Distributed Hoeffding Trees for Pocket Data Mining

Frederic Stahl, Mohamed Medhat Gaber, Max Bramer
University of Portsmouth
{Frederic.Stahl, Mohamed.Gaber, Max.Bramer}@port.ac.uk,
Philip S Yu
University of Illinois at Chicago
psyu@cs.uic.edu

ABSTRACT

Collaborative mining of distributed data streams in a mobile computing environment is referred to as Pocket Data Mining PDM. Hoeffding trees techniques have been experimentally and analytically validated for data stream classification. In this paper, we have proposed, developed and evaluated the adoption of distributed Hoeffding trees for classifying streaming data in PDM applications. We have identified a realistic scenario in which different users equipped with smart mobile devices run a local Hoeffding tree classifier on a subset of the attributes. Thus, we have investigated the mining of vertically partitioned datasets with possible overlap of attributes, which is the more likely case. Our experimental results have validated the efficiency of our proposed model achieving promising accuracy for real deployment.

KEYWORDS: Pocket Data Mining, Data Stream Mining, Distributed Data Mining.

1. INTRODUCTION

Pocket Data Mining *PDM* has been first coined by the authors in [10]. This new area of study aims at enabling collaborative mining of distributed streaming data in mobile computing environments. To realise such an application, autonomy and intelligence are two important features that the system should be characterised with as discussed in sufficient details in [10]. In response, we have proposed and experimentally validated in [10] that using mobile software agents [33] technology can efficiently realise such a system with a broad range of applications.

However, in [10] we have only provided a proof of concept without deploying any stream mining algorithms. In this paper, we proposed the use of distributed *Hoeffding trees* over vertically partitioned data streams with potential overlap of features. Hoeffding trees [5] is a data stream classification technique that makes use of the Hoeffding bound [7] to provide an approximate model with statistically guaranteed error bounds. To the best of our knowledge, this is the first work that adopts Hoeffding trees in a distributed environment over vertically partitioned data sets. The adoption of Hoeffding trees in this application is based on its proved efficiency as first reported in the work by Domingos et al [5,6], and then extended by Kirkby et al in [8,9].

We have used the implementation of Hoeffding trees technique from the Massive Online Analysis *MOA* tool [3] to implement our system. The technique has been wrapped in a mobile software agent using the JADE toolkit [12] adding both autonomy and intelligence to the technique. Our experimental results show the efficiency of the proposed technique. Various settings over real datasets have been used in the experiments.

The paper is organised as follows. Section 2 reviews the related work in the stream mining area. Our proposed method is detailed in Section 3. A thorough experimental study is given in Section 4. The paper is concluded in Section 5.

2. RELATED WORK

Related work in this area includes systems developed by Kargupta el al in [25,30,27,23,11] for mobile data mining in mobile brokering and road safety, and by Pirttikangas et al [29] for context-aware health club. Brief descriptions of these systems will follow.

Kargupta et al [25,30,27] have developed the first ubiquitous data stream mining system termed *MobiMine*. It is a client/server PDA-based distributed data mining application for financial data streams. The system prototype has been developed using a single data source and multiple mobile clients; however, the system is designed to handle multiple data sources. The server functionalities in the proposed system are data collection from different financial web sites and storage, selection of active stocks using common statistics methods, and applying online data mining techniques to the stock data. The client functionalities are portfolio management using a mobile micro-database to store portfolio data and information about user's preferences, and construction of the WatchList, which is the first point of interaction between the client and the server. The server computes the most active stocks in the market, and the client in turn selects a subset of this list to construct the personalized WatchList according to an optimisation module. The second point of interaction between the client and the server is that the server performs online mining, then transforms the results using Fourier transformation and finally sends this to the client. The client in turn visualises the results on the PDA screen. It is worth pointing out that the data mining process in MobiMine has been performed at the server side given the resource constraints of a mobile device.

With the increasing need for onboard data mining in resource-constrained computing environments and the notable rapid advances in the computational power of mobile devices, Kargupta et al [23] have developed *Vehicle Data Stream Mining System (VEDAS)*. It is a ubiquitous data stream mining system that allows continuous monitoring and pattern extraction from data streams generated on-board a moving vehicle. The mining component is located on the *PDA*. *VEDAS* uses online incremental clustering for modeling of driving behaviour. A commercial version of *VEDAS* termed as *MineFleet* has been successfully deployed [24,11].

