No	Yes

In some applications, mostly database oriented, an approximate answer should be within an admissible error margin. DSMS developed a set of techniques that store compact stream summaries that are enough to approximately solve queries. All these approaches require a trade-off between accuracy and the amount of memory used to store the summaries, with an additional constrain of small time to process data items.[3,7] The most common problems end up to compute quantiles, frequent item sets, and to store frequent counts along with error bounds on their true frequency. The techniques developed are used in very high dimensions both in the number of examples and in the cardinality of the variables (Table 1).

Approximation and Randomization

Many fundamental questions, like counting, require linear space in the input to obtain exact answers. Within data stream framework, *approximation* techniques—i.e., answers that are correct within some small fraction ϵ of error—and *randomization*[8]—that allows a small probability δ of failure—are used to obtain answers with probability $1-\delta$ are in an interval of radius ϵ. Algorithms that use both approximation and randomization are referred to as (ϵ, δ) approximations. The base idea consists of mapping a very large input space to a small synopsis of size $O\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right)$.

Approximation and randomization techniques have been used to solve problems like measuring the entropy of a stream,[9] association rule mining frequent items,[10] k-means clustering for distributed data streams using only local information,[11] and so on. Most of the approximate tools for stream processing are based on hash functions. They are used to project huge domains into lower space dimensions.[3,12]

Time Windows

Most of the time, we are not interested in computing statistics over all the past, but only over the *recent* past. The assumption behind all these models is that most recent information is more relevant than historical data. The simplest situation uses *sliding windows* of fixed size. These types of windows are similar to *first in, first out* data structures. Whenever an element j is observed and inserted into the window, another element $j-w$, where w represents the window size, is forgotten.

Several window models have been presented in the literature. Babcock et al.[13] defines two basic types of sliding windows: (1) Sequence based—the size of the window is defined in terms of the number of observations. Two different models are *sliding windows* of fixed size w and *landmark windows*, where the window grows from a specific point in time (the landmark); (2) Timestamp based—the size of the window is defined in terms of *duration*. A timestamp window of size t consists of all elements whose timestamp is within a time interval t of the current time period.

Monitoring, analyzing, and extracting knowledge from high-speed streams might explore multiple levels of granularity, where the recent history is analyzed at fine levels of granularity and the need of precision decreases with the age of the data. As a consequence, the most recent data can be stored at the finest granularity, while more distant data at coarser granularity. This is called the *tilted* time window model. It might be implemented using exponential histograms.[14]

Sequence-based window is a general technique to deal with changes in the process that generates data. Concept drift and change detection are key issues in stream processing. They are fundamental blocks to maintain a decision model consistent with the most recent data. A reference algorithm is the AdWin (ADaptive sliding WINdow) presented by Bifet and Gavaldà.[15] AdWin keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis *there has been no change in the average value inside the window*. More precisely, an older fragment of the window is dropped if and only if there is enough evidence that its average value differs from that of the rest of the window. This has two consequences: first, that change is reliably declared whenever the window shrinks; and second, that at any time the average over the existing window can be reliably taken as an estimate of the current average in the stream (barring a very small or very recent change that is still not statistically visible). AdWin is parameter- and assumption-free in the sense that it automatically detects and adapts to the current rate of change. Its

only parameter is a confidence bound δ indicating how confident we want to be in the algorithm's output, a property inherent to all algorithms dealing with random processes. AdWin does not maintain the window explicitly, but compresses it using a variant of the exponential histogram technique. This means that it keeps a window of length W using only $O(\log W)$ memory and $O(\log W)$ processing time per item.

Sampling

Sampling is a common practice for selecting a subset of data to be analyzed. Instead of dealing with an entire data stream, we select instances at periodic intervals. Sampling is used to compute statistics (expected values) of the stream. While sampling methods reduce the amount of data to process, and, by consequence, the computational costs, they can also be a source of errors, namely in monitoring applications that require to detect anomalies or extreme values.

The main problem is to obtain a *representative* sample, i.e., a subset of data that has approximately the same properties of the original data. In statistics, most techniques require to know the length of the stream. For data streams, we need to modify these techniques. The simplest form of sampling is *random sampling*, where each element has equal probability of being selected.[16] The *reservoir sampling* technique[17] is the classic algorithm to maintain an online random sample. The base idea consists of maintaining a sample of size k, called the reservoir. As the stream flows, every new element has a probability k/n, where n is the number of elements seen so far, of replacing an old element in the reservoir. A similar technique, load shedding, drops sequences in the stream, when bursts cause bottlenecks in the processing capabilities. Tatbul et al.[18] discuss load shedding techniques in querying high-speed data streams.

Synopsis, Sketches, and Summaries

Synopsis are compact data structures that summarize data for further querying. Several methods have been used, including wavelets,[19] exponential histograms,[14] frequency moments,[20] and so on. Data sketching via random projections is a tool for dimensionality reduction. Sketching uses random projections of data points with dimension d to a space of a subset of dimensions. It has been used for moment estimation,[20] computing L-norms,[3] and dot product of streams.[16]

Cormode and Muthukrishnan[21] present a data stream summary, the so called *count-min* sketch, used for (ϵ, δ) approximations to solve *point queries*, *range queries*, and *inner product queries*. Consider an implicit vector $\mathbf{a}$ of dimension n that is incrementally updated over time. At each moment, the element a_i represents the counter associated with element i. A point-query is to estimate the value of an entry in the vector $\mathbf{a}$. The *count-min* sketch data structure, with parameters (ϵ, δ), is an array of $w \times d$ in size, where $d = \log(1/\delta)$, and $w = 2/\epsilon$. For each incoming value of the stream, the algorithm uses d hash functions to map entries to $[1, \ldots, w]$. The counters in each row are incremented to reflect the update. From this data structure, we can estimate at any time, the number of occurrences of any item j by taking $\min_d \mathrm{CM}[d, h_d(j)]$. Since the space used by the sketch CM is typically much smaller than that required to represent the vector $\mathbf{a}$ exactly, there is necessarily some approximation in the estimate. The estimate $\widehat{a}_j$, has the following guarantees: $a_j \leq \widehat{a}_j$, and, with probability at least $1 - \delta$, $\widehat{a}_i \leq a_i + \epsilon ||a||_1$. The error of the estimate is at most ϵ with probability at least $1 - \delta$ in space $O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$.

Algorithms for Learning from Data Streams

Data Mining studies the automated acquisition of domain knowledge looking for the improvement of systems performance as result of experience. Data stream mining systems address the problems of data processing, modeling, prediction, clustering, and control in the changing and evolving environment.[4]

One of the most successful stream mining algorithms to learn from arbitrarily large databases was presented by Domingos and Hulten.[22] The method consists of deriving an upper bound for the learner's loss as a function of the number of examples used in each step of the algorithm. Then it is used to minimize the number of examples required at each step, while guaranteeing that the model produced does not differ significantly from the one that would be obtained with infinite data. This methodology has been successfully applied in hierarchical clustering of variables,[23] decision trees,[22,24] regression trees,[25] decision rules,[26] and so on.

The most representative algorithm in this line is the Very Fast Decision Tree (VFDT).[22] VFDT (aka Hoeffding trees) is a decision-tree learning algorithm that dynamically adjusts its bias accordingly to the availability of data. In decision-tree induction, the main issue is the decision of when to expand the tree, installing a splitting-test and generating new leaves. The basic idea consists of using a small set of examples to select the splitting-test to incorporate in a decision-tree node. If after seeing a set of examples, the difference of the merit between the two best splitting-tests does not satisfy a statistical test (the Hoeffding bound), VFDT proceeds by examining more examples. It only makes a decision (i.e., adds a splitting-test in that node), when there is enough statistical

evidence in favor of a particular test. This strategy guarantees model stability (low variance) and controls overfitting—examples are processed once without the need of model regularization (pruning). This profile is quite different from the standard decision-tree models using greedy search and static datasets. Using static datasets, the decisions in deeper nodes of the tree are based on less and less examples. Statistical support decreases as the tree grows, and regularization is mandatory. In VFDT-like algorithms, the number of examples needed to grow a node is only defined by the statistical significance of the difference between the two best alternatives. Deeper nodes of the tree might require more examples than those used in the root of the tree!

VFDT has been extended to deal with time evolving streams. System concept-adapting very fast decision tree (CVFDT)[24] (aka adaptive Hoeffding trees), continuously monitors the quality of previous decisions (splitting features) with respect to a sliding window of fixed size over the most recent data stream. CVFDT maintains a window of training examples and keeps its learned tree up-to-date with this window by monitoring the quality of its old decisions as data move into and out of the window.

Distributed Streams

Data streams are distributed in nature. Learning from distributed data, we need efficient methods in minimizing the communication overheads between nodes.[27,28]

The strong limitations of centralized solutions is discussed in depth in Refs 29–31. The authors point out *a mismatch between the architecture of most off-the-shelf data mining algorithms and the needs of mining systems for distributed applications*. Such mismatch may cause a bottleneck in many emerging applications, namely hardware limitations related to the limited bandwidth channels. Most importantly, in applications like monitoring, centralized solutions introduce delays in event detection and reaction that can make mining systems useless.

Another direction, for distributed processing, explore multiple models.[32,33] Kargupta et al.[33] propose a method that offer an effective way to construct a redundancy-free, accurate, and meaningful representation of large decision-tree ensembles often created by popular techniques such as Bagging, Boosting, Random Forests, and many distributed and data stream mining algorithms.

Having discussed the main topics in data stream mining in this section, the following section will provide a comprehensive review of key techniques and applications when performing data stream mining in the ubiquitous and mobile environment.

UBIQUITOUS DATA MINING: FROM ALGORITHMS TO APPLICATIONS

We have presented an overview and general principles of data stream mining techniques in the previous section. This section will focus on the mobile and ubiquitous data stream mining. In particular we will present:

- An overview of the strategies or techniques to enable data stream mining algorithms to be 'adapted' to facilitate their efficient functioning in mobile or ubiquitous environments;
- Algorithms for mobile or ubiquitous data stream mining;
- The Open Mobile Miner Toolkit for developing and deploying mobile or ubiquitous data mining applications;
- Case studies and exemplars of mobile data mining applications.

