Homogeneous and Heterogeneous Distributed Classification for Pocket Data Mining

Frederic Stahl[1], Mohamed Medhat Gaber[1], Paul Aldridge[1], David May[1] Han Liu[1], Max Bramer[1], and Philip S. Yu[2]

[1] School of Computing, University of Portsmouth
Portsmouth, PO1 3HE, UK
[2] Department of Computer Science, University of Illinois at Chicago
851 South Morgan Street, Chicago, IL 60607-7053, USA

Abstract. Pocket Data Mining (*PDM*) describes the full process of analysing data streams in mobile ad hoc distributed environments. Advances in mobile devices like smart phones and tablet computers have made it possible for a wide range of applications to run in such an environment. In this paper, we propose the adoption of data stream classification techniques for *PDM*. Evident by a thorough experimental study, it has been proved that running heterogeneous/different, or homogeneous/similar data stream classification techniques over vertically partitioned data (data partitioned according to the feature space) results in comparable performance to batch and centralised learning techniques.

1 Introduction

Thanks to continuing advances on mobile computing technology, more and more data mining applications are running on mobile devices such as 'Tablet PCs', smart phones and Personal Digital Assistants (PDAs). The ability to make phone calls and send SMS messages nowadays seems to be merely an additional feature rather than the core functionality of a smart phone. Smart phones offer a wide variety of sensors such as cameras and gyroscope as well as network technologies such as Bluetooth, and Wi-Fi with which a variety of different data can be generated, received and recorded. Furthermore smart phones are computationally able to perform data analysis tasks on these received, or sensed data such as data mining. Many data mining technologies for smart phones are tailored for data streams due to the fact that sensed data is usually received and generated in real time and due to the fact that limited storage capacity on mobile devices requires that the data is analysed and mined on the fly while it is being generated or received. For example the Open Mobile Miner (OMM) tool [25] allows the implementation of data mining algorithms for data streams that can be run on smart phones.

Existing data mining systems for smart phones such as MobiMine [19] or VEDAS [20] and its commercial version *MineFleet* [21, 23] are some examples of systems that deploy data stream mining technology to mobile phones. However to our knowledge all existing data mining systems for mobile devices either

facilitate data mining on a single node or follow a centralised approach where data mining results are communicated back to a server which makes decisions based on the submitted results. Constraints that require the distribution of data mining tasks among several smart phones are, large and fast data streams, subscription fees to data streams, and data transmission costs in terms of battery and bandwidth consumption. The data transmission cost can be lowered by processing parts of the same data stream locally on different smart phone devices that only collaborate only by exchaning local statistics, or locally generated data mining models rather than raw data. The collaborative data mining on smart phones and 'Tablet PCs' facilitated by building an ad hoc network of mobile phones will allow to build significantly useful analysis tasks, however this area remains widely unexplored.

In this paper we describe and evaluate the Pocket Data Mining (PDM) framework, coined and proven to be computationally feasible in [3]. PDM has been built as a first attempt to explore collaborative and distributed data mining using stream mining technologies [6], mobile software agents technologies [7, 8] and embedded programming for mobile devices such as smart phones and 'Tablet PCs'. The main motivation in developing PDM is to facilitate the seamless collaboration between users of mobile phones which may have different data, sensors and data mining technology available. The usage of mobile agent technology is motivated by the agent's autonomous decentralised behaviour, which enables PDM to be applied on highly dynamic problems and environments with a changing number of mobile nodes. A second motivation for using mobile agent technology is the communication efficiency of mobile agent based distributed data mining [9, 10]. A general agent based collaborative scenario could look like the following. A mobile device that has a data mining task sends out a mobile agent that roams the network of mobile devices and collects for the data mining task useful information, such as which mobile devices have which data sources and/or sensors available and which data mining technologies are embedded on these devices. Next a further agent is sent out to consult the data mining task relevant mobile devices (in terms of their data sources, sensors and data mining technology) and uses the collective information to synthesize the data mining task.

Possible applications for PDM comprise:

- Stock market analysis tasks for investors and brokers, brokers can retrieve real time stock market data anytime anywhere they want using smart phones and can perform data mining on this data in order to support decisions to sell or buy shares [11]. However brokers may only want to subscribe to data of companies they are directly interested in, as the data transfer is expensive in terms of bandwidth consumption, processing power and battery life. Hence locally installed data mining technology may not pick up and learn direct/indirect dependencies between the subscribed shares and non subscribed shares. Collaborative data mining using PDM can overcome these limitations by sharing local models rather than data.

- Recent budget cuts of the British coalition government due to the current economic crisis could lead to a reduction of about 60000 police officers [26]. Reduction in police staff will have to be compensated. PDM could help streamlining the process of knowledge acquisition on the crime scene. The crime scene investigators could form an ad hoc network using their smart mobile phones. They could capture pictures, video data and finger prints as well as any other sensory data on the crime scene or from online data sources. If the task is to know more information about an aspect of the crime, the distribution of tasks could be that one device is to do an Internet search, another is to take pictures and a further one may retrieve data from other sensors close to the crime scene for example CCTV cameras. Without PDM the information recorded would have to be formatted and entered into a central data server and a data analyst would have to be employed to evaluate the gathered information, whereas with PDM the information could be fused together in real-time to give insights and knowledge about the crime.
- Sensors of smart phones can collect in situ continuously data about the healthy condition of a patient, for example some earphones of mobile phones can read a person's blood pressure [27], the accelerometer could detect physical activity, also the body temperature could be recorded and the fact that the mobile phone is used indicates that the user is conscious and probably well. Behaviour patterns can be mined from this sensory data. Nurses and 'mobile medical staff' could be equipped with mobile devices as well, if the nurse is idle, then the mobile phone can send out a mobile agent that roams the network of 'patients' and makes a decision where the nurse is needed most and instructs the nurse to go there. This decision may take the health status, the location of the patient and the nurse into account as well as the nurse's particular area of expertise.

The growing demand for commercial knowledge discovery and data mining techniques has led to an increasing demand of classification techniques that generate rules in order to predict the classification of previously unseen data. Hence classification rule induction is also a strong candidate technology to be integrated into the PDM framework. For example the classification of data streams about the stock market could help brokers to make decisions whether they should buy or sell a certain share. Also in the area of health care, classification of streaming information may be beneficial, for example the smart phone may record various health indicators such as the blood pressure and/or the level of physical activity and derive rules that may indicate if a patient needs medical attention, the urgency and what kind of medical attention is needed. For this reason a version of PDM that incorporates two strong data stream classification technologies has been created and is evaluated in this paper.

The paper is organised as follows. Section 2 highlights some related work in the area of distributed data mining on mobile devices; Section 3 describes the PDM framework in the context of classification rule induction on data streams; Section 4 evaluates several configurations of PDM comprising different classi-

fiers; Section 5 highlights ongoing work on PDM and future developments; some concluding remarks can be found in Section 6.

2 Related Work

The topic of this research paper lies in the intersection of distributed data mining, mining data streams and mobile software agents.