Genie of the Net is an early attempt of adopting mobile software agents in ubiquiotous data stream mining. Pirttikangas et al [29] have implemented a mobile agent-based ubiquitous data mining for a context-aware health club for cyclists. The process starts by collecting information from sensors and databases in order to recognize the needed information for the specific application. This information includes user's context and other needed information collected by mobile agents. The main scenario for the health club system is that the user has a plan for an exercise. All the needed information about the health such as heart rate is recorded during the exercise. This information is analysed using data mining techniques to advise the user after each exercise.

Other related work includes the large body of data stream mining algorithms. Key techniques and approaches in the area are discussed in [13] and more recently in the tutorial presented by Gama et al in [29].

Addressing the resource constraints of small computational devices like smart phones and Personal Digital Assistants *PDAs* has been reported in work conducted by Gaber el al in [14,16,15,17]. The approach taken in this body of work has been termed as *Granularity-based* approach. It adapts the data mining algorithm to adjust the resource consumption pattern according to availability of resources. Notably, successful applications of the approach in road safety and healthcare have been reported in [20,21,22].

3. PDM: THE POCKET DATA MINING ARCHITECTURE

The architecture of our *PDM* framework is illustrated in Figure 1 [10]. The figure shows, the data stream mining process runs onboard the users' smart mobile phones. It is based on three basic agents:

- The **M**obile Agent **R**essource **D**iscoverer (MRD) which roams the network searching for data streams and available AMs that are relevant for the data mining task.
- **A**gent **M**iner (AM) which implements a data mining algorithm that trains a model on a local data source such as a data stream.
- The **M**obile **A**gent **D**ecision **M**aker (MADM) which moves to AMs that have been discovered by the MRD and appraised to be relevant for the data mining task, retrieves information about the data mining task from the AMs on which it bases a collective decision.

The abbreviations KNN, HT and NB stand for possible data mining algorithms that can be embedded in the AMs. *KNN* stands for the K-Nearest Neighbours, *HT* stands for Hoeffding Tree and *NB* for Naive Bayes classifiers.

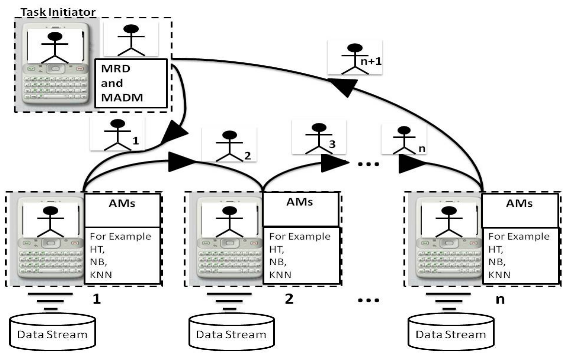

Figure 1. PDM Architecture

As the data streams come in, the model derived by the AMs is continuously updated to cope with the possible concept drift of the streaming environments. The process of stream mining is carried out using an AM. AMs are already distributed in the network before the data mining task is initiated. Some of these miners could be stationary and some others could be mobile. The smart phone on which the data mining task is initiated is called the `task initiator'. Stationary agents are instructed by the MRD to mine the streaming data on the mobile device without making any hops. However, the mobile agents could travel, instructed by the MRD, to one or more nodes in order to perform the mining task. The choice of using stationary or mobile agent 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 according to 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 of unlabeled instances, an MADM is fired to visit the nodes consulting the models about the local class label. While these agents are visiting the different nodes, it may decide to terminate its itinerary given that clearly there is a dominant class. This clearly makes the agent framework the suitable technology for this task.

4. DISTRIBUTED HOEFFDING TREES

A prototype of the PDM framework has been evaluated in computational terms elsewhere [10], in particular the communication performance and the parallel performance. With respect to communication performance it has been found that the communication overhead of mobile MADMs is very low. With respect to the parallel performance it has been found that the more MADMs are used the quicker all the AMs are visited. However, the version of PDM presented in [10] replaced the actual data mining algorithms by random result generators in order to simulate large numbers of AMs. Whereas [10] shows the computational applicability of PDM, this paper evaluates the first prototype of PDM with real data mining algorithms in terms of its applicability to mining streaming data in the context of classification.