Key Principles for Data Stream Mining in Ubiquitous Environments

From the earliest study and investigation of data stream mining algorithms for resource-constrained environments such as mobile and ubiquitous devices, it became apparent that there is a need for algorithms that are *computationally efficient* and *adaptive* to their dynamic and varying operational context. The fundamental premise of mobile and ubiquitous environments is not merely that the available computational resources are limited, but that these resources (and other factors such as the data rates) are subject to continuous variability. Thus, to address these twin requirements, data stream mining algorithms that are developed for mobile and ubiquitous environments need to be *cost-efficient and light-weight in terms of their resource-utilization profile as well as adaptive to the changing context*. There are three fundamental models or strategies for enabling adaptive behavior in mobile or ubiquitous data stream mining algorithms: *resource-awareness*, *situation-awareness*, *and a hybrid approach which combines the former two approaches*. Analyzing large amounts of sensory-originated data in real time is a very challenging task. This challenge is further exacerbated when data are processed with resource-constrained devices such as mobile phones. Resource constraints include limited computational resources such as memory, processor speed, network

bandwidth, battery power, and screen real-estate. Current smart phones (e.g., iPhone and Samsung Galaxy) have RAM memory up to 1 GB, processing speed up to 1.2 GHz Dual Core while modern desktops and laptops have up to 16 GB RAM memory and include powerful processors such as Intel Core i5 or i7. The most constraining resource on smart phones compared to desktop computers is their limited battery life which is up to 8-h talk time, up to 400-h stand-by time, and about 4 h for tethering and mobile Access Point (AP).

Thus adaptation in the context of mobile or ubiquitous data stream mining simply implies that the continuous processing of incoming data streams onboard a resource-constrained device must factor in the changing availability of resources while doing the computational processing or mining tasks. At a very abstract level, this adaptation takes the form of continuously (on the basis of some defined temporal window) monitoring for the status of the environmental and application constraints such as data rates, resource levels, and so on. Thus, this time window is the temporal unit for performing processing of the incoming streams. On the basis of the status of the current context, the temporal window itself can be made smaller or larger. Thus, a larger temporal window implies that the processing of the streams happens less frequently, and smaller temporal window conversely implies that the processing of the data stream happens much more frequently. A higher frequency of processing (i.e., a smaller temporal window) implies that the consumption of resources will be high and also the accuracy of the processing during that window will be higher though closer to real-time results. The converse applies to longer temporal windows.

Previous studies on resource-aware adaptation[34–37] show that dynamic adaptation to data rates and fine tuning of processing parameters can significantly enhance the longevity of continuous real-time processing of data streams in mobile environments. The Granularity-based Adaptation (GA)[34] is a generic efficient adaptation approach that can be used with any data stream mining technique running on a resource-constrained device. This approach facilitates adaptation of data stream mining algorithms to varying data rates and available computational resources in mobile devices by innovative strategies to perform knowledge integration, controlling the rate of learning and varying the accuracy levels of the discovered patterns. In addition to availability of resources, mobile data mining application's accuracy requirements vary according to the occurring situations. A situation-aware adaptation technique controls the data stream mining settings according to the occurring situations and accuracy requirements of the running application.[38] There can be other scenarios in which it is important to adjust mining algorithms considering both, the current situation and resource availability in order to maintain the application's accuracy needs as well as the continuity of mining operations. In such cases, there is a need to apply a hybrid adaptation strategy that combines situation and resource-aware adaptation methods.[39,38]

Resource-Aware Adaptation

The dominating factor of mining stream data on mobile devices is the high input rate with regard to the available computational resources. Data streams are generated and sent in real time in a stream format. The input rates of data streams can range from hundreds of records per second to megabytes or terabytes of tuples per second.[37] Given the fact that the state-of-the-art techniques in the area have only focused on data reduction or approximating the results in a low complexity of space and time, we have proposed to adapt the mining algorithm according to resource availability and data stream rate. This approach is termed GA.[37] The GA approach has three different variations:

- Algorithm Input Granularity (AIG) is a process that adapts the data rates feeding into the algorithm according to the battery charge (see Figure 2).
- Algorithm Output Granularity (AOG) provides adaptability by adjusting the algorithm output rate (e.g., the number of clusters; see Figure 2). We will discuss this process in more detail in Section *Key Principles for Designing*

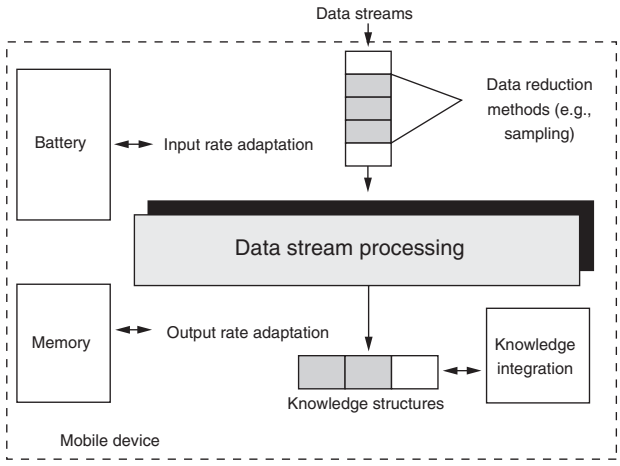

FIGURE 2 | Input and output rate adaptation based on resource levels using Algorithm Input Granularity (AIG) and Algorithm Output Granularity (AOG).

Ubiquitous Data Stream Mining Algorithms where the algorithm design considerations for mobile or ubiquitous data stream mining are discussed.

- Algorithm Processing Granularity (APG) performs adaptation of the processing settings of the algorithm with respect to the CPU usage.

Resource-aware adaptation focuses on resources (i.e., memory, battery, and CPU). Yet, the mobile data mining algorithm's cost-efficiency with regard to resource utilization can be improved further by factoring in the entire situational context of the application. This is due to the fact that the data mining application's requirements in terms of the accuracy (and therefore resource consumption) vary according to the current situations.

Situation-Aware Adaptation

Resource-aware adaptation aims to adjust the algorithm input and output rates (i.e., the algorithm accuracy) according to the resource levels of mobile devices to preserve resources. When the resource levels are low, a resource-aware adaptation moderately reduces the algorithm accuracy by decreasing the input or output rates. A high level of accuracy (without using adaptation) consumes resources quickly and can result in the mobile application failure.

The accuracy requirements of a mobile data mining application can change based on the situations that occur. By situations we mean application constraints. There are certain situations in which applications do not need high accuracy such as the 'healthy' situation in a health monitoring application. However, there are other situations like 'hypertension' (caused by high blood pressure) which will require a higher level of accuracy. A situation-aware approach can increase the accuracy in critical situations where there is a need for closer monitoring and more detailed output. However, when the current situation requires less frequent data analysis and less detailed mining results (i.e., low accuracy), this adaptation technique can decrease the algorithm accuracy to preserve resources.

To provide situation-awareness, there is a need for a context modeling and reasoning technique that can represent the current situation and more importantly is able to infer the situations from low-level context. Individual contextual parameters provide a limited view of the real-world and a partial understanding of the environment.[40] Multiple contextual parameters can be aggregated by employing reasoning techniques and used for inferring situations.[40] Fuzzy situation inference (FSI) technique[39] is a novel context modeling and reasoning approach that was developed to identify and represent real-world situations as well as the uncertainty associated with these situations. The inferred situations are used to enable a smooth and fine-grained adaptation of data mining algorithms' settings according to application constraints. FSI integrates fuzzy logic into the context spaces (CS) model.[40] It uses the benefits of the CS model for supporting pervasive computing environments while incorporating fuzzy logic to deal with uncertainty associated with real-world situations. In FSI, situations of interest are defined using fuzzy rules by domain experts. To model the importance of conditions, a weight is assigned to each condition with a value ranging between 0 and 1. The sum of weights is 1 per rule. A weight represents the importance of its assigned condition relative to other conditions in defining a situation. To reason about a situation, rules need to be evaluated to produce a single output that determines the membership degree of the consequent. Using fuzzy logic, the FSI model is able to compute the individual contribution levels of context values using the trapezoidal membership function. The membership degree of an element represents its contribution level according to the definition of the CS model. The inferred situations by FSI and their membership degrees are used by situation-aware and hybrid strategies for adaptation of data mining algorithm settings.

In the Situation-Aware strategy, a situation-aware adaptation technique controls the data stream mining settings (i.e., input and output rates) according to the occurring situations and accuracy requirements of the running application. During adaptation, the pre-initialized parameters of mining algorithms, such as sampling rate, are adjusted according to the degree of membership (i.e., a value between 0 and 1) of occurring situations. The pre-initialized parameters are defined for each situation and reflect the accuracy needs of application during occurrence of that situation.

In the Hybrid strategy (i.e., the combination of resource- and situation-aware strategies for adaptation) each algorithm computes a parameter value by considering the criticality values of situations and resources (i.e., battery and memory). Since the adaptation approach includes situation- and resource-aware and hybrid strategies, it is important that the appropriate strategy is selected at run-time. The selection is performed based on the concepts of situation and resource criticality. Situation criticality models the application's accuracy requirement (and resource consumption) for a situation by the concept of situation criticality. The criticality of a situation can be expressed by a value between 0 and 1. If a situation requires closer monitoring and more detailed data

analysis output, it should be given a higher criticality value (closer to 1), and if the situation needs a lower level of accuracy, it should be assigned a lower criticality value (closer to 0). Resource Criticality models the availability of resources and is expressed as a value between 0 and 1. When a resource such as memory is fully available (i.e., 100%), its criticality value is 0 which implies it is not critical.

To define the low and high criticality levels, there is a need for using a point of reference that values can be compared to. This is achieved by assigning thresholds (i.e., a value between 0 and 1) to resource and situation criticality. These thresholds are application-specific and determined by system designers and application domain experts. For example, the situations above the upper bound threshold with a value of 0.7 can be considered as critical situations requiring high accuracy. The situations below the lower bound threshold which is assigned a value of 0.3 can be regarded as noncritical. Noncritical situations do not need high accuracy. Using criticality values and thresholds enables the adaptation process to compare resources according to their levels and situations with regard to the application's accuracy requirement, and determine which strategy can achieve the required accuracy while using resources efficiently. We refer the readers to Ref 38 for a formal and detailed specification of situation-aware adaptation for mobile or ubiquitous data stream mining.

While the above adaptation process is general, the actual control operations effected to reduce or increase the accuracy of the mining process is of course dependent on the algorithm itself.

Key Principles for Designing Ubiquitous Data Stream Mining Algorithms

While the above adaptation process is generally critical for ensuring the continuity and longevity of data stream mining process, there are also key considerations that need to be factored into the design and development of mobile or ubiquitous data stream mining algorithms. These are important to enable the algorithms to work within the adaptive framework and remain aligned with the principles of being computationally light-weight and cost-efficient.

As discussed earlier, during a given temporal window, the mining algorithm needs to perform 'algorithm granularity' processes that adapt or adjust its operations to cope with either reduced or increased availability of resources and/or faster or slower data rates. This adjustment or adaptation actions typically has the resultant effect of increasing or reducing the accuracy of the stream mining algorithms. Thus, there is a strong positive co-relationship between the availability of resources and the accuracy of the stream mining process. This relationship is also impacted by the rate of the data stream during that temporal window. Since the stream mining process is continuous, at the commencement of a new 'temporal window' there is a need to re-adjust the processing of the mining algorithm to be cognizant of new changes in the operational context.