Distributed data mining has been surveyed thoroughly by Park and Kargupta in [22]. Mainly, there are two broad categories of distributed data mining, namely, homogeneous and heterogeneous. It has to be noted that this categorisation is made in reference to the attributes. Homogeneous distributed data mining refers to the process of mining the same set of attributes over all the participating nodes. On the other hand, heterogeneous distributed data mining refers to mining different sets of attributes in each participating node. In this paper, our focus is on the vertically partitioned data, i.e., the heterogeneous distributed data mining scenario. This choice has been made to provide a more realistic scenario to the applications discussed in this paper. It is worth noting that in this paper, we use the terms homogeneous and heterogeneous to refer to the data mining algorithms used in the process, where homogeneous refers to the use of only one data mining algorithm across all the participating nodes, and heterogeneous refers to the use of different data mining algorithms.

Mining data streams, on the other hand, is a more recent topic of research. A concise review of the area by Gaber et al is given in [6]. A more detailed review is given in [28]. The area is concerned with analysis of data generated in a high speed relative to the state-of-the-art computational power, with a constraint of real-time demand of the results. Hundreds of algorithms have been proposed in the literature addressing the research challenges of data stream mining. Notable success of the use of Hoeffding bound to approximate the data mining models for streaming data has been recognised [13]. The two-stage process of online summarisation and offline mining of streaming data, proposed by Aggarwal el al [1, 2], has been also recognised as a feasible approach to tackle the high data rate problem. Addressing both the resource constraints and high speed aspects of data stream mining has been addressed by Gaber et al [31, 29, 30] by proposing the algorithm granularity approach. The approach is generic and could be plugged into any stream mining algorithm to provide resource-awareness and adaptivity. Mining data stream algorithms is at the heart of our *pocket data mining* framework.

Finally, mobile software agents are computer programs that autonomously and intelligently move from one node to the other to accomplish its task. Potential applications and obstacles of this technology have been detailed in [7, 8]. The use of mobile agent technology for distributed data mining has been recognised, as an alternative paradigm to the client/server technologies for large databases. A cost model has been developed by Krishnaswamy et al [12], suggesting a hybrid approach to distributed data mining, combining both client/server and mobile agent paradigms. Our choice of mobile agent paradigm in this re-

search project has been due to the fact that our approach follows a peer-to-peer computation mode, and also that centralisation of the stream data mining in the mobile computing environment is infeasible.

3 PDM: Pocket Data Mining

PDM describes the collaborative data mining of data streams using mobile devices and mobile agent technology which is executed on an ad hoc mobile network. Section 3.1 illustrates the basic framework and workflow of PDM whereas Section 3.2 highlights a particular implementation of PDM using two different classifiers.

3.1 Architecture

The basic architecture is depicted in Figure 1. The mobile device that has a data mining task and utilises PDM to solve it is called the task initiator. PDM consists of three generic software agents that may be mobile and thus able to move between mobile phones within the ad hoc mobile network [3].:

- (Mobile) **A**gent **M**iners (AM) are distributed over the ad hoc network. They may be static agents used and owned by the user of the mobile devices. Or they may be mobile agents remotely deployed by the user of a different mobile device. They implement the basic stream mining algorithms; however they could also implement batch learning algorithms if required by the application.
- **M**obile Agent **R**esource **D**iscoverers (MRD) are mobile agents that are used to roam the network in order to discover for the data mining task relevant data sources, sensors, AMs and mobile devices that fulfil the computational requirements. They can be used to derive a schedule for the Mobile Agent Decision Makers described below.
- **M**obile **A**gent **D**ecision **M**akers (MADM) can move the mobile devices that run AMs and consult the AMs in order to retrieve information or partial results for the data mining task. The MADMs can use the schedule derived by the MRDs.

Algorithm 1 describes the basic data mining workflow that PDM employs collaboratively. The task initiator forms an ad hoc network of participating mobile devices within reach. Next the task initiator starts the MRD agent which will roam the network searching for data sources that are relevant for the data mining task, and for mobile devices that fulfill the computational requirements (battery life, memory capacity, processing power, etc.). While the MRD is collecting this information it will decide on the best combination of techniques to perform the data mining task. On its return to the task initiator the MRD will decide which AMs can and need to be deployed to remote mobile devices. There might be restrictions, some mobile phone owners may not allow alien AMs to

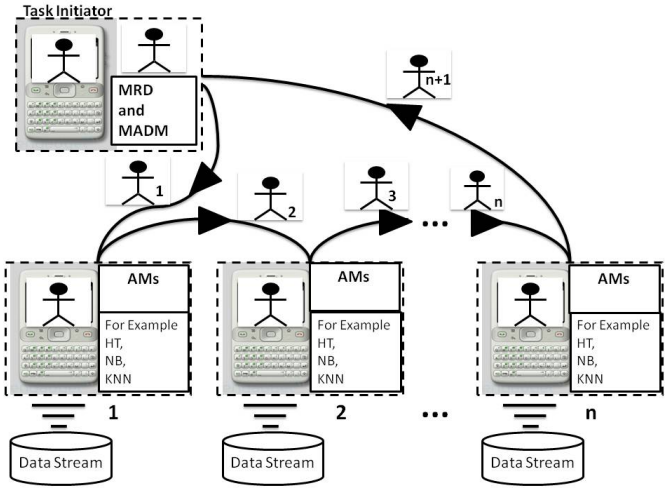

Fig. 1. The PDM Architecture.

be deployed, for example they may have limited computational capacity such as battery life, their data might be confidential etc. Concerning the confidentiality issues the owner of the mobile device may still allow its own AMs to be consulted as he will have control over its own AMs and thus about which information they release to alien MRD and MADM agents. The AMs are executed concurrently as indicated by the parallel for loop (parFor) in Algorithm 1. Finally the task initiator starts the MADM agent with the schedule provided by the MRD agent. The MADM agent visits the in the schedule listed AMs and will use their model in order to gain information for their decision making process. Finally on the return to the task initiator the MADM agent will make a collective decision based in the information gathered from the AMs distributed in the network.

Algorithm 1 PDM's collaborative data mining workflow

Task Initiator: Form an ad hoc network of mobile phones;
Task Initiator: start *MRD* agent;
MRD: Discover data sources, computational resources and techniques;
MRD: Decide on the best combination of techniques to perform the task;
MRD: Decide on the choice of stationary AMs and deploy mobile *AMs*;
Task Initiator: Start *MADM* agent with schedule provided by the *MRD*;
parFor $i = 1$ **to** $i =$ *number of AMs* **do**
  AM_i: starting mining streaming data until the model is used by the MADM agent.
end parFor

The current implementation of PDM offers AMs that implement classifiers for data streams, in particular the Hoeffding trees and the Naive Bayes classifiers which will be described in Section 3.2. It is assumed that the AMs are subscribed to the same data stream, however potentially to different parts of it. For example in the context of the stock market application outlined in Section 1, the mobile device may only have data about shares available in which the user of the mobile device is interested in, or more general the mobile device may only be subscribed to specific features of the data stream. A broker may train its own AM classifier on the data he has subscribed to, this could be for example by updating a model which is based on classes 'buy', 'sell', 'do not sell' and 'undecided' whenever he makes a new transaction. He may also use the current model to support his decisions to 'buy' or 'sell' a share. However, if the broker is now interested in buying a new share he has not much experience with, thus he may be interested in what decisions other brokers are likely to make in the same situation. Other brokers may not want to disclose their actual transactions but may share their local AM or even allow alien AMs to be deployed and for this the brokers can use PDM. With the current version of PDM the data mining workflow outlined in Algorithm 1 may look like the following, where the mobile device of the broker interested in investing in a new share is the task initiator. In the steps below and elsewhere in the paper, if we refer to a PDM agent hopping, we mean that the agent stops is execution, is transferred by PDM to a different mobile device and resumes its execution on this device. Also in the steps below it is assumed that the ad hoc network is already established:

1. Task Initiator: Send a MRD agent in order to discover mobile devices of brokers that have subscribed to relevant stock market data, i.e. data about the shares the broker is interested in.
2. The MRD agent hops from mobile device to mobile device and if it finds a device subscribed to relevant data it memorises the device and also if there are any useful AMs already available. If there are no useful agents it will memorise if the device allows alien agents to be deployed.
3. The MRD agent returns to the task initiator. From there the MRD agent will remotely deploy relevant AMs to mobile devices in its list that have relevant data but no relevant AMs however, allow alien AMs to be deployed remotely.
4. Once all AMs are deployed the MRD agent composes a schedule of all relevant classifier AMs subscribed to relevant data and passes it on to the MADM agent.
5. The MADM agent loads the data about the new shares the broker is interested in and starts hopping to each AM in the schedule.
6. On each AM the MADM agent hands over the 'shares data' to the AM and asks to classify it for example with class 'buy', 'do not buy', 'sell', 'do not sell' or 'undecided. The MADM may also retrieve some estimate how reliable the AMs thinks its classification is, for example its local classification accuracy.
7. Once the MADM returns to the task initiator it may employ a majority voting on the collected classifications from each AM or a weighted majority voting incorporating the AMs local accuracy (we will call this the AMs

Algorithm 2 Hoeffding tree induction algorithm.

1: Let HT be a tree with a single leaf (the root)
2: **for all** training examples **do**
3: Sort example into leaf l using HT
4: Update sufficient statistics in l
5: Increment n_l, the number of examples seen at l
6: **if** $n_l \bmod n_{min} = 0$ **and** examples seen at l not all of same class **then**
7: Compute $\overline{G}_l(X_i)$ for each attribute
8: Let X_a be attribute with highest $\overline{G}_l$
9: Let X_b be attribute with second-highest $\overline{G}_l$
10: Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
11: **if** $X_a \neq X_\emptyset$ **and** $(\overline{G}_l(X_a) - \overline{G}_l(X_b) > \epsilon$ **or** $\epsilon < \tau)$ **then**
12: Replace l with an internal node that splits on X_a
13: **for all** branches of the split **do**
14: Add a new leaf with initialized sufficient statistics
15: **end for**
16: **end if**
17: **end if**
18: **end for**

weight). The outcome of the (weighted) majority voting is used as recommendation for the broker to the investment in the new share.

3.2 Implementation of PDM Using Distributed Hoeffding Trees and Distributed Naive Bayes for Mining Data Streams on Mobile Devices

PDM in its current version offers two AMs for classification tasks on data streams. One of the AMs implements the Hoeffding Tree classifier [13] and one that implements the Naive Bayes classifier. The AM that employs the Hoeffding Tree classifier uses the Hoeffding Tree implementation from the **M**assive **O**nline **A**nalysis (MOA) tool [4] as outlined by Bifet and Kirkby [5] and shown in Algorithm 2. Hoeffding tree classifiers have been designed for high speed data streams.

The Naive Bayes classifier has been originally developed for batch learning; however its incremental nature makes it also applicable on data streams. Again the AM employing the Naive Bayes classifier uses the Naive Bayes implementation from the MOA tool [4]. Naive Bayes is based on the Bayes Theorem [14] which states that if C is an event of interest and $P(C)$ is the probability that event C occurs, and $P(C|X)$ is the conditional probability that event C occurs under the premise that X occurs then:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

The Naive Bayes algorithm uses the Bayes Theorem to assign to a data instance to the class it belongs to with the highest probability.

4 Evaluation of PDM

This paper examines the PDM's applicability of classification rule induction to data streams. Three different configurations of PDM have been thoroughly tested. One PDM configuration solely based on Hoeffding Tree AMs, the second configuration of PDM is solely based on Naive Bayes AMs and the third configuration is a mixtures of both Hoeffding Tree and Naive Bayes AMs. Section 4.1 outlines the general experimental setup, Section 4.2 outlines the experimental results obtained using only Hoeffding Tree AMs, Section 4.3 outlines the experimental results obtained using only Naive Bayes AMs and Section 4.4 outlines the experimental results obtained using a mix of Hoeffding tree and Naive Bayes AMs.

4.1 Experimental Setup

The two classifier AMs use the Hoeffding Tree and Naive Bayes implementations from the MOA toolkit [4] with the reasoning that MOA is based on the WEKA [15] data mining workbench and thus supports the usage of the well known .arff data format. PDM is also built on the well known **J**ava **A**gent **D**evelopment **E**nvironment (JADE) [16]. JADE agents are hosted and executed in *JADE containers* that can be run on the mobile devices and PCs. JADE agents can move between different JADE containers and thus between different mobile devices and PCs. As JADE agents can be developed on PCs and run on both PCs and mobile phones it is possible to develop and evaluate PDM on a LAN of PCs. The used LAN consists of 9 workstations with different software and hardware specifications and is connected with a CISCO Systems switch of the catalyst 2950 series. In the configurations of PDM examined in this paper 8 machines were either running one Hoeffding Tree or one Naive Bayes AM. The 9th machine was used as the task initiator, however any of the 8 machines with AMs could have been used as task initiator as well. The task initiator starts the MADM in order to collect classification results from the AMs.

The data streams for PDM have been simulated using the datasets described in Table 1. The datasets have been labelled with test 1 to 6 for simplicity when referring experiments to a particular data stream. The data for test 1, 2, 3 and 4 have been retrieved from the UCI data repository [17] and datasets 5 and 6 have been taken from the Infobiotics benchmark data repository [18]. All datasets are stored in the .arff format and the data stream is simulated by taking a random data instance from the .arff file and feeding it to the AM. Instances may be selected more than once for training purposes.

Table 1. Evaluation Datasets

Test Number	Dataset	Number of Attributes	Number of Instances
1	kn-vs-kr	36	1988
2	spambase	57	1999
3	waveform-500	40	1998
4	mushroom	22	1978
5	infobiotics 1	20	$\approx$ 200000
6	infobiotics 2	30	$\approx$ 200000

As mentioned in Section 3 each AM may be subscribed to only a subset of the total feature space of a data stream, we call this a vertically partitioned data stream. For example a stock marked broker may only subscribe to data about companies he is interested in investing, or a police officer may only access data he has clearance for. Even if a user of a mobile device may have access to the full data the owner of the device may not want or be able to subscribe to 'for him' unnecessary features for computational reasons, such as bandwidth consumption, also the more data is processed by AMs the more power they will consume, or simply the processing time of the data stream is longer the more features are streamed in and need to be processed and higher subscription fees may be imposed. Yet the current subscription may be insufficient for classifying new data instances. However the task initiator can send a MADM with the unclassified data instances. This MADM visits and consults all relevant AMs that belong to different owners that may have subscribed to different features that are possibly be more relevant for the classification task.