The AMs implement Hoeffding classification trees from *MOA* [3]. The Hoeffding tree algorithm as depicted by Bifet and Kirkby in [4] is shown in Figure 2. Hoeffding trees have been designed for classifying high-speed data streams. Each AM runs a Hoeffding tree induction algorithm and the MADMs are used to collect the classification results. The owner of each AM may have subscribed to a certain subset of the data stream, or to different features of the same data. These might be features the owner of the local AM is particularly interested in. For some analysis tasks the currently subscribed features might be enough information, however, for classification of previously unseen whole data instances these features might be insufficient to achieve acceptable classification accuracy. As the owner of the local AM may not be subscribed to the full data stream, it cannot access the missing features itself. However, it can consult further AMs that belong to different owners and that may be subscribed to different features. These consulted AMs collaborate in order to classify the unseen data instances.

Algorithm 1 Hoeffding tree induction algorithm.

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

Figure 2. Hoeffding Tree Algorithm

The user can consult each available AM by sending one or more MADMs with the data instances to be classified to each AM. The MADM will collect the locally derived predictions on which it will base its decision for predicting the previously unseen data instances. However, the local prediction is not the only information delivered by the AM. Each AM also gives an estimate about its accuracy, which we refer to as `weight'. Basically each AM is fed with partial data instances it knows the classification of. The AM will treat a newly streamed data instance either as a test, or training instance. A probability with which a data instance is selected as test or training instance is to be defined by the owner of the AM. If the instance is selected as test instance, then the Hoeffding tree tries to classify it in order to estimate the `weight' of the local AM.

In order to take concept drifts into consideration when estimating the `weight', it is important that older test instances are not being taken into consideration. The maximum number of test instances that are taken into consideration is defined at the startup of the AM. Now if the maximum number of test instances is for example 20 and there have been already 20 test instances selected,

then the oldest of the 20 test instances is replaced by the next newly selected test instance.
The scenario of a classification task of the task initiator agent could be described in the following steps:

1. Start the MRD agent to discover `classifier AMs' with a for the classification task relevant data stream. The MRD agent will return with a list of relevant AMs to be used.
2. When the MRD agent returns, an MADM is started that loads the unlabelled data instances for classification and hops to the AMs in the list derived by the MRD agent.
3. On each visited AM the MADM asks the AM to predict the classification of its unlabelled instances and also retrieves the AM's `weight' or estimated accuracy.
4. The MADM returns to the task initiator and performs a weighted majority voting for each unlabelled data instance using each AM's prediction and `weight' in order to give the final prediction.

The basic idea is that the MADM hops with its instances for classification to each AM and retrieves the predicted classification for each instance plus the `weight' of the AM on the data stream. After the MADM visited all AMs, a `weighted' majority voting is applied in order to derive the final classification. For example if there are three AMs *A*, *B* and *C* and one data instance to predict its class, assuming that *A* predicts class *X* with a `weight' of 0.55, AM *B* predicts *X* with a `weight' of 0.2 and AM *C* predicts class *Y* with a `weight' of 0.8, then the `weighted' classification result would be for class *X* 0.75 (0.55+0.2=75) and for class *Y* 0.8. Thus the MADM would choose *Y* as the predicted class.

In general the `weight' is referred to the estimated accuracy on the data stream evaluated by the AM, whereas the actual `predictive accuracy' is only for experimental purposes, in order to assess the AM's actual performance.

4.1. Experimental Setup

This version of the PDM framework is based on the implementation discussed in [10], which is described in Section 3 and has been empirically evaluated. For the implementation the well known JADE framework has been used [34], with the reasoning that there exist a version of JADE, JADE-LEAP *(Java Agent Development Environment-Lightweight Extensible Agent Platform)*, that is designed for the implementation of agents on mobile devices and can be retrieved from the JADE project website as an `add on' [34]. As JADE works on standard PCs as well as on mobile devices, it was possible to develop and test the first prototype of the PDM framework on a test LAN. The LAN consists of 8 PCs with different hardware configurations, which are connected using a standard `CISCO Systems' switch of the catalyst 2950 series.