In the following discussion on mobile data stream mining algorithms, we will briefly describe some of the algorithms that have been 'customized' to operate on mobile or ubiquitous data stream mining based on these principles. Several algorithms have been developed specifically for mobile or ubiquitous data stream mining. These include:

- Clustering—Light-Weight Cluster (LWC), RA-Cluster (Resource-Aware Cluster), RA-VFKM (Resource-Aware Very Fast *K*-Means)
- Classification—Light-Weight Class (LWC), Naïve Bayes for Stream Mining
- Change Detection—CHANGE-DETECT
- Frequent Items and Associations—LWF (Light-Weight Frequent Items), HiCoRE (Highly Correlated Energy-Efficient Rules)
- Time-Series Analysis—RA-SAX and RA-HOT SAX
- Visualization—Clutter-Adaptive, Energy-efficient Visualizer for Mobile Data Analysis

In each of these algorithms, the adaptation process is an integral part of the data stream mining task. For instance in RA-VFKM, the Very Fast *K*-Means algorithm developed by Domingos and Hulten[41] is made resource-aware by leveraging the ϵ- and δ- parameters which are error parameters in the clustering algorithm. These parameters are controlled and changed during each processing cycle based on resource availability. By increasing the error parameter values, the algorithm was found to consume less computational resources and vice-versa. It was experimentally shown that using these control parameters, RA-VFKM was able to achieve significantly improved continuity and longevity in terms of resource-utilization and continued functioning onboard a mobile device, relative to the original or nonresource-aware version of the algorithm.

Similarly, in RA-SAX, the resource-aware version of the well-known time-series representation symbolic approximation (SAX),[42] the algorithm parameters of word-length and segment-size are

used as control parameters to enable resource-aware adaptation.

Thus in summary, it is really important to identify parameters or subprocesses in a data stream mining algorithm that have the capacity to improve or reduce the accuracy of the algorithm, and also have the potential to reduce or increase its computational footprint. These parameters are then used to align the algorithms to be scalable for onboard mobile or ubiquitous data stream mining. These algorithm parameters will be 'controlled' according to resource-availability, data stream rates, and application criticality levels—that is either through the previously discussed resource- or situation-aware or hybrid adaptation strategies.

Open Mobile Miner for Building Mobile Data Mining Applications

On the basis of the principles of adaptation to enable scalable and efficient data stream mining onboard mobile devices, an integrated and extensible toolkit to support the rapid development mobile data mining application's has been developed and readers are referred to Refs 43, 44 for a detailed overview of the toolkit. This toolkit named the Open Mobile Miner comes with a number of built-in implementations of adaptive and light-weight mobile data mining algorithms. The toolkit is operational on the Android OS platform and comes with the resource-aware adaptation strategy built-in. It also has a library of mobile data mining algorithms for clustering, classification, change detection, frequent pattern mining, and time-series analysis. The key stated objective of this technology is to facilitate the rapid development of mobile data mining applications, while at the same time being extensible to include new data sources, as well as integration of new algorithms and custom- or application-specific visualizations into the framework. The conceptual architecture of the OMM toolkit is shown in Figure 3.

An Overview of Mobile Data Mining Applications and Case Studies

This section presents a brief review of some key mobile or ubiquitous data mining applications and case studies that have been presented. These applications typically use the adaptation strategies and algorithms discussed in the preceding sections. Some of the applications also leverage the Open Mobile Miner toolkit implementations to support the application development and deployment. These applications or case studies help to illustrate the key areas and domains where mobile or ubiquitous data mining can bring advantages and benefits.

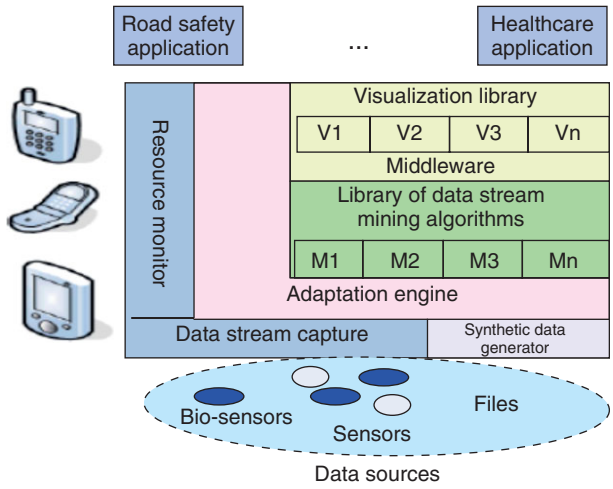

FIGURE 3 | Conceptual architecture of the open mobile miner toolkit.

- Mobile Crowd-Sensing is the process of collecting large-scale sensory data from mobile devices for purposes such as understanding traffic congestion, pollution levels, noise levels from large spatial regions, and so on. In Sherchan et al.[45] the use of mobile data mining to collect data from mobile devices for crowd-sensing applications was presented. The application used mobile data stream clustering coupled with change detection to demonstrate the significant reductions in data transfer, energy usage for data collection, while at the same time managing to capture high quality sensed information. The architecture developed is shown in Figure 4.
- Mobile Activity Recognition (MARS) is the process of recognizing and inferring a mobile user's activities through mobile sensor data such as the accelerometer. In Gomes et al.[46,47] a highly accurate application for recognizing user activities using an adaptive onboard version of Naive Bayes is presented. The advantage of onboard mobile activity learning and recognition is that it can be personalized as well as preserve the privacy of the individual. The MARS system is shown in Figure 5.
- Next Location Prediction (*NextLocation*)[48–50] is a framework for the challenging problem of predicting the next location of a mobile user given data on his or her current location. *NextLocation* is a personalized mobile data mining approach that not only uses spatial and temporal data, but also other contextual mobile data such as accelerometer, Bluetooth, and call or SMS log. In addition, *NextLocation* represents a new paradigm for privacy-preserving next place

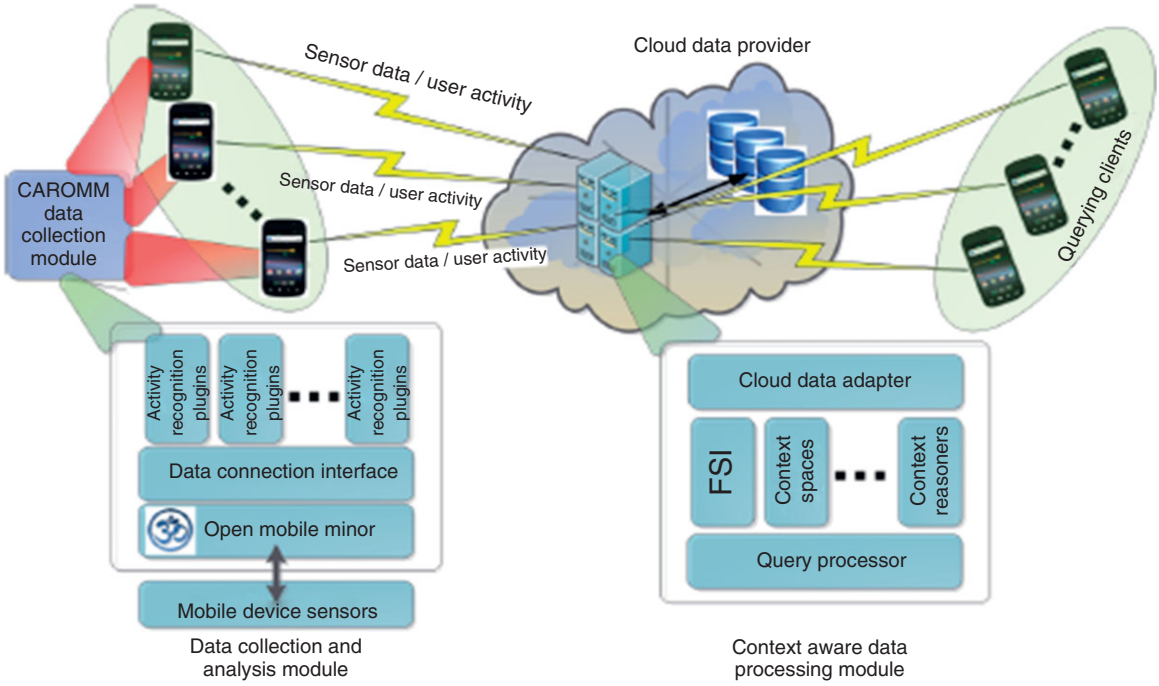

FIGURE 4 | Mobile data mining for mobile crowd-sensing.

FIGURE 5 | Mobile data mining for mobile activity recognition.

prediction as the mobile phone data is not shared without user permission.

- Context-Aware Energy Conservation in Sensor Networks is an application that uses mobile and ubiquitous data stream mining to learn highly co-related rules in sensor networks and then leverages these to control the sensor network to improve the lifetime of the wireless sensor network itself.[51] The CASE application uses the HiCoRE algorithm for learning the correlations between sensors and sensor data being collected in the network. The CASE framework is shown in Figure 6.
- Remote Health Monitoring is an application where mobile data stream mining can be used to support 24 × 7 monitoring for patients wearing biosensors. Typical remote health monitoring applications use the mobile phone

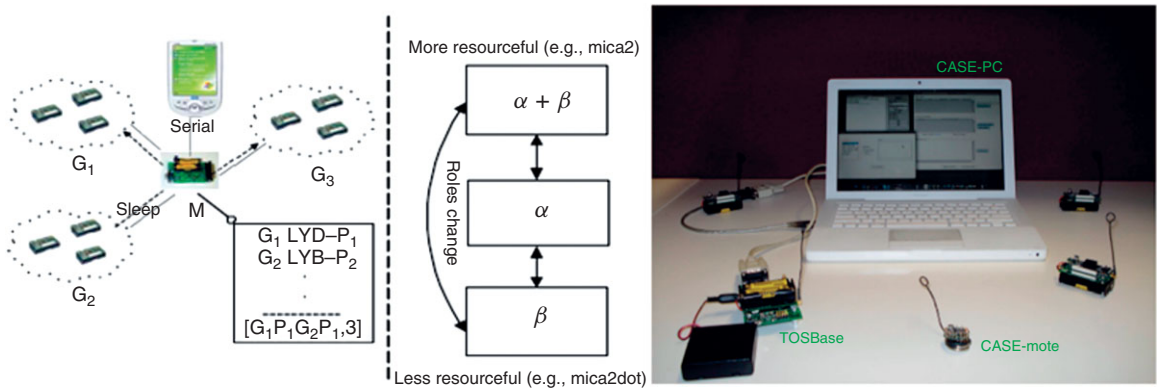

FIGURE 6 | Mobile data mining for energy conservation in wireless sensor networks.