The MADM collects predictions from each AM for each unclassified data instance and the estimated 'weight' (accuracy) of the AM, which it uses to decide on the final classification. In the *PDM* framework each AM treats a streamed labelled instance either as train or as test instance with a certain probability which is set by the owner of the AM. The default probability used in the current setup is 20% for the selection as a test and 80% for the selection as a training instance. Each training instance is put back into the stream and may be selected again as training instance, this allows to simulate endless data streams with reoccurring patterns. The test instances are used to calculate the 'weight' of the AM. The AM also takes concept drifts into account when it calculates its 'weight' by defining a maximum number of test instances to be used. For example if the number of test instances it 20 and there are already 20 test instances selected then the AM replaces the oldest test instance by the newly incoming test instance and recalculates the 'weight' using the 20 test instances.

After the MADM finished consulting all AMs in its schedule it returns to the task initiator and uses the local predictions from each AM and the AMs weights in order to derive a final classification using a 'weighted majority voting'. For example for the classification of one data instance, if there are three AMs, *AM1*, *AM2* and *AM3*. *AM1* predicts class A and has a weight of 0.57, *AM2* also predicts class *A* and has a weight of 0.2 and *AM3* predicts class *B* and has a

weight of 0.85. The MADM's 'weighted' prediction for class A is $0.57A + 0.2A = 0.77A$ and for class B $0.85B = 0.85B$. Thus the MADM yielded the highest weighted vote for classification B and will label the concerning instance with class B.

The user of PDM can specify which features its AM shall subscribe to, however in reality we may not know the particular subscription, thus in the experimental set-up each AM subscribes to a random subset of the feature space. In particular experiments with each AM holding 20%, 30% and 40% of the total feature space have been conducted.

The terminology that is used in Sections 4.2, 4.3 and 4.4 is explained below:

- The **weight** refers to the local accuracy of the AM calculated using randomly drawn test instances from the local data stream.
- **MADM's accuracy** or **PDM's accuracy** is the accuracy achieved by the MADM using the test dataset classified by 'weighted majority voting' by the MADM.
- **local accuracy** is not to be confused with the weight. The local accuracy is the actual accuracy that a particular AM achieved on classifying the MADM's test data. This accuracy is only calculated for evaluation purposes, it would not be calculated in the real application as the real classifications of the MADM's test set would be unknown.
- the **average local accuracy** is calculated by averaging the local accuracies of all AMs. The average accuracy is used to show if the 'weighted majority voting' performs better than simply taking a majority vote.

4.2 Case Study of PDM using Hoeffding Trees

The datasets listed in Table 1 are batch files. Using batch files allows us to induce classifiers using batch learning algorithms and thus to compare PDM's classification accuracy to the ideal case of executing batch learning algorithms on the whole datasets using all attributes. In particular the C4.5 [24] and Naive Bayes batch learning algorithms have been used from the WEKA workbench [15]. The choice of C4.5 is based on its wide acceptance and use; and to the fact that the Hoeffding tree algorithm is based on C4.5. The choice of Naive Bayes is based on the fact that it is naturally incremental, computationally efficient and also widely accepted.

In general it is expected that the more features the AMs have available the more likely it is that they achieve a high classification accuracy and thus the more likely it is that the MADM achieves an high classification accuracy as well. Yet some features may be highly predictive and others may not be predictive and even introduce noise. Thus in some cases having more features available may decrease the AMs and thus PDM's accuracy. 70% of the data instances from each dataset in Table 1 have been used to simulate the local data stream and the remaining 30% have been used as test data in order to evaluate PDM's and respectively MADM's accuracy. All experiments outlined in this paper have been

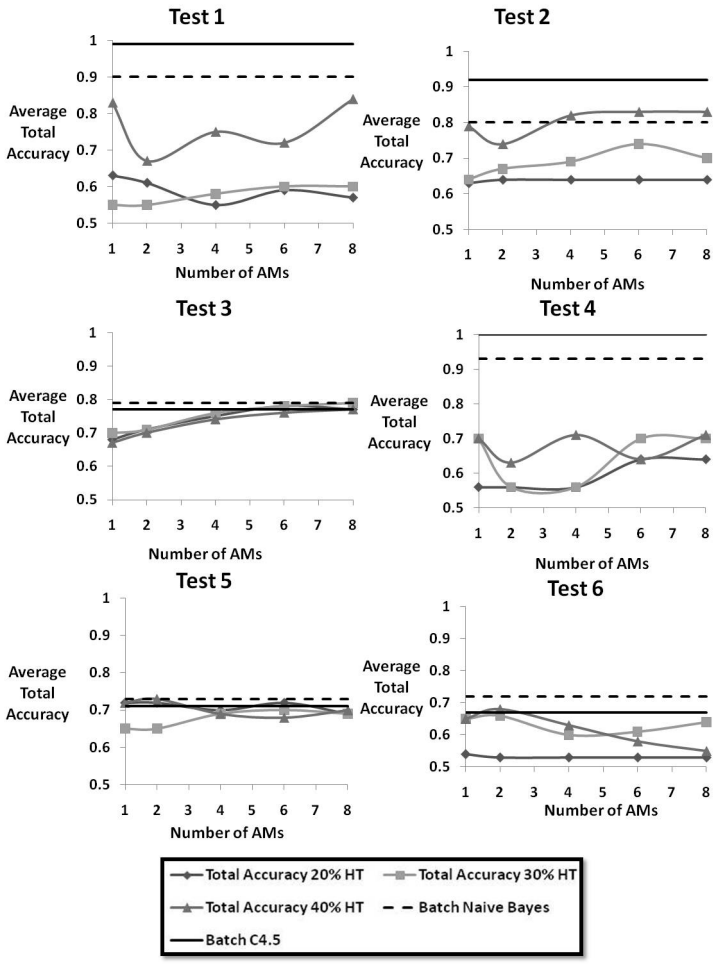

Fig. 2. PDM's average classification Accuracy based on Hoeffding Trees.

conducted 5 times, the average local accuracy of each AM has been calculated and recorded as well as PDM's or respectively MADM's average accuracy.

Figure 2 shows PDM's average classification accuracy plotted versus the number of AMs visited. The experiments have been conducted for configurations where all AMs either subscribe to 20%, 30% or 40% of the features of the data stream. The features each AM subscribes to are selected randomly thus some AMs may have subsets of their features in common and some not. That two or more AMs have features in common is realistic, for example for the stock market broker application briefly outlined in Section 1. Two brokers may be interested in the 'Compaq' share but only one of them may be interested in the 'Hewlett-Packard' share and one in the 'Microsoft' share.

The largest difference between Naive Bayes's accuracy and C4.5 is for test 2 where Naive Bayes's accuracy is 80% and C4.5 91%, otherwise both batch learning algorithms achieve similar accuracies. Concerning PDM's accuracy based on Hoeffding trees it can be seen that PDM generally achieves accuracies above 50% for all datasets. In general PDM configurations with AMs using just 20% of the feature space generally perform much worse than configurations with 30% or 40% which can be explained by the fact that predictive features are more likely not to be selected. In some cases, for example for test 2 it seems that configurations of PDM with 30% achieve a higher accuracy than configurations with 40% which can be due to the fact that with subscribing to 40% of the features it is also more likely that non predictive features that also introduce noise are selected compared with subscribing to 30%. In general it can be observed that if subscribing to 30% instances achieves better results than subscribing to 40% instances the difference in accuracy between both configurations is not very large. In general PDM achieves accuracies close to the batch learning algorithms C4.5 and Naive Bayes, notably in tests 3 and 5 but also for the remaining tests PDM achieves close accuracies to those of Naive Bayes and C4.5. In general PDM based on Hoeffding trees achieves acceptable classification accuracy in most cases.