The Hoeffding tree implementation used in these experiments is extracted from the `Mobile Online Analysis' tool which is based on WEKA [35,3] libraries and thus allows the usage of data files in the `.arff' format. In our experimental setup we used 8 AMs each running a Hoeffding tree induction algorithm and one MADM collecting classification results. Each Hoeffding tree on each AM is only using the features the AM is subscribed to in order to train and update the classifier. The AMs in the current implementation can be configured that they only take specific features into account or a certain percentage of randomly selected features out of the total number of features. The latter configuration is for our experimental purposes.

The data streams were simulated using the datasets described in Table 1. Datasets for tests 1, 2, 5 and 6 were retrieved from the UCI data repository [1] and datasets for tests 3 and 4 were retrieved from the Infobiotics benchmark data repository [2]. As discussed above, the data stream takes a random data instance from the dataset and with a predefined probability uses it as a test instance in order to calculate the `weight' of the AM or uses it as a training instance for the local Hoeffding tree. Please note that instances might be selected more than once by the data stream, however, if a data instance has been used as a test instance, it will be removed from the stream and never selected again, in order to avoid overfitting on test instances. Also as discussed above, a maximum number of test instances is predefined. If the maximum number is reached, then the oldest test instance is replaced by the next newly selected test instance.

Table 1. Evaluation Datasets

Test Number	Dataset	Number of Attriubtes
1	waveform-5000	40
2	mushroom	22
3	infobiotics 1	20
4	infobiotics 2	30
5	kn-vs-kr	36
6	spambase	57

4.2. Evaluation

PDM using Hoeffding trees has been evaluated in terms of its classification accuracy using the datasets described in Table 1. The fact that the datasets in Table 1 are batch files allowed us to compare PDM's accuracy with Hoeffding trees to batch learning classification algorithms, in particular PDM's accuracy is compared to the C4.5 algorithm [31]. 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. In general it is expected that the accuracy of PDM will become better, the more features each AM has available to produce its Hoeffding tree. This is due to the reason that some features are more relevant and some are less. Thus simply the more features the AM has available, the more likely it is that these features are sufficient to describe the concept. Also we expect that the more AMs are visited by the MADM the more likely it is that a good predictive accuracy is achieved. This is because it is also more likely that there is at least one AM visited with a good performance on the MADM's instances and also produces a high `weight'. For all experiments in this Section 30% of the data file has been taken as test instances and the remaining 70% as streaming instances for the AM.

Figure 3 shows the total accuracy of PDM's MADM plotted versus the number of AMs visited by the MADM. The experiments have been conducted for AM's holding a percentage of features from the total feature space, in particular 20%, 30% and 40% of the total feature space. The features an AM holds have been randomly selected for these experiments, however, it is possible that different AMs may have selected the same or partially the same features.

It can be seen in Figure 3 that the general tendency is: the more features the AM holds the higher the accuracy achieved by the MADM. Also plotted in Figure 3 is the accuracy of C4.5 achieved by learning the stream data instances in batch mode using all attributes. It can be seen that the accuracies achieved by PDM using Hoeffding trees often come close to the accuracies achieved by C4.5 especially for tests 1, 2, 3 and 4. In tests 5 and 6 even if the accuracy cannot compete with C4.5 it is still acceptable, ranging around 70% except for Test 5 with 20% attributes.

Now looking in Figure 3 there are at least 4 striking outliers to the overall trend of the data series. In particular for Test 6 with 40% attributes and 1 and 2 visited AMs; Test 4 for 40% attributes and 1 and 2 visited AMs; Test 4 for 30% attributes and 1 and 2 visited AMs and Test 4 for 20% attributes and 8 visited AMs. The actual MADM has been implemented that it also shows the actual accuracies achieved by each AM by classifying the MADM's test data. This local accuracy is different to the `weight', as the `weight' is calculated using data instances from the stream and not the ones from the MADM.