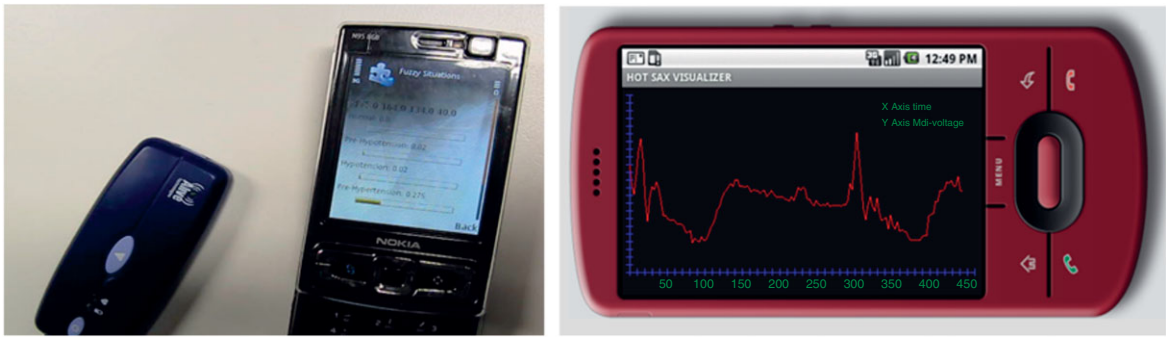

FIGURE 7 | Mobile data mining for remote health monitoring.

as a transmission or communications device to transfer the data collected from the patient wearing the biosensors. Using mobile data stream mining it is possible to have some of the analysis being done immediately onboard the mobile device, issue immediate alerts to care-givers when required, while at the same time being scalable and cost-effective in terms of data transfer and energy-usage on the phone. Figure 7 shows the use of mobile data stream mining techniques such as RA-SAX and LWC for remote health monitoring.[52,53]

- Logistics and Supply Chain applications often require real-time analysis and visualizations of where vehicles are for task allocation and dynamic scheduling. Typical applications in this domain include allocation of tasks to couriers for drop-off or pick-up, taxi allocation to passengers and so on. In Gillick et al.[54] the use of mobile data mining through real-time clustering and energy-efficient and clutter-adaptive visualizations is described and presented (as shown in Figure 8).
- Other applications where mobile and ubiquitous data mining has shown to be useful, include intelligent transportation systems[29,30], support for the mobile workforce such as mobile police personnel and mobile financial applications such as stock prices monitoring.[55] As such, there are many potential applications that can significantly benefit from mobile and ubiquitous data stream mining.

This section has presented a review of mobile and ubiquitous data stream mining from the perspectives of enabling strategies, algorithm design, toolkits for application development, and several application exemplars. The focus thus far has been on enabling and realizing mobile and ubiquitous data stream mining onboard a single mobile device. The next section presents the current research directions which have focused on multiple devices performing collaborative and distributed tasks using mobile or ubiquitous data stream mining.

PDM: A GENERAL FRAMEWORK FOR *AD HOC* MOBILE AND DISTRIBUTED DATA MINING

Owing to the networking capabilities of the small mobile and ubiquitous devices, one can make use of

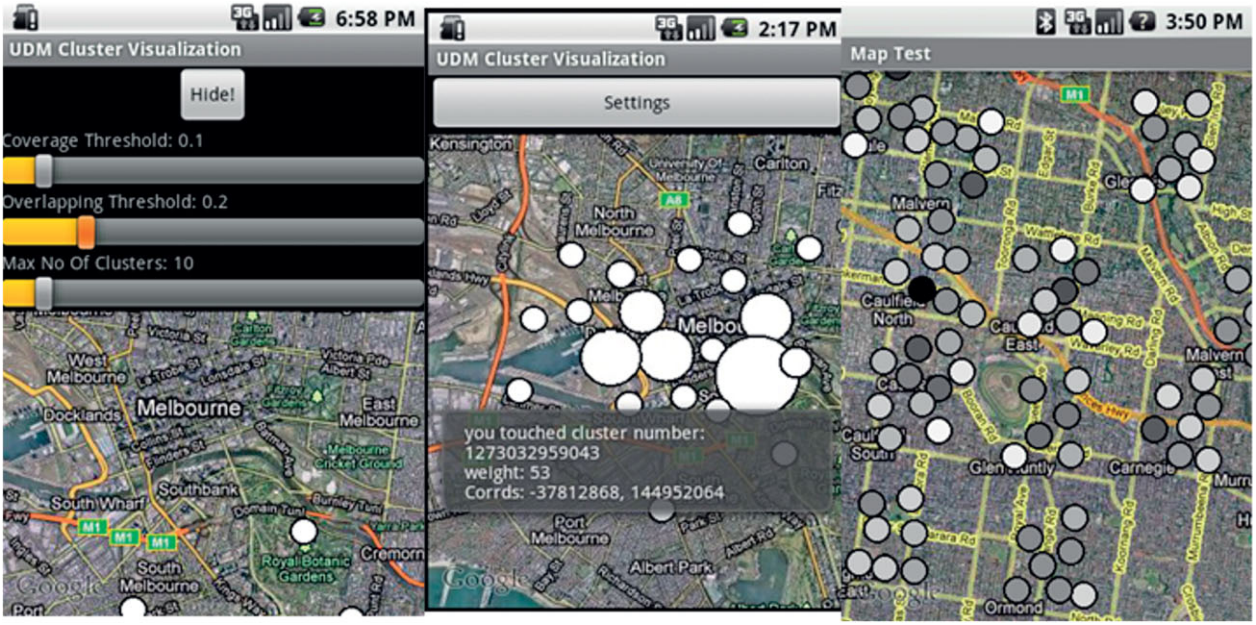

FIGURE 8 | Mobile data mining for logistics and Global Positioning System (GPS) analysis.

a whole *ad hoc* computing environment in order to perform significantly useful analysis tasks. This can be realized with the help of several established areas of study including: (1) data stream mining,[34,35] (2) mobile software agents,[56] and (3) programming for small devices. *PDM* makes use of these areas to enable collaborative mining of streaming data in the mobile environment.

A typical scenario for *ad hoc* data analysis would include a number of computationally capable devices like smartphones and smart sensors, and a number of applications that run onboard these devices. *PDM* is a framework and system architecture that realizes this *ad hoc* data analysis process. *PDM* is defined as the performance of mobile data stream mining in an *ad hoc* distributed computing environment adopting mobile software agents.[57]

PDM makes use of three types of software agents: (1) Agent Miners (AMs) run data mining algorithms on local data streams on the individual mobile device, (2) Mobile agent Resource Discoverers (MRDs) roam the network for resources, and (3) Mobile Agent Decision Makers (MADMs) hop through the network of smartphones to visit local AMs in order to complete a data mining task. Details of the roles of the different agents are more deeply discussed when describing the system architecture in Section *PDM Architecture*. The primary motive for developing this framework is to enable seamless collaboration among users of mobile data mining applications. Two constraints necessitate the distribution of the task resulting in a collaborative environment. First, the large amounts of data that challenge the state-of-the-art of our smartphones and embedded devices. Second, subscription fees that apply for this data to be streamed to the user's mobile device. Thus, collaborative mining addresses these constraints realizing the potential of this important application.

Two stimulating factors have motivated the adoption of the mobile software agent technology in this application. The first is the autonomous behavior that the agent framework supports. This is important to cope with the dynamic nature of the application of varying number of nodes and the data mining algorithms used. Communication efficiency as reported in Kargupta et al.[58] of using mobile software agents in distributed data mining has been the second factor.

PDM Architecture

An agent platform like the Java Agent Development Environment (*JADE*) Framework[59] is running on all the devices. A computational task is initiated by one of these devices by firing a number of mobile software agents roaming an *ad hoc* formed network. The agents discover the data sources, the computational capabilities of the devices that formed the network and the available applications onboard these devices. The agents in turn take a collective decision on the distribution of the processing subtasks to perform the initiated task according to several criteria like proximity to the source of data, the

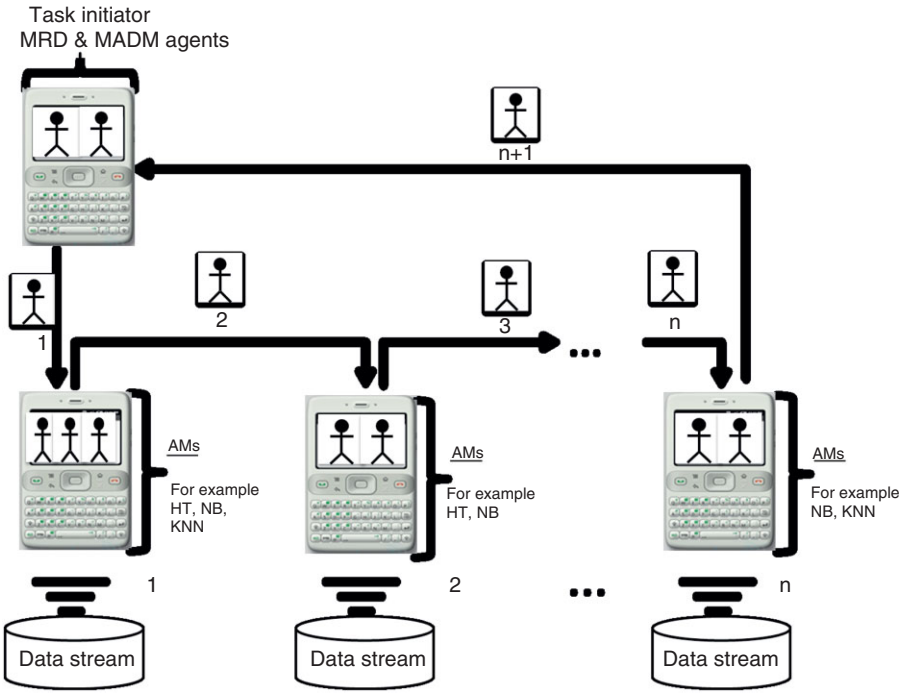

FIGURE 9 | The architecture of pocket data mining.

available applications to perform the process, and so on.

This generic scenario, when applied to collaborative data mining, includes mobile software agents of different types. These types can be identified as follows:

- *(Mobile) Agent Miners (AM)*: these agents are either distributed over the network when the mining task is initiated or are already located on the mobile device. They can be stationary agents that cannot move but be consulted by other agents, or they can be mobile and deployed dynamically.
- *Mobile Agent Resource Discoverers (MRD)*: these agents are used to explore the available computational resources, processing techniques, and data sources and derive a schedule for the third kind of agents, the *MADM* agents as described below.
- *Mobile Agent Decision Makers (MADM)*: these agents roam the network using the schedule derived by the *MRD* agent and consult the *AMs* to collaborate in reaching the final decision.