Varying the number of AMs generally is dependent on the dataset used. Highly correlated attributes in one dataset would only need small number of AMs and vice versa.

Figure 3 compares PDM's accuracy (achieved by the MADM through 'weighted majority voting') with the average local accuracy of all AMs versus the number of AMs visited. Each row of graphs corresponds to one of the tests in Table 1 and each column of graphs corresponds to a percentage of features the AMs are subscribed to. The lighter line in the graphs is the accuracy of PDM and the darker line is the average local accuracy of all AMs. PDM's accuracy is in most cases higher or even better than the average local accuracy, hence the MADM's 'weighted majority voting' achieves a better result compared with simply taking the average of the predictions from all AMs.

4.3 Case Study of PDM using Naive Bayes

A further configuration of PDM solely based on Naive Bayes AMs has been evaluated the same way as PDM solely based on Hoeffding trees has been. PDM solely based on Naive Bayes is expected to produce similar results compared with PDM solely based on Hoeffding trees evaluated in Section 4.2.

Figure 4 presents the data obtained of PDM solely based on Naive Bayes the same way as Figure 2 does for PDM solely based on Hoeffding trees. Again the experiments have been conducted for configurations where all AMs either subscribe to 20%, 30% or 40% of the features of the data stream. The features each AM subscribes to are selected randomly thus some AMs may have subsets of their features in common and some not. Concerning PDM's accuracy based on Hoeffding trees it can be seen that PDM generally achieves accuracies above 50% for all datasets. Similar compared with Figure 2 PDM configurations with AMs using just 20% of the feature space generally perform much worse than

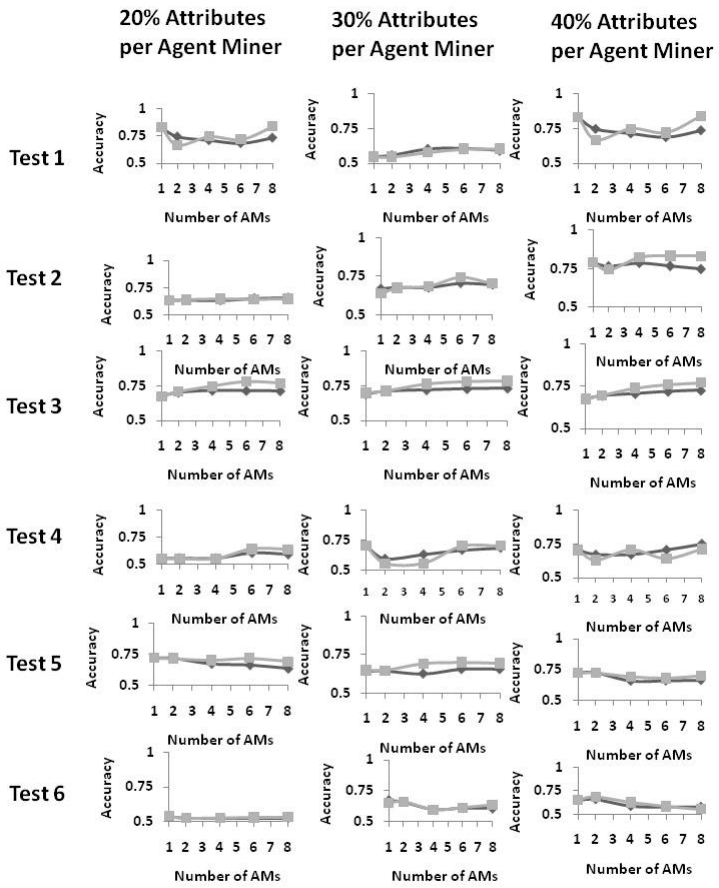

Fig. 3. PDM's average classification accuraciy versus the average local accuracy of the AMs with Hoeffding Trees

configurations with 30% or 40% which can be explained by the fact that predictive features are more likely not to be selected. Yet in some cases, for example for test 2 it seems that configurations of PDM with 30% achieve a higher accuracy than configurations with 40% which can be due to the fact that with subscribing to 40% of the features it is also more likely that non predictive features that introduce noise are selected compared with subscribing to 30%. In general PDM achieves accuracies close to the batch learning algorithms C4.5 and Naive Bayes, notably in tests 3, 4, 5 and 6. However also for the remaining tests PDM achieves close accuracies to those of Naive Bayes and C4.5. In general PDM based on Hoeffding trees achieves acceptable classification accuracy in most cases.

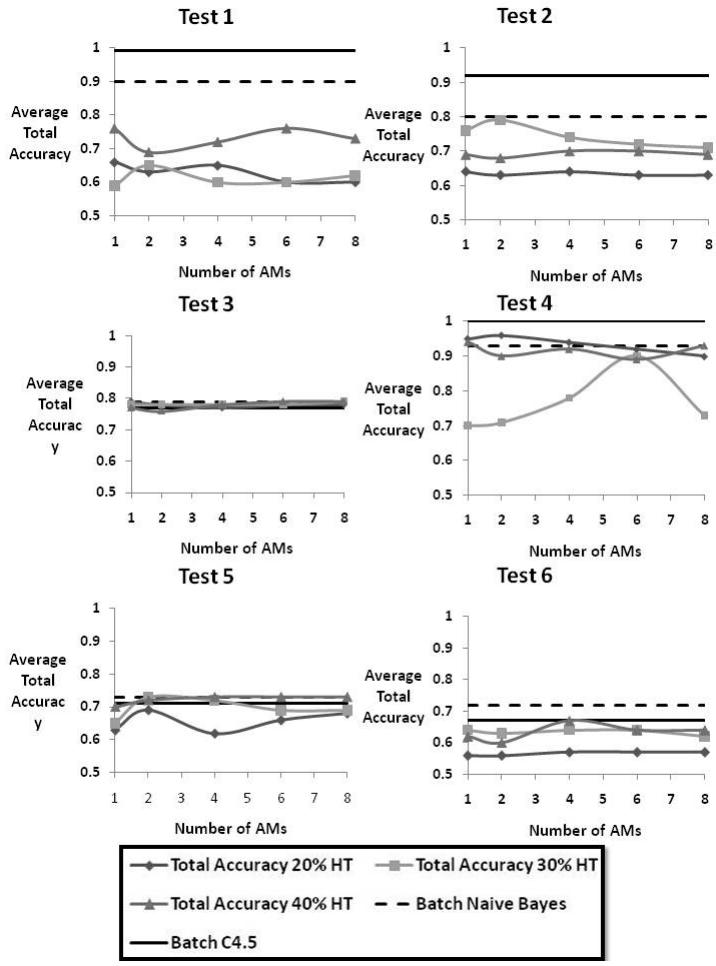

Fig. 4. PDM's average classification Accuracy based on Naive Bayes.