Table 2 compares the weights and accuracies for each of these outliers. For Test 6 with 40% attributes and two visited AMs it can be seen that the discrepancy between the weight (0.87) and actual accuracy (0.57) of AM 1 is quite large, whereas the weight of AM 2 is close to the actual accuracy. This discrepancy in `weight' compared with the actual local accuracy causes the concerning AM to have a too strong influence on the MADM's classification mechanism. For Test 6 with only one visited AM the discrepancy is quite large and thus causes a very low classification accuracy. Also for Test 4 with 40% attributes and one visited AM there is a large discrepancy at AM 1; for Test 4 with 40% attributes and two visited AMs there is a large discrepancy at AM 1; for Test 4 with 30% attributes and one visited AMs there is a large discrepancy; for Test 4 with 30% attributes and two visited AMs there is a large discrepancy at AM 1 and for Test 4 with 20% attributes and 8 visited AMs there are large discrepancies for AMs 4, 6, 7 and 8. These discrepancies have also been observed for non outliers, however it has also been observed that only for outliers at least half of the visited AMs had large discrepancies.

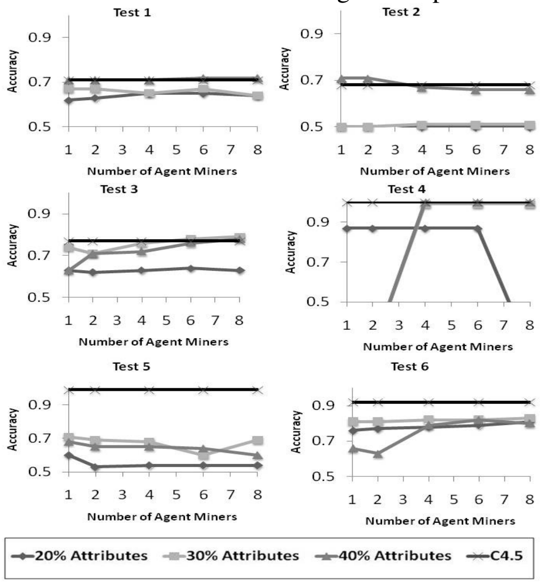

Figure 3. PDM Classification Accuracy

Table 2. Weights and Local Accuracies for Outliers

	Test 6 40% Attributes		Test 6 40% Attributes	
MADM's	0.63		0.66	
	Weight	Accuracy	Weight	Accuracy
AM 1	0.87	0.57	0.86	0.66
AM 2	0.85	0.85		
	Test 4 40% Attributes		Test 4 40% Attributes	
	0.66		0.26	
	Weight	Accuracy	Weight	Accuracy
AM 1	0.86	0.66	0.99	0.26
AM 2			0.99	0.99
	Test 4 30% Attributes		Test 4 30% Attributes	
	0.66		0.26	
	Weight	Accuracy	Weight	Accuracy
AM 1	0.86	0.66	0.99	0.26
AM 2			0.99	1
	Test 4 20% Attributes			
	0.26			
	Weight	Accuracy		
AM 1	0.91	0.87		
AM 2	0.9	0.87		
AM 3	0.9	0.87		
AM 4	0.99	0.26		
AM 5	0.9	0.87		
AM 6	0.99	0.26		
AM 7	0.99	0.26		
AM 8	0.99	0.26		

Figure 4 shows the actual accuracy achieved by the MADM as already shown in Figure 3 and the average of the local accuracies achieved by the visited AMs versus the number of AMs that have been visited. Please note that when the term `local accuracy' is used then this accuracy is calculated on the test data from the MADM and is not the estimated `weight'. Each row of plots in Figure 4 corresponds to the data of one of the datasets from Table 1 and each column of plots corresponds to a different percentage of features loaded by the AMs. The darker line in the plots in Figure 4 corresponds to the actual achieved accuracy by the MADM and the lighter line to the total average accuracy of all AMs.

It can be seen in Figure 4 that the general tendency is that the global accuracy achieved by the MADM by weighted majority voting is often higher than the average local accuracies of all AMs visited. In general it can be seen that the more AMs are visited the more likely it is that the global accuracy is higher than the average accuracy. The reason is that with more AMs it is simply more likely that there is at least one AM that achieves a good accuracy together with a good `weight' and thus overall has a higher influence on the total voting system of the MADM.