The architecture of the *PDM* framework is illustrated in Figure 9. From this point onward in the article, we shall use the terms *PDM architecture* and *PDM framework* interchangeably.

As Figure 9 shows, the data stream mining process runs onboard the users' smart mobile phones. As the data streams in the model are continuously updated to cope with the possible concept drift of the streaming environments. The abbreviation *HT* stands for *Hoeffding Tree* classifier which is also known as VFDT[22] (described in Section *Algorithms for Learning from Data Streams*), *NB* stands for the *Naive Bayes* classifier, and *K-NN* stands for the *K-Nearest Neighbor* classification algorithms. However, these are just a few examples of the algorithms implemented in the PDM framework. The process of stream mining is carried out using an *Agent Miner*, denoted as *AM*. *AMs* are distributed at the beginning of initiating the mining task. Some of these miners could be stationary and some others could be mobile. Stationary agents are instructed by the task initiator to mine the streaming data on the mobile device on which they are hosted without making any hops. However, the mobile agents could travel to one or more nodes in order to perform the mining task. The choice of using stationary or mobile agents relies on the nature of the task and the number of nodes involved in the processing. Typically, AMs are data stream classification techniques. But the use of other techniques is also possible depending on the required task.

If at any point in time, a user decides to use the models built using the different *AMs* on all the mobile phones to collaborate in finding the class label of a set

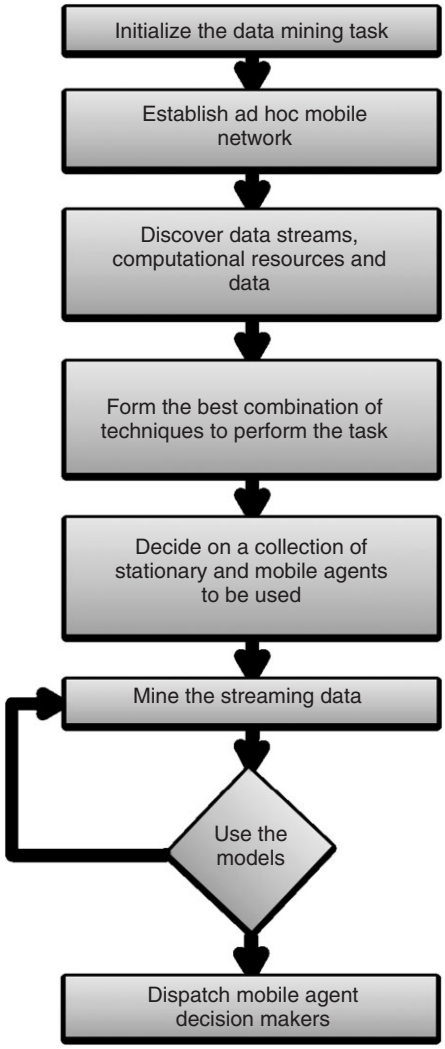

FIGURE 10 | A flow chart describing the collaborative data mining process implemented in the pocket data mining framework.

of unlabeled instances, then a *Mobile Agent Decision Maker* denoted as *MADM* is fired to visit the nodes consulting the models about the local class label. While the *MADM* agents are visiting the different nodes, an *MADM* agent may decide to terminate its itinerary, for example if there is a predominant class. This clearly makes the agent framework the suitable technology for this task. A simple flow chart of the process of collaborative data mining using the *PDM* architecture is given in Figure 10.

The *PDM* framework raises a number of research issues that are important to be addressed to optimize the task. The following is a list of these issues.

- The number of *AMs* that are decided by the task initiator or *MRD* agent. In an *ad hoc* environment, the number of participants may vary. Thus, it is important to involve the number of *AMs* that cover the largest number of features and data instances. For example, if the number of mobile phones in a setting is five, and two of these share the same instances and features and run the same data mining technique, only one of the two would be chosen for the task depending on other factors like computational resources and proximity to the data source.
- The number of *MADMs* that are decided by a participant. This is done when that participant is ready to use the output of the stream mining process for decision making. This number relies on the number of participants. Although it would be faster to fire a number of *MADMs* that is equivalent to the number of participants, this could be a burden on the communication network if the number of participants is large. Also, the decision relies on the different data mining techniques that run concurrently. If using a particular model is known to take longer time due to the structure the technique produces (rules, trees, etc.), another agent could visit two different nodes while one agent is consulting the node with the longer runtime. Thus, an optimum number of *MADMs* has to be decided.
- The combination of data mining techniques to be used is essential to ensure that the built models are of optimum accuracy. Once the *MRDs* have found out the different techniques and the data sources in the network, stream mining techniques that will be used are decided. This is done according to the request made by the task initiator. For example, if a classification task is initiated and some nodes host more than one classification technique, it is important to decide whether the node runs the two techniques on the same streaming data, or runs one of them according to availability of resources and importance of the features in the decision making. It is also essential to decide whether these techniques would be stationary or mobile. If the mobility of agents is decided, it is important to decide the timing of the migration from one node to the other.
- The combination of features and data instances that is used as input to the data mining algorithm. This decision relies on what has been discovered by the mobile agent resource discoverers. If some features and instances are shared among classifiers, it is important to decide whether to use the shared features and instances by two

or more different classifiers, or to use disjoint subsets of the data to accelerate the process.

- The decision on whether each node is required to adapt to resource availability is important. Some nodes would be in a better position to cope with high speed streaming data due to their high performance computational power. However, some other would need to adapt. The *Granularity-based* approach developed by Gaber[37] is able to adjust the algorithm settings in real-time to change the consumption pattern of the algorithm to cope with low availability of resources. The decision of using one or more of the algorithm granularity settings will be taken in the light of the running techniques and the configuration of the mobile phone.

PDM Implementation

The competition in the smartphone industry is pushing the rapid development of operating systems such as *Apple's iOS* for the *iPhone* or *Google's Android*. The *PDM* project has targeted the *Android* operating system which is adopted by a wide range of smartphones. Nevertheless *PDM* is extensible to work on further operating systems in the future.

To ensure interoperability and to ease the development of mobile and distributed agents, frameworks such as *Grasshopper*[60] and *JADE*[59] have been developed. The basic requirements for such a framework's use in *PDM* are its ability to move agents such as the *MADM* between mobile devices and computers. The well-known *JADE* framework just fits these requirements. One of its other strong points is that it is open source, and written in *Java*. Agents in the *JADE* are hosted in so called containers between which they can freely move. In *PDM* each mobile device runs a *JADE* container and each *PDM* agent lives in such a container. A *PDM* agent hops to a different mobile device by logically moving from one container to another. In order to support the communication and mobility of agents, *JADE* provides a set of special agents that are hosted in the main container. Those agents are the *Agent Management System* (*AMS*) and the *Directory Facilitator* (*DF*). The *AMS* provides the names for the created agents and makes sure that there are no duplicated names; and the *DF* provides a communication service to the agents so they can communicate between each other.

For the implementation of the *AM* agents, the *MOA*[61] tool which is based on the *WEKA* workbench[62] has been used. Its libraries are well established in the data stream mining community.

FIGURE 11 | Android version of pocket data mining running on four Android smartphones and one Windows 7 laptop.

Figure 11 shows a typical setup of *PDM* with five devices connected. As it can be seen, four of the connected devices are different Android smartphones and one of the devices is a laptop hosting a Windows 7 operating system, which shows that the system can integrate PCs as well as Android smartphones. Figure 11 has been captured from a demonstration video about PDM which can be found at http://www.youtube.com/watch?v=MOvlYxmttkE.

Figure 12 shows the graphical user interfaces of *PDM* on *Android* smartphones. The left-hand side of the figure shows the main screen of the *PDM* implementation. Here the user can decide if he wants to connect to an existing *PDM* network or if he wants to setup a new one. The middle part of the figure shows the main screen for setting up an *Agent Miner*. The user has to select the algorithm to be used (in this case *Hoeffding Trees* have been selected), the data source and several parameters such as the probability with which an incoming data instance is selected as test or as training instance. The right-hand side of the figure shows the main *GUI* for creating a *Mobile Agent Decision Maker*. The user has to provide several parameters, for example the location of the test data.

In general some of the parameters displayed in the *GUI* in Figure 12 are for debugging purposes and are planned to be removed subsequently in future versions of *PDM*.

Ad hoc networks can be built by selecting one or several smartphones as wireless hotspots. *Android* currently supports up to eight devices connected. However, several hotspots could potentially be created, and thus more phones could be used and the coverage be increased. *PDM* is not limited to a

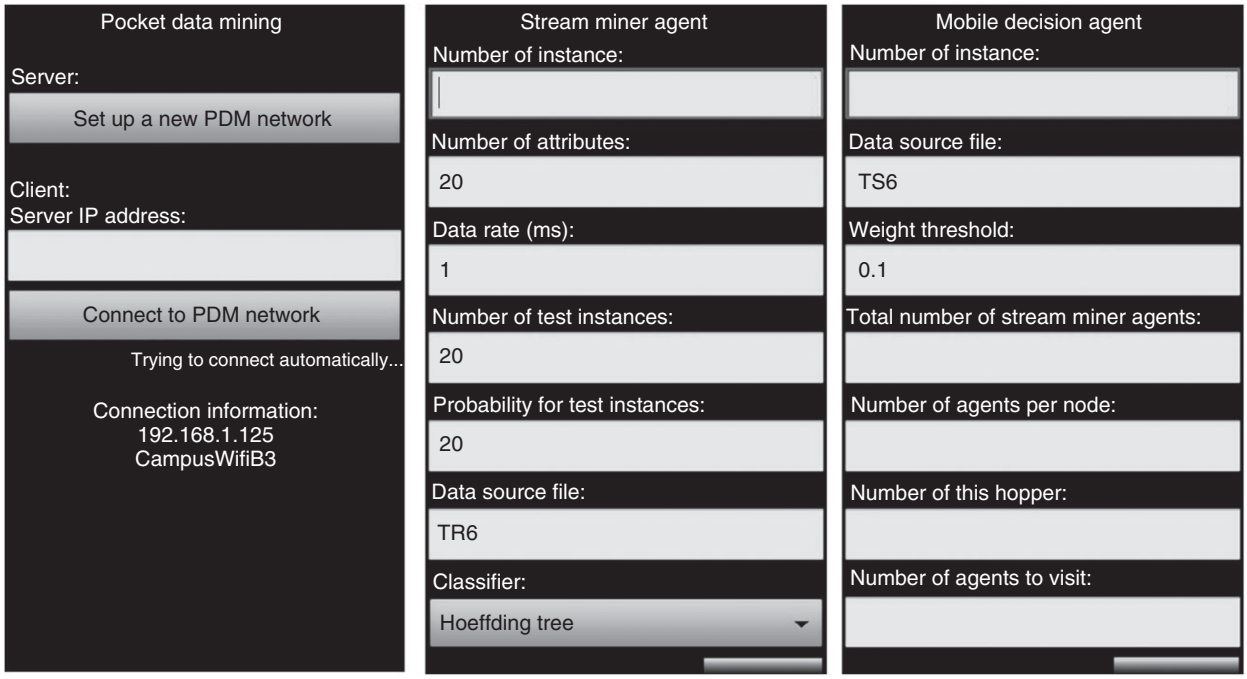

FIGURE 12 | The left-hand side of the figure shows the start screen of *pocket data mining*. The screen displayed in the middle shows the main screen used for creating a new Agent Miner. The right-hand side of the figure shows the main screen for creating a *Mobile Agent Decision Maker*.

smart phone as wireless hotspot, it can also make use of other means of communication such as a wireless router, *Wi-Fi*, *Bluetooth*, and so on.