Similar to the previous set of experiments, varying the number of AMs generally is dependent on the dataset used. Highly correlated attributes in one dataset would only need small number of AMs and vice versa.

Figure 5 analogous to Figure 3 opposes PDM's accuracy (achieved by the MADM through 'weighted majority voting') and the average local accuracy of all AMs versus the number of AMs visited. Each row of graphs corresponds to one of the tests in Table 1 and each column of graphs corresponds to a percentage of features the AMs are subscribed to. The lighter line in the graphs is the accuracy of PDM and the darker line is the average local accuracy of all AMs. Similar to the Hoeffding tree results PDM's accuracy is in most cases higher or even better than the average local accuracy, hence the MADM's 'weighted majority voting'

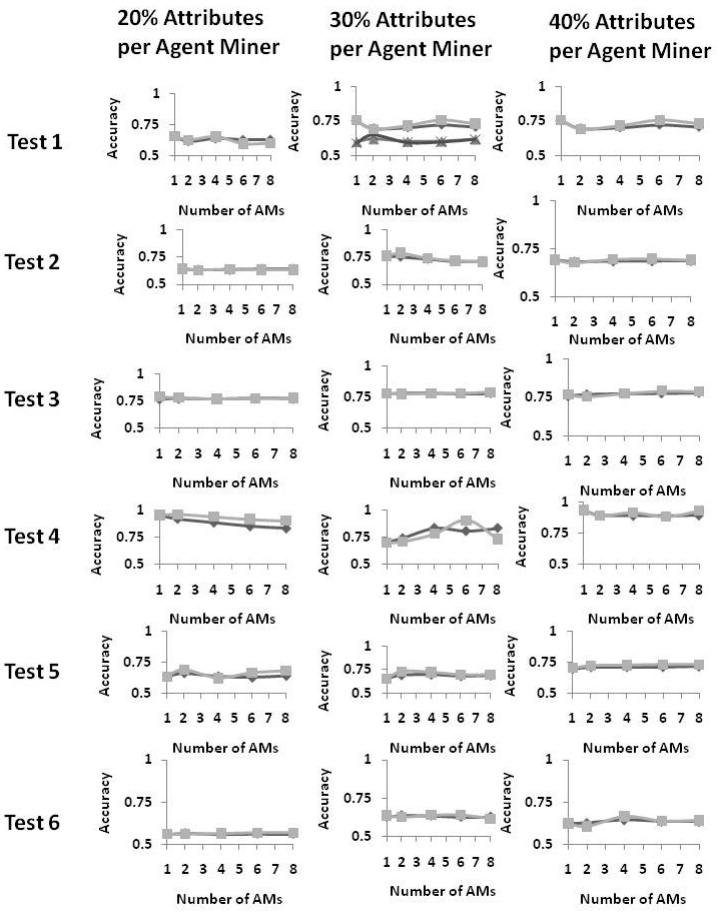

Fig. 5. PDM's average classification accuracy versus the average local accuracy of the AMs with Naive Bayes.

either achieves a better result than simply taking the average of the predictions from all AMs.

4.4 Case Study of PDM using a Mix of Hoeffding Trees and Naive Bayes

Figure 6 highlights the accuracies of the two PDM configurations solely based on Hoeffding and solely based in Naive Bayes for different numbers of visited AMs. The bars in the figure are in the following order from left to right: Total accuracy of PDM with Hoeffding Trees with 20% attributes; total accuracy of *PDM* with Hoeffding Trees with 30% attributes; total accuracy of PDM with Hoeffd-

ing Trees with 40% attributes; total accuracy of PDM with Naive Bayes with 20% attributes; total accuracy of PDM with Naive Bayes with 30% attributes; total accuracy of PDM with Naive Bayes with 40% attributes; accuracy for batch learning of Naive Bayes with all attributes; and finally accuracy for batch learning of C4.5 with all attributes.

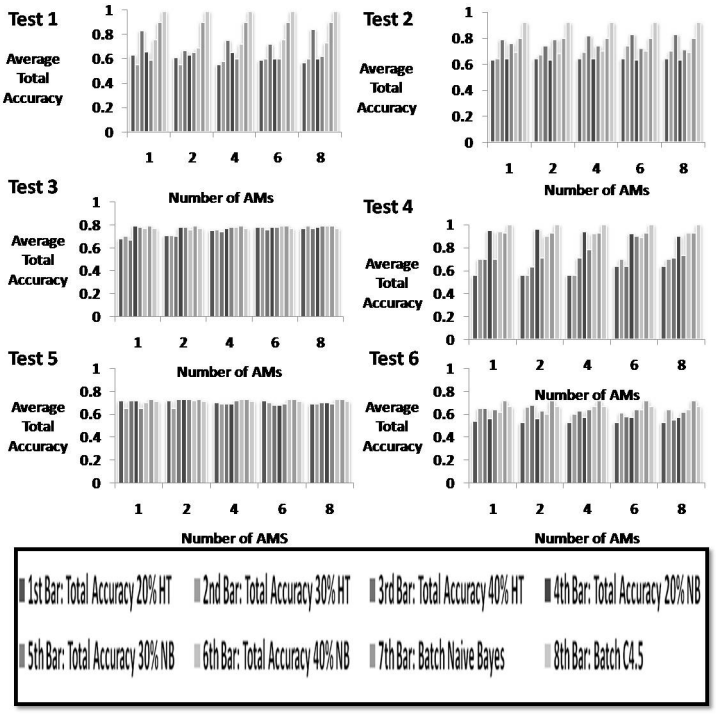

Fig. 6. PDM's average classification accuracy for both configurations with Hoeffding Trees and Naive Bayes.

On tests 3 and 5 both configurations of PDM achieve an almost equal classification accuracy. Also for tests 1, 2, 4 and 6, the classification accuracies of PDM are very close for both configurations and there does not seem to be a bias towards one of the classifiers used. Hence a heterogeneous configurations of PDM with a mixture of both classifiers would be expected to achieve a similar performance than a homogeneous configuration solely based on Hoeffding Trees or Naive Bayes. Such a heterogeneous set-up of AM classifiers would also be a more realistic set-up as owners of mobile devices may use their individual classification techniques tailored for the data they subscribed to.

In order to show that a heterogeneous set-up of AM classifiers achieves a similar accuracy to a homogeneous solely based on Hoeffding Trees or Naive Bayes we have evaluated configurations of PDM that use both classifiers. Again

all experiments have been conducted five times and the average of the achieved accuracy by the MADM has been calculated.