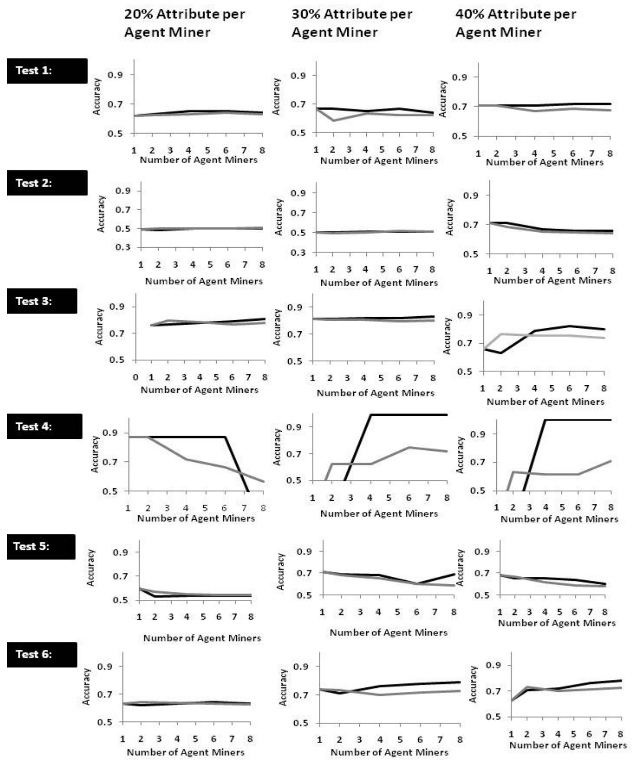

Figure 4. Averages versus Actual Results

5. CONCLUSIONS

The paper introduced an advanced version of the *Pocket Data Mining* framework to enable collaborative mining of streaming data in mobile environments. The framework uses mobile software agents technology in order to mine local data streams collaboratively. Whereas the computational feasibility has been proved elsewhere [10], this paper experimentally evaluates the applicability of the system for classification tasks based on Hoeffding trees. In general it could be observed that the more agents are visited, the better the classification accuracy. There have also been a few outliers for which the classification accuracy dropped considerably. This is expected to be fixed using a rating approach, which we are currently working on.

REFERENCES

[1] 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.

[2] Bacardit J. and Krasnogor N., “The Infobiotics PSP benchmarks repository”., http://www.infobiotic.net/PSPbenchmarks, 2008.

[3] Bifet A., Holmes G., Pfahringer B., Kranen P., Kremer H., Jansen T., Seidl T., *MOA: Massive online analysis*, Journal of Machine Learning Research (JMLR), 2010.

[4] Bifet A. and Kirkby R., “Data Stream Mining: A Practical Approach”, Center for Open Source Innovation, August 2009.

[5] Domingos P. and Hulten G., “Mining high-speed data streams”, In International Conference on Knowledge Discovery and Data Mining, pages 71-80, 2000.

[6] Hulten G., Spencer L., Domingos P., “Mining time-changing data streams”, Proceedings of ACM KDD 2001, pp. 97-106, ACM press

[7] Hoeffding W., *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association, 58(301):13-30, 1963.

[8] Holmes G., Kirkby R., and Pfahringer B., “Stresstesting Hoeffding trees”, In European Conference on Principles and Practice of Knowledge Discovery in Databases, pages 495-502, 2005.

[9] Kirkby R., IMPROVING HOEFFDING TREES. PhD thesis, University of Waikato, November 2007. Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby. New options for hoeffding trees. In AI, pages 90-99, 2007.

[10] 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.

[11] Agnik, "MineFleet Description", http://www.agnik.com/mineeet.html

[12] Bellifemine F., Poggi A., and Rimassa G., "Developing multi-agent systems with JADE", in C. Castelfranchi and Y. Lesperance, (eds.), Intelligent Agents VII. Agent Theories Architectures and Languages Workshop Proceedings, Boston, MA, USA, July 7-9, 2000, volume 1986 of LNCS, pages 89 - 103. Springer Verlag, 2000.