Extensive experimental studies assessing the performance of *PDM* have been reported in Stahl et al.[63] Homogeneous and heterogeneous *Agent Miners* have been used in the experiments adopting *Naive Bayes* and *Hoeffding Trees* stream classifiers.[61] A great potential for the framework has been shown especially with having eight nodes (connected ubiquitous devices).

PDM's new niche of distributed data mining is expected to benefit from recent advances in smart phone and data stream mining technology. Future directions in this research can explore the numerous alternatives of collaborative mining techniques and strategies. These include varying the mining techniques, the sharing of features and instances of the data among nodes, and the distribution of the roles among agents that would yield the highest accuracy. *PDM* as a distributed data mining framework can also benefit from the work by Talia et al.,[64] where the machine learning tool Weka[65] has been extended to allow running the mining algorithms as web services. The system (termed *Weka4WS*) adopted the Web Services Resource Framework (WSRF) for the implementation of Grid services. The GUI of the system allows local and remote execution of data mining techniques. Experimental results of the system have shown the high performance of Grid-based distributed data mining. Despite the fact that *Weka4WS* did not utilize mobile software agents, experiences gained from running the different data mining techniques in a distributed mode can greatly help optimizing the performance of *PDM*.

Having discussed the generic framework to perform distributed *ad hoc* data stream mining; *PDM*, an important issue in this collaborative context is to consider the context in which each model performs, and consequently affects the subspace of features or attributes that each model can rely on. The following section will discuss the issue in details, providing a solution, *Coll-Stream*.

COLLABORATIVE DATA STREAM MINING IN UBIQUITOUS ENVIRONMENTS

Mobility of users and the number of them open the door for another important research issue; context-awareness and collaboration. This section deals with collaborative data stream mining onboard mobile devices. The goal is to learn an anytime classification model that represents the underlying concept from a stream of labeled records. Such model is used to predict the label of the incoming unlabeled records available in the device. However, it is common for

the underlying concept of interest to change over time and sometimes the labeled data available in the device is not sufficient to guarantee the quality of the results.[66] Therefore, we describe *Coll-Stream*,[48–50,67] a framework that exploits the knowledge available in other devices to collaboratively improve the accuracy of local predictions.

The data mining problem is assumed to be the same in all the devices; however, the feature space and the data distributions may not be static, as assumed in traditional data mining approaches.[68,69] We are interested in understanding how the knowledge available in the community can be integrated to improve local predictive accuracy in a ubiquitous data stream mining scenario.

As an illustrative example, collaborative spam filtering[70] is one of the possible applications for the collaborative learning approach. Each ubiquitous device learns and maintains a local filter that is incrementally learned from a local data stream based on features extracted from the incoming mails. In addition, user usage patterns and feedback are used to supervise the filter that represents the target concept (i.e., the distinction between spam and ham). The ubiquitous devices collaborate by sharing their knowledge with others, which can improve their local device predictive accuracy. Furthermore, the dissemination of knowledge is faster, as peers new to the mining task, or that have access to fewer labeled records, can anticipate spam patterns that were observed in the community, but not yet locally. Moreover, the privacy and computational issues that would result from sharing the original mail are minimized, as only the filters are shared. Consequently, this increases the efficiency of the collaborative learning process.

However, many challenges arise from this scenario, the two major ones are:

1. How the knowledge from the community can be exploited to improve local predictiveness; and
2. How to adapt to changes in the underlying concept.

We will describe *Coll-Stream*,[48–50] that uses an incremental ensemble approach where the models from the community are selected and weighted based on their local accuracy for different partitions of the instance space. Such technique is motivated by the possible conflicts among the community models. The technique allows to exploit the fact that each model can be accurate only for certain subspaces, where its expertise matches the local underlying concept.

Collaborative Data Stream Mining

In collaborative and distributed data mining, the data are partitioned and the goal is to apply data mining to different, usually very small and overlapping, subsets of the entire data.[57,71] In *Coll-Stream*, the goal is not to learn a global concept, but to learn from other devices their concepts, while maintaining a local or subjective point of view. Wurst and Morik[72] explore this idea by investigating how communication among peers can enhance the individual local models without aiming at a common global model. The motivation is similar to what is proposed in domain adaptation[73] or transfer learning.[74] However, these assume a batch scenario. When the mining task is performed in a ubiquitous environment,[2,75] an incremental learning approach is required.

In ubiquitous data stream mining, the feature space of the records that occur in the data stream may change over time[68] or be different among devices.[72] For example, in a stream of documents where each word is a feature, it is impossible to know in advance which words will appear over time, and thus what is the best feature space to represent the documents with. Using a very large vocabulary of words is inefficient, as most of the words will likely be redundant and only a small subset of words is finally useful for classification. Over time, it is also likely that new important features appear and that previously selected features become less important, which brings change to the subset of relevant features. Such change in the feature space is related to the problem of concept drift, as the target concept may change due to changes in the predictiveness of the available features.

The following sections describe the details of *Coll-Stream* and Figure 13 illustrates the collaborative learning process.

Collaborative Data Stream Mining in Ubiquitous Environments Using Dynamic Classifier Selection

Coll-Stream is a collaborative learning approach for ubiquitous data stream mining that combines the knowledge of different models from the community.

There is a large number of ensemble methods to combine models, which can be roughly divided into:

1. Voting methods, where the class that gets more votes is chosen.[76–78]
2. Selection methods, where the 'best' model for a particular instance is used to predict the class label.[79,80]

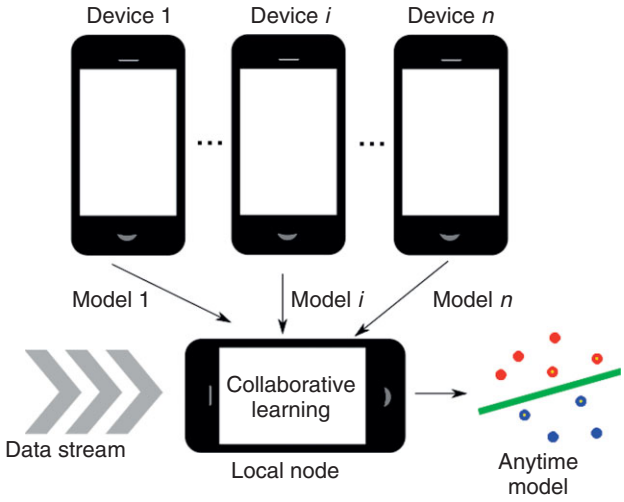

FIGURE 13 | Collaborative learning process.

The *Coll-Stream* is a selection method that partitions the instance space X into a set of regions R. For each region, an estimate of the models accuracy is maintained over a sliding window. This estimated value is updated incrementally as new labeled records are observed in the data stream or new models are available. This process can be considered a meta-learning task where we try to learn for each model from the community how it best represents the local underlying concept for a particular region $r_i \in R$. When *Coll-Stream* is asked to label a new record $\vec{x}_i$, the best model is used. The best model is considered to be the one that is more accurate for the partition r_i that contains the new record. The accuracy for a region r_i is the average accuracy for each partition of its attributes. For r_{15} in Figure 14, we average the accuracy for value $V1$ of attribute $A1$ and value $V5$ of attribute $A2$. The accuracy is the number of correct predictions divided by the total number of records observed, as is illustrated in Figure 14. The next section explains how the regions are created using the attribute values.

Creating Regions

An important part of *Coll-Stream* is to learn for each region of the instance space X which model m_j performs better. This way m_j predictions can be used with confidence to classify incoming unlabeled records that belong to that particular region.

The instance space can be partitioned in several ways. Here we follow the method used by Zhu et al.,[79] where the partitions are created using the different values of each attribute. For example, if an attribute has two values, two estimators of how the classifiers perform for each value are kept. If the attribute is numeric, it is discretized and the regions use the values that result from the discretization process. This method has shown good results and it represents a natural way of partitioning the instance space. However, there is an increased memory cost associated with a larger number of regions. To minimize this cost the regions can be partitioned into higher granularity ones, aggregating attribute values into a larger partition. This is illustrated in Figure 14, where the values $V4$ and $V5$ of attribute $A1$ are grouped into regions $r41$ to $r45$. Figure 14 illustrates the training and classification procedures of *Coll-Stream*.

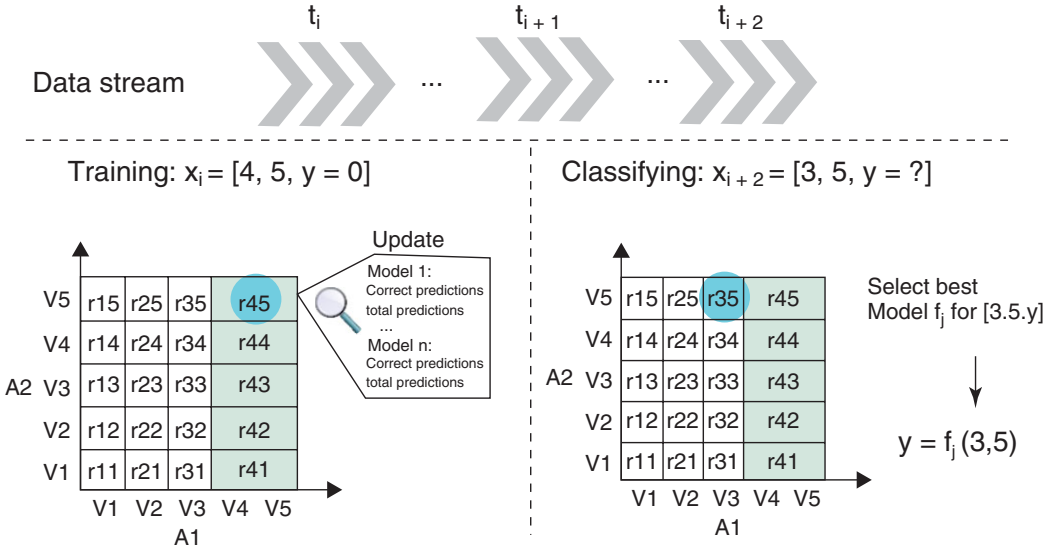

FIGURE 14 | Coll-Stream: training and classifying.