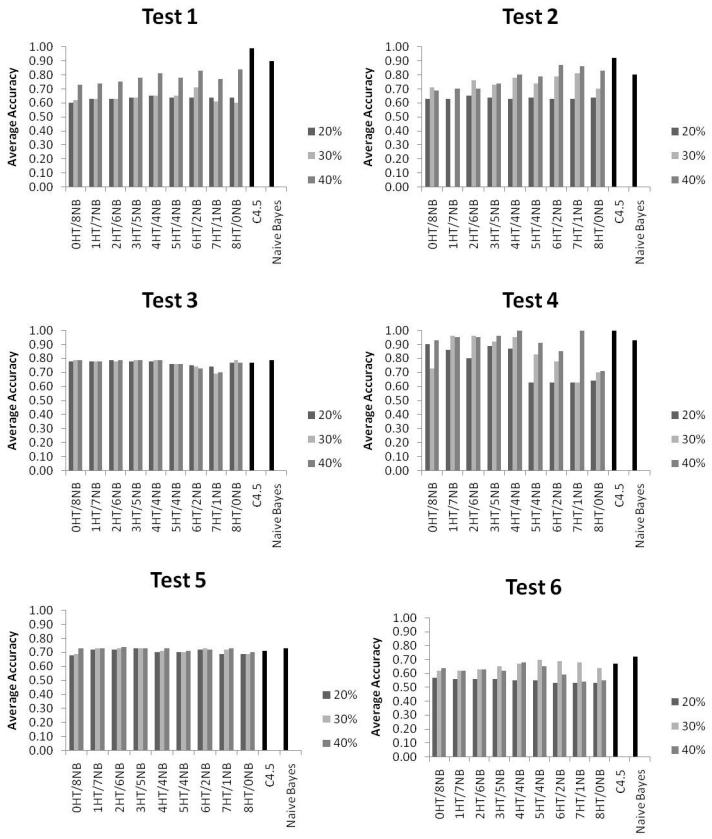

Fig. 7. Average classification accuracy for a heterogeneous set-up of classifier AMs in PDM.

Figure 7 highlights experiments conducted with different heterogeneous setups of PDM for all 6 datasets listed in Table 1. The average accuracy of the MADM is plotted against the combination of algorithms embedded in the deployed AMs. The graph is split showing AMs working with 20%, 30% and 40% of the features. All possible combinations of Hoeffding Trees and Naive Bayes AMs have been evaluated. The horizontal labels in Figure 7 are read the following way. HT stands for Hoeffding Tree and NB for Naive Bayes, the number before HT and NB is the number of Naive Bayes or HT classifiers visited respectively. For example label '3HT/5NB' means that 3 Hoeffding Tree AMs and 5 NB agents have been visited by the MADM. Also plotted in Figure 7 is the

result the batch learning algorithms C4.5 and Naive Bayes achieve using all the features. The achieved accuracies are close compared with those achieved by the batch learning algorithms which have the advantage over *PDM* of having all the features available which would again not be the case in a realistic scenario where subscribers of a data stream limit their subscription only to properties they are particularly interested in for reasons stated in Sections 1 and 4.

In Figure 7 for test 1 it can be seen that using 20% features or 30% seems to achieve very similar classification accuracies. However for using 40% features the classification accuracy improves considerably and gets close to the batch learning accuracies which use all features, also for using 40% of the features it can be seen that configurations with more Hoeffding tree AMs perform slightly better than configurations with more Naive Bayes AMs. For test 2 it can be seen that using 30% instances already improves the classification accuracy and there is a tendency for the usage of 30% and 40% instances that configurations with more Hoeffding tree AMs perform slightly better, in particular configurations 6HT/2NB and 7HT/1NB seem to achieve high accuracies between 80% and 90%. Regarding test 3 all percentages of features used achieve a very similar and very good classification accuracies that can well compete with the accuracies achieved by the batch learning algorithms on all features. Also configuration wise it seems that using configurations with more Naive Bayes then Hoeffding trees seem to perform better, however also a configuration solely based on Hoeffding trees achieves very good classification accuracies close to batch learning algorithms. In test 4 the tendency seems that using more Naive Bayes classifiers achieve a higher accuracy than using more Hoeffding tree classifiers. In test 4 Naive Bayes seems to work better on configurations with less features subscribed to compared with Hoeffding trees. There is no noticeable tendency for test 5, all configurations and percentages of features seem to achieve a high accuracy very close to the one observed for the batch learning algorithms. On test 6 any configuration and even batch learning algorithms do not achieve a good classification accuracy. This suggests that the dataset for test 6 is not very well suited for classification in its current form. Also there is no particular tendency detectable for test 6.

Figure 7 also displays the data from homogeneous configurations of PDM solely based on Hoeffding trees, which is labelled as configuration 8HT/0NB and solely based on Naive Bayes, which is labelled 0HT/8NB. The results clearly show that heterogeneous configurations of PDM achieve very similar accuracies to homogeneous configurations of PDM. Also earlier in this section we stated that a heterogeneous set-up of AM classifiers would also be a more realistic setup as owners of mobile devices may use their individual classification techniques. Furthermore PDM may well benefit from using individual AMs from different owners as they are likely to be optimised on the local subscription of the data stream.

5 Ongoing and Future Work

5.1 Rating System for AMs

The current implementation of the MADM agent assumes that the local AMs are of good quality and thus in the case of classification of unlabelled data instances it is assumed that the weights are calculated correctly and truly reflect the AMs classification accuracy. This assumption may be true for the AMs we developed in-house, which we used for the evaluation in Section 4, but third party implementations may not be trusted. For this reason a rating system about AMs is currently being developed based on historical consultations of AMs by the MADM. For example if the MADM remembers the classifications and weights obtained from AMs visited, and the true classification of the previously unknown instances is revealed, then the MADM could implement its own rating system and rate how reliable a AM's weight was in the past. If an AM is rated as unreliable, then the MADM may even further lower its weight. However it is essential that this rating system is also able to loosen given ratings, as the AM's performance might well change if there is a concept drift in the data stream. In order to detect such concept drifts it is necessary that AMs that have a bad rating are still taken into consideration, even if it is with a low impact due to bad ratings.

5.2 Intelligent Schedule for MADMs

In its current implementation the MADM visits all available AMs, however this may be impracticable if the number of AMs is very large. Currently a mechanism is being developed for MADMs according to which the MADM can decide when to stop consulting further AMs. A possible stopping criteria could be that a certain time has elapsed or the classification result is reliable enough. Also the rating system outlined above can be used to determine an order in which AMs are visited. If there are time constraints the MADM may prioritise more reliable AMs.

6 Conclusions

The paper presents the Pocket Data Mining (PDM) framework for mining data streams in a collaborative fashion in a mobile environment consisting of smart phones. PDM is based on mobile agent technology using three types of agents. Agent Miners that embed stream mining technologies, Mobile Agent Resource Discoverers that are used to roam the network and search for relevant data streams and the Mobile Agent Decision Makers that visit and consults Agent Miner on whose results they base their final decision on. This paper presents a implementation of PDM for distributed classification of data streams and examines its feasibility. Two different configurations of PDM based on either Hoeffding tree Agent Miners or Naive Bayes Agent Miners have been examined. The experiments have also been conducted with different percentages of features available

to the Agent Miners. The classifiers used were the Hoeffding tree and the Naive Bayes classifiers. In general it has been observed that any configuration of PDM based on any of the two or both classifiers achieved an acceptable classification accuracy. Also it has been observed that the more features the Agent Miners have available the closer the accuracies are to the ideal case where batch learning algorithms C4.5 and Naive Bayes have the advantage of having 100% of the features available to train the classifier. Furthermore it has been observed that PDM's 'weighted' majority voting achieves higher classification accuracies than simply taking the Agent Miners local average accuracies. In general it does not seem that one of the classifiers is superior in general, this indicates that heterogeneous configurations of PDM using different classifiers in the same network will perform equally well. Also this would be the more realistic scenario as owners of smart phones are likely to employ the classifiers that are likely to perform well on their data subscription. To examine this further a heterogeneous setup of PDM using both Naive Bayes and Hoeffding tree classifiers have been evaluated. Again the classification accuracies were very similar to the ones achieved with homogeneous setups of PDM.