Variations

Some variations of the *Coll-Stream* approach can be considered:

- **Multiple classifier selection** If more than one model is selected, their predictions are weighed according to the corresponding accuracy for region r_i, and the record to be labeled gets the class with the highest weighted vote. This is similar to weighted majority voting but with a variable number of classifiers, where the weights are calculated in relation to the region that contains the unlabeled record to classify.
- **Feature weighting** The models used from the community can represent a heterogeneous feature space as each one is trained according to a different data stream DS_d. One possible variation is for each device to measure feature relevance. Then at the time of classification the accuracy estimates for each region are weighted according to the feature weight for that region. The predictive score of each feature can be computed using popular methods such as, the information gain, χ^2 or mutual information.[68] However, these must be calculated incrementally given the data stream scenario where this approach is framed. Moreover, this takes into account that features that were relevant in the past can become irrelevant at some point in the future for a different target concept.
- **Using local base learner** One base learner that is trained using the available records in the device can be always part of the ensemble. This integration is simple as it only requires an additional step of training the classifier when a new record arrives in addition to updating the ensemble estimates for the new record region.
- **Resource awareness** Resource-awareness is an important issue in ubiquitous data stream mining.[2] In such a dynamic ubiquitous usage scenario, it is common for *Coll-Stream* method to receive too much knowledge from the community over time. In such situations we propose to discard past models that have the lowest performance and allow the integration of new models.
- **Feature selection** Feature selection is used to reduce the size of the models kept. This process can be executed before the models are shared, in order to additionally reduce the communication costs associated with transferring the models between devices. When the model is shared with other devices, only its most predictive features are used. For example in the case where the *Naive Bayes* algorithm is used as base learner, only the corresponding estimators for the selected predictive attributes given the class take part in the shared model. The features predictiveness is evaluated and these are then selected. However, many selection methods can be used for this task and according to the application some methods may be superior to other. Some simple possibilities that have been studied in Gomes et al.[48–50] are as follows:
  - Fixed N, select the top N highly scored features for each model.
  - Fixed threshold, which defines the cut point between predictive and irrelevant features.
  - Adaptive threshold, that adaptively defines the threshold. The method uses the desired percentile of the scores given by the feature evaluation method as a threshold.
- **Context-awareness** Context awareness is an important part of ubiquitous computing.[40] Most ubiquitous applications concepts are associated with context, this means that they may reappear when a similar context occurs. For example, a weather prediction model usually changes according to the seasons. The same applies with product recommendations or text filtering models where the user interest and behavior might change over time due to fashion, economy, spatial–temporal situation, or any other context.[81] This has motivated some works[82–84] to analyze how to use context to track concept changes. Finally, a context-aware version of *Coll-Stream*, named *CC-Stream*[83,84] takes into account the context similarity in relation to the current context when performing model selection.

CONCLUSIONS AND FUTURE DIRECTIONS

This work presents the emerging state-of-the-art in developing the next generation of mobile and ubiquitous data stream processing algorithms, systems, and applications. The continuous flow of data produced in mobile devices requires data streams techniques for querying, analyzing, and mining for successful ubiquitous data mining. This article includes background theory of data stream mining, theoretical foundations of mobile or ubiquitous data stream mining and explanation of the algorithms that represent

state-of-the-art in this area, as well as an overview of real-world applications and case studies. From the state-of-the-art to a more of current directions, the article also discussed two recent developments in this area, namely, *PDM* and *Coll-Stream*, addressing *ad hoc* flexible distributed data stream mining, and context-aware collaborative mining of data streams, respectively.

The capabilities of smartphones are increasing everyday. More and more sophisticated devices with embedded sensors are collecting data from user activities. Smartphones know where we are, who are we with, and what is around us. This opens an opportunity for novel applications that analyze these continuous information to make our lives easier.

Wearable devices like smart watches and Big Data analytics techniques open the door for a number of research areas. These include:

- *Personalized health*: the wearable devices can provide the user with important physiological measurements. Applications that continuously monitor and analyze this data can help users identify health-related risks timely and effectively. Interaction between data collected using the wearable devices and other mobile devices can be an important application for the *PDM* framework.
- *Mobile Big Data analytics*: Big Data techniques are tailored toward high performance computing facilities like the cloud. With *Big Data* pushed to the mobile environment, techniques that are tailored to operate in this environment are required. This area is not currently explored in the literature. However, we expect that it will grow rapidly over the next few years.
- *Mobile crowdsourcing of streaming knowledge*: the availability of up-to-date models generated from streaming data collected by individuals can turn to a powerful collective wisdom. Users may decide to share their models to get a more powerful ensemble of models. *Coll-Stream* presented in this survey article is a one step toward realization of this direction.

ACKNOWLEDGMENTS

We acknowledge the contribution of the two visiting students to the University of Portsmouth, Oscar Campos and Victor Mandujano in porting the system from the desktop to the *Android* smartphones, realizing the first implementation in the mobile environment. This work was supported by research project *Knowledge Discovery from Ubiquitous Data Streams* (PTDC/EIA-EIA/098355/2008), and by the European Regional Development Fund through the COMPETE Program, by the Portuguese Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-022701. J. Gama also acknowledges the support of the European Commission through the project MAESTRA (Grant number ICT-2013-612944).

REFERENCES

1. Gama J, May M. Ubiquitous knowledge discovery. *Intell Data Anal* 2011, 15:1.
2. Gaber M, Krishnaswamy S, Zaslavsky A. Ubiquitous data stream mining. In: *Current Research and Future Directions Workshop Proceedings held in conjunction with The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Sydney, Australia. Citeseer, 2004.
3. Muthukrishnan S. *Data Streams: Algorithms and Applications*. Hanover, MA: Now Publishers; 2005.
4. Gama J. *Knowledge Discovery from Data Streams. Data Mining and Knowledge Discovery*. Atlanta, US: Chapman & Hall CRC Press; 2010.
5. Chaudhry N. *Stream Data Management, Chapter Introduction to Stream Data Management*. New York, NY: Springer; 2005.
6. Babu S, Widom J. Continuous queries over data streams. *SIGMOD Rec* 2001, 30:109–120.
7. Aggarwal C, ed. *Data Streams—Models and Algorithms*. New York, NY: Springer; 2007.
8. Motwani R, Raghavan P. *Randomized Algorithms*. New York, NY: Cambridge University Press; 1997.
9. Chakrabarti A, Ba KD, Muthukrishnan S. Estimating entropy and entropy norm on data streams. In: *STACS: 23rd Annual Symposium on Theoretical Aspects of Computer Science*, Marseille, France, 2006, 196–205.
10. Manku GS, Motwani R. Approximate frequency counts over data streams. In: *Proceedings of 28th International Conference on Very Large Data Bases*, Hong Kong; 2002, 346–357.
11. Cormode G, Muthukrishnan S, Zhuang W. Conquering the divide: continuous clustering of distributed data

streams. In: *ICDE: Proceedings of the International Conference on Data Engineering*, Istanbul, Turkey; 2007, 1036–1045.

12. Flajolet P, Martin GN. Probabilistic counting algorithms for data base applications. *J Comput Syst Sci* 1985, 31:182–209.
13. Babcock B, Datar M, Motwani R. Sampling from a moving window over streaming data. In: *Proceedings of the Annual ACM SIAM Symposium on Discrete Algorithms*, San Francisco, USA, SIAM, 2002, 633–634.
14. Datar M, Gionis A, Indyk P, Motwani R. Maintaining stream statistics over sliding windows. In: Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, USA, Society for Industrial and Applied Mathematics; 2002, 635–644.
15. Bifet A, Gavaldà R. Kalman filters and adaptive windows for learning in data streams. In: Todorovski L, Lavrac N, eds. *Proceedings of the 9th Discovery Science, Volume 4265 of Lecture Notes Artificial Intelligence*. Barcelona, Spain: Springer; 2006, 29–40.
16. Aggarwal C. On biased reservoir sampling in the presence of stream evolution. In: *Proceedings International Conference on Very Large Data Bases*, Seoul, Korea, 2006, 607–618.
17. Vitter JS. Random sampling with a reservoir. *ACM Trans Math Softw* 1985, 11:37–57.
18. Tatbul N, Çetintemel U, Zdonik S, Cherniack M, Stonebraker M. Load shedding in a data stream manager. In: *Proceedings of the 29th International Conference on Very Large Data Bases*, vol 29, VLDB '03, VLDB Endowment, 2003, 309–320.
19. Gilbert AC, Kotidis Y, Muthukrishnan S, Strauss M. Surfing wavelets on streams: one-pass summaries for approximate aggregate queries. In: *VLDB*, Rome, Italy, 2001, 79–88.
20. Alon N, Matias Y, Szegedy M. The space complexity of approximating the frequency moments. *J Comput Syst Sci* 1999, 58:137–147.
21. Cormode G, Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications. *J Algorithms* 2005, 55:58–75.
22. Domingos P, Hulten G. Mining high-speed data streams. In: Parsa I, Ramakrishnan R, Stolfo S, eds. *Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining*. Boston, MA: ACM Press; 2000, 71–80.
23. Rodrigues PP, Gama J, Pedroso JP. Hierarchical clustering of time series data streams. *IEEE Trans Knowl Data Eng* 2008, 20:615–627.
24. Hulten G, Spencer L, Domingos P. Mining time-changing data streams. In: *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA: ACM Press; 2001, 97–106.
25. Ikonomovska E, Gama J, Dzeroski S. Learning model trees from evolving data streams. *Data Min Knowl Discov* 2011, 23:128–168.
26. Gama J, Kosina P. Learning decision rules from data streams. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI; 2011, 1255–1260.
27. Sharfman I, Schuster A, Keren D. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans Database Syst* 2007, 32:301–312.
28. May M, Saitta L. *Ubiquitous Knowledge Discovery*. Springer; 2010, LNAI 6202.
29. Kargupta H, Bhargava R, Liu K, Powers M, Blair P, Bushra S, Dull J, Sarkar K, Klein M, Vasa M, et al. VEDAS: a mobile and distributed data stream mining system for real-time vehicle monitoring. In: Proceedings of SIAM International Conference on Data Mining, Volume 334, 2004.
30. Kargupta H, Joshi A, Sivakumar K, Yesha Y. *Data Mining: Next Generation Challenges and Future Directions*. New York, NY: AAAI Press and MIT Press; 2004.
31. Kargupta H, Park B-H. Mining decision trees from data streams in a mobile environment. In: *IEEE International Conference on Data Mining*. San Jose, CA: IEEE Computer Society; 2001, 281–288.
32. Chen R, Sivakumar K, Kargupta H. Collective mining of Bayesian networks from heterogeneous data. *Knowl Inf Syst J* 2004, 6:164–187.
33. Kargupta H, Park B-H, Dutta H. Orthogonal decision trees. *IEEE Trans Knowl Data Eng* 2006, 18:1028–1042.
34. Gaber MM, Krishnaswamy S, Zaslavsky A. On-board mining of data streams in sensor networks. In: *Advanced Methods for Knowledge Discovery from Complex Data, Springer;* 2005, 307–335.
35. Gaber MM, Zaslavsky A, Krishnaswamy S. Mining data streams: a review. *ACM SIGMOD Rec* 2005, 34:18–26.
36. Gaber M, Yu P. A holistic approach for resource-aware adaptive data stream mining. *New Generation Comput* 2006, 25:95–115.
37. Gaber MM. Data stream mining using granularity-based approach. In: *Foundations of Computational, Intelligence*, vol. 6. Ohmsha, Ltd.: Springer; 2009, 47–66.
38. Haghighi PD, Gaber MM, Krishnaswamy S, Zaslavsky A. Situation-aware adaptive processing (saap) of data streams. In: *Pervasive Computing*. New York, NY: Springer; 2010, 313–338.
39. Haghighi PD, Zaslavsky A, Krishnaswamy S, Gaber MM, Loke S. Context-aware adaptive data stream mining. *Intell Data Anal* 2009, 13:423–434.
40. Padovitz A, Loke SW, Zaslavsky A. Towards a theory of context spaces. In: *Proceedings of the Second*