Ongoing work comprises a rating system to rate the quality of third party AMs; the development of an optimised schedule for MADM agents in order to derive data mining results faster if there are many available AMs.

PDM is a new niche of distributed data mining however hardly explored. The current implementation of PDM focuses on classification techniques, however, there exist many more data mining technologies tailored for data streams and mobile devices. For example there are stream mining techniques that classify unlabelled data streams [30, 32] which could be introduced into PDM. Also resource aware data mining algorithms as proposed in [29] will boost PDM's applicability in resource constraint mobile networks once integrated in PDM. But not only data streams can be used also the mobile phones sensors such as the gyroscope, camera, etc. could be used as data sources as well. In general PDM's applicability will benefit with the recent advances in smart phone technology and data stream mining technology and vice versa.

References

1. C. C. Aggarwal, J. Han, J. Wang, P. Yu. A Framework for Clustering Evolving Data Streams. Proceedings of the VLDB Conference, 2003.
2. C. C. Aggarwal, J. Han, J. Wang, P. Yu. On Demand Classification of Data Streams. Proceedings of the ACM KDD Conference, 2004.
3. Stahl F., Gaber M. M., Bramer M., and Yu P. S., Pocket Data Mining: Towards Collaborative Data Mining in Mobile Computing Environments, Proceedings of the IEEE 22nd International Conference on Tools with Artificial Intelligence (ICTAI 2010), Arras, France, 27-29 October, 2010.
4. Bifet A., Holmes G., Pfahringer B., Kranen P., Kremer H., Jansen T., Seidl T., Journal of Machine Learning Research (JMLR) (2010).
5. Bifet A. and Kirkby R., Data Stream Mining: A Practical Approach, Center for Open Source Innovation, August 2009.

6. Gaber M, M., Zaslavsky A., and Krishnaswamy S., Mining Data Streams: A Review, ACM SIGMOD Record, Vol. 34, No. 1, pp. 18-26, 2005, ISSN: 0163-5808.
7. A. Zaslavsky, Mobile Agents: Can They Assist with Context Awareness? IEEE MDM, Jan. 2004 ,Berkeley, California.
8. J. Page, A. Padovitz, M. Gaber, Mobility in Agents, a Stumbling or a Building Block? Proceedings of Second International Conference on Intelligent Computing and Information Systems, Cairo, Egypt, 5-7 March 2005.
9. J. da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klusch, Distributed Data Mining and Agents, Engineering Applications of Artificial Intelligence Journal, 2005 volume 18, pp. 791–807.
10. H. Kargupta, I. Hamzaoglu and B. Stafford, Scalable, Distributed Data Mining Using an Agent-Based Architecture. Proceedings of Knowledge Discovery and Data Mining. Eds: D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy, pp. 211214, 1997, AAAI Press.
11. S. Pittie, H. Kargupta, and B. Park. (2003). Dependency Detection in MobiMine: A Systems Perspective. Information Sciences Journal. Volume 155, Issues 3-4, pp. 227-243, Elsevier.
12. S. Krishnaswamy, S. W. Loke, A. B. Zaslavsky: A hybrid model for improving response time in distributed data mining. IEEE Transactions on Systems, Man, and Cybernetics, Part B 34(6): 2466-2479 (2004)
13. Domingos P. and Hulten G., Mining high-speed data streams, In International Conference on Knowledge Discovery and Data Mining, pages 71-80, 2000.
14. Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of bayesian classifiers. In National Conference on Artificial Intelligence, pages 223–228, 1992.
15. Witten I. and Frank E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, Second Edition, 2005.
16. Bellifemine F., Poggi A., and Rimassa G., Developing multi-agent systems with JADE. 7th International Workshop, ATAL 2000, Boston, MA, USA, July 7-9, 2000, Proceedings, LNCS 1986, pages 89-103. Springer Verlag, 2000.
17. Blake C. L. and Merz C. J., *UCI Repository of Machine Learning Databases* (Technical Report). University of California, Irvine, Department of Information and Computer Sciences, 1998.
18. Bacardit J. and Krasnogor N., *The Infobiotics PSP benchmarks repository.*, http://www.infobiotic.net/PSPbenchmarks, 2008.
19. Kargupta H., Park B., Pittie S., Liu L., Kushraj D., and Sarkar K. (2002). MobiMine: Monitoring the Stock Market from a PDA. ACM SIGKDD Explorations. January 2002. Volume 3, Issue 2. Pp. 37–46. ACM Press.
20. Kargupta H., Bhargava R., Liu K., Powers M., Blair P., Bushra S., Dull J., Sarkar K., Klein M., Vasa M., and Handy D. (2004). VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. Proceedings of the SIAM International Data Mining Conference, Orlando.
21. Kargupta H., Puttagunta V., Klein M., Sarkar K., On-board Vehicle Data Stream Monitoring using MineFleet and Fast Resource Constrained Monitoring of Correlation Matrices. Next Generation Computing. Invited submission for special issue on learning from data streams, volume 25, no. 1, pp. 5–32, 2007.
22. B. Park and H. Kargupta, Distributed Data Mining: Algorithms, Systems, and Applications, Data Mining Handbook, Editor: Nong Ye, 2002.
23. Agnik, MineFleet Description, http://www.agnik.com/minefleet.html
24. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

25. Krishnaswamy S., Gaber M. M., Harbach M., Hugues C., Sinha A., Gillick B., Haghighi P. D., and Zaslavsky A., Open Mobile Miner: A Toolkit for Mobile Data Stream Mining, Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2009, June 28 - 1 July, Paris, France. (Demo paper).
26. BBC, Budget Cuts of Police Force, http://www.bbc.co.uk/news/uk-10639938
27. Poh M., Kim K., Goessling A.D., Swenson N.C.,Picard R.W., Heartphones: Sensor Earphones and Mobile Application for Non-obtrusive Health Monitoring, IEEE InternationalSymposiumonWearableComputers, Austria, pp. 153–154, 2009.
28. M. M. Gaber, A. B. Zaslavsky, S. Krishnaswamy, Data Stream Mining, in Data Mining and Knowledge Discovery Handbook 2010, pp. 759-787, Springer Verlag.
29. Gaber, M, M., Krishnaswamy, S., and Zaslavsky, A., Resource-Aware Mining of Data Streams, Journal of Universal Computer Science, Vol. 11, No. 8 (2005), pp. 1440-1453, ISSN 0948-695x, Special Issue on Knowledge Discovery in Data Streams, Verlag der Technischen Universitt Graz, Know-Center Graz, Austria, August 2005.
30. M. M. Gaber and P. S. Yu. A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering. In Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), April 23-27, Dijon, France, pages 649656. ACM Press, 2006.
31. M. M. Gaber, Data Stream Mining Using Granularity-Based Approach, Foundations of Computational Intelligence (6) 2009, pp. 47-66, Springer
32. N. D. Phung, M. M. Gaber, and U. Rohm. Resource-aware online data mining in wireless sensor networks. In Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007), April 1-5, 2007.