IEEE Annual Conference on Pervasive Computing and Communications Workshops. Los Alamitos, CA: IEEE; 2004, 38–42.

41. Domingos P, Hulten G. A general method for scaling up machine learning algorithms and its application to clustering. In: *ICML*, 2001, 106–113.
42. Lin J, Keogh E, Wei L, Lonardi S. Experiencing sax: a novel symbolic representation of time series. *Data Min Knowl Discov* 2007, 15:107–144.
43. Krishnaswamy S, Gaber M, Harbach M, Hugues C, Sinha A, Gillick B, Haghighi P, Zaslavsky A. Open mobile miner: a toolkit for mobile data stream mining. In: *Demo and Short paper, the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2009*, Paris, 2009. Available at: http://eprints.port.ac.uk/4140/1/D02-kdd09demo.pdf. (Accessed May 2012).
44. Haghighi P. D, Krishnaswamy S, Zaslavsky A, Gaber M. M, Sinha A, Gillick B. Open mobile miner: a toolkit for building situation-aware data mining applications. *J Organ Comput Electr Commer* 2009, 23:224–248.
45. Sherchan W, Jayaraman PP, Krishnaswamy S, Zaslavsky A, Loke S, Sinha A. Using on-the-move mining for mobile crowdsensing. In: *IEEE 13th International Conference on Mobile Data Management (MDM), 2012*. Washington, DC: IEEE; 2012, 115–124.
46. Gomes JB, Krishnaswamy S, Gaber MM, Sousa PA, Menasalvas E. Mars: a personalised mobile activity recognition system. In: *2012 IEEE 13th International Conference on Mobile Data Management (MDM)*, IEEE; 2012, 316–319.
47. Gomes JB, Krishnaswamy S, Gaber MM, Sousa PA, Menasalvas E. Mobile activity recognition using ubiquitous data stream mining. In: *Data Warehousing and Knowledge Discovery*. Vienna, Austria: Springer; 2012, 130–141.
48. Gomes J, Gaber M, Sousa P, Menasalvas E. Mining recurring concepts in a dynamic feature space. *IEEE Trans Neural Netw Learn Syst* 2013, 24:1–16.
49. Gomes JB, Medhat Gaber M, Sousa P, Menasalvas Ruiz E. Collaborative data stream mining in ubiquitous environments using dynamic classifier selection. *Int J Inf Technol Decis Making* 2013, 12:1–12.
50. Gomes JB, Phua C, Krishnaswamy S. Where will you go? mobile data mining for next place prediction. In: *Data Warehousing and Knowledge Discovery*. Prague, Czech Republic: Springer; 2013, 146–158.
51. Chong SK, Gaber MM, Krishnaswamy S, Loke SW. Energy conservation in wireless sensor networks: a rule-based approach. *Knowl Inf Syst* 2011, 28:579–614.
52. Tayebi H, Krishnaswamy S, Waluyo AB, Sinha A, Gaber MM. Ra-sax: resource-aware symbolic aggregate approximation for mobile ECG analysis. In: *12th IEEE International Conference on Mobile Data Management (MDM), 2011*, vol. 1. Milan, Italy: IEEE; 2011, 289–290.
53. Sinha A, Tayebi H, Krishnaswamy S, Waluyo AB, Gaber MM. Resource-aware ecg analysis on mobile devices. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*. Taichung, Taiwan: ACM; 2011, 1012–1013.
54. Gillick B, AlTaiar H, Krishnaswamy S, Liono J, Nicoloudis N, Sinha A, Zaslavsky A, Gaber MM Clutter-adaptive visualization for mobile data mining. In: *2010 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2010, 1381–1384.
55. Kargupta H, Park B, Pittie S, Liu L, Kushraj D, Sarkar K. MobiMine: monitoring the stock market from a PDA. *ACM SIGKDD Explor Newsl* 2002, 3:37–46.
56. Page J, Padovitz A, Gaber M. Mobility in agents, a stumbling block or a building block? In: *Proceedings of Second International Conference on Intelligent Computing and Information Systems*, 2005.
57. Stahl F, Gaber MM, Bramer M, Yu PS. Pocket data mining: towards collaborative data mining in mobile computing environments. In: *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2010*, vol. 2. Arras, France: IEEE; 2010, 323–330.
58. Kargupta H, Hamzaoglu I, Stafford B. Scalable, distributed data mining-an agent architecture. In: *Proceedings of Third International Conference on Knowledge Discovery and Data Mining*, 1997, 211–214.
59. Bellifemine FL, Caire G, Greenwood D. *Developing Multi-Agent Systems With JADE*, vol. 7. Wiley; 2007.
60. Bäumer C, Magedanz T. Grasshopper: a mobile agent platform for active telecommunication networks. In: *Intelligent Agents for Telecommunication Applications*. Stockholm, Sweden: Springer; 1999, 19–32.
61. Bifet A, Holmes G, Kirkby R, Pfahringer B. Moa: Massive online analysis. *J Mache Learn Res* 2010, 99:1601–1604.
62. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The weka data mining software: an update. *ACM SIGKDD Explor Newsl* 2009, 11:10–18.
63. Stahl F, Gaber MM, Aldridge P, May D, Liu H, Bramer M, Philip SY. Homogeneous and heterogeneous distributed classification for pocket data mining. In: *Transactions on Large-Scale Data-and Knowledge-Centered Systems V*. New York, NY: Springer; 2012, 183–205.
64. Talia D, Trunfio P, Verta O. The weka4ws framework for distributed data mining in service-oriented grids. *Concurrency Comput: Pract Exp* 2008, 20:1933–1951.
65. Frank E, Hall MA, Holmes G, Kirkby R, Pfahringer B. Weka—a machine learning workbench for data mining. In: *The Data Mining and Knowledge Discovery Handbook;* 2005, 1305–1314.
66. Masud MM, Gao J, Khan L, Han J, Thuraisingham B. A practical approach to classify evolving data streams:

training with limited amount of labeled data. In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. Washington, DC: IEEE Computer Society; 2008, 929–934.

67. Gomes JB. Learning recurring concepts from data streams in ubiquitous environments. PhD Thesis, Technical University of Madrid, 2011.
68. Katakis I, Tsoumakas G, Vlahavas I. On the utility of incremental feature selection for the classification of textual data streams. In: *Proceedings of the 10th Panhellenic conference on Advances in Informatics, PCI'05*. Berlin, Heidelberg: Springer-Verlag; 2005, 338–348.
69. Tsymbal A. *The Problem of Concept Drift: Definitions and Related Work*. Dublin, Ireland: Computer Science Department, Trinity College; 2004.
70. Cortez P, Lopes C, Sousa P, Rocha M, Rio M. Symbiotic data mining for personalized spam filtering. In: *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, 2009. WI-IAT'09, Volume 1, IEEE, 2009, 149–156.
71. Datta S, Bhaduri K, Giannella C, Wolff R, Kargupta H. Distributed data mining in peer-to-peer networks. *IEEE Internet Comput* 2006, 10:18–26.
72. Wurst M, Morik K. Distributed feature extraction in a p2p setting–a case study. *Future Generation Comput Syst* 2007, 23:69–75.
73. Daumé H III, Marcu D. Domain adaptation for statistical classifiers. *J Artif Int Res* 2006, 26:101–126.
74. Pan S, Yang Q. A survey on transfer learning. *IEEE Trans Knowl Data Eng* 2010, 22:1345–1359.
75. Hotho A, Pedersen R, Wurst M. Ubiquitous data. *Ubiquitous Knowl Discov* 2010:61–74.
76. Street W, Kim Y. A streaming ensemble algorithm (SEA) for large-scale classification. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM; 2001, 377–382.
77. Wang H, Fan W, Yu P, Han J. Mining concept-drifting data streams using ensemble classifiers. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York: ACM; 2003, 226–235.
78. Kolter J, Maloof M. Dynamic weighted majority: an ensemble method for drifting concepts. *J Mach Learn Res* 2007, 8:2755–2790.
79. Zhu X, Wu X, Yang Y. Effective classification of noisy data streams with attribute-oriented dynamic classifier selection. *Knowl Inf Syst* 2006, 9:339–363.
80. Tsymbal A, Pechenizkiy M, Cunningham P, Puuronen S. Dynamic integration of classifiers for handling concept drift. *Inf Fusion* 2008, 9:56–68.
81. Widmer G, Kubat M. Learning in the presence of concept drift and hidden contexts. *Mach Learn* 1996, 23:69–101.
82. Gomes JB, Sousa PA, Menasalvas E. Tracking recurrent concepts using context. *Intell Data Anal* 2012, 16:803–825.
83. Gomes J, Menasalvas E, Sousa P. Learning recurring concepts from data streams with a context-aware ensemble. In: *ACM Symposium on Applied Computing*. TaiChung, Taiwan: ACM; 2011, 994–999.
84. Gomes JB, Gaber MM, Sousa PA, Menasalvas E. Context-aware collaborative data stream mining in ubiquitous devices. In: *Advances in Intelligent Data Analysis X*. Porto, Portugal: Springer; 2011, 22–33.