[13] Gaber, M, M., Zaslavsky, A., and Krishnaswamy, S., *Mining Data Streams: A Review*, ACM SIGMOD Record, Vol. 34, No. 1, pp. 18-26, June 2005, ISSN: 0163-5808.

[14] Gaber M. M., and Yu P. S., *A Holistic Approach for Resource-aware Adaptive Data Stream Mining*, Journal of New Generation Computing, Volume 25, Number 1, 2006, pp. 95-115, Ohmsha, Ltd., and Springer Verlag.

[15] Phung N. D., Gaber M. M., and Roehm U, "Resource-aware Online Data Mining in Wireless Sensor Networks", Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2007, pp. 139-146.

[16] Gaber M. M., "Data Stream Mining Using Granularity-based Approach", a book chapter in Foundations of Computational Intelligence, Volume 6, Volume 206/2009, pp. 47-66, Springer, Germany, 2009.

[17] Gaber, M, M., Zaslavsky, A., and Krishnaswamy, S., A "Cost-Effcient Model for Ubiquitous Data Stream Mining", Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2004), pp. 747-754, Italy, July 4-9.

[18] Gama J., and Gaber M. M. (Eds), LEARNING FROM DATA STREAMS: PROCESSING TECHNIQUES IN SENSOR NETWORKS, Springer Verlag, 2007.

[19] Gama J., Gaber M. M., and Krishnaswamy S., "Data Stream Mining: From Theory to Applications and From Stationary to Mobile", a tutorial presented in the ACM 25th Symposium On Applied Computing.

[20] Haghighi P. D., Zaslavsky A., Krishnaswamy S., Gaber M. M., "Mobile Data Mining for Intelligent Healthcare Support", Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS08), pp. 1-10, Hawaii, USA, January 5-8, 2009, IEEE 2009.

[21] Horovitz O., Gaber M. M., and Krishnaswamy S., "Making Sense of Ubiquitous Data Streams: A Fuzzy Logic Approach", Proceedings of Knowledge-Based Intelligent Information and Engineering Systems, KES 2005, pp. 922-928, Melbourne, Australia, September 14-16, 2005, LNCS 3682, Springer, 2005.

[22] Horovitz, O., Krishnaswamy, S., and Gaber, M, M., *A Fuzzy Approach for Interpretation of Ubiquitous Data Stream Clustering and Its Application in Road Safety*, Intelligent Data Analysis, Vol. 25, No. 1, pp 89-108, 2007, IOS Press.

[23] Kargupta H., Bhargava R., Liu K., Powers M., Blair P., Bushra S., Dull J., Sarkar K., Klein M., Vasa M., Handy D., VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring, Proc. of the SIAM International Data Mining Conference, 2004.

[24] 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, Volume 25, no. 1, 2007.

[25] 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.

[26] Kargupta H., Hamzaoglu I. and Stafford B., "Scalable, Distributed Data Mining Using an Agent-Based Architecture", Proceedings of Knowledge Discovery and Data Mining, pp. 211-214, 1997, AAAI Press.

[27] Kargupta H., Sivakumar K., and Ghosh S., Dependency "Detection in MobiMine and Random Matrices", Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 250-262, 2002.

[28] Page J., Padovitz A., and Gaber M. M., "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.

[29] Pirttikangas S., Riekki J., Kaartinen J., Miettinen J., Nissila S., and Roning J., "Genie Of The Net: A New Approach For A Context-Aware Health Club", Proceedings ECML'01/ PKDD'01. September 3-7, 2001, Freiburg, Germany.

[30] Pittie S., Kargupta H., and Park B., *Dependency Detection in MobiMine: A Systems Perspective*, Information Sciences Journal. Volume 155, Issues 3-4, pp. 227-243, Elsevier.

[31] Quinlan, J. R. C4.5: PROGRAMS FOR MACHINE LEARNING. Morgan Kaufmann Publishers, 1993.

[32] Silva J. D., Giannella C., Bhargava R., Kargupta H., and Klusch M., *Distributed Data Mining and Agents*, Engineering Applications of Artificial Intelligence Journal, 2005 volume 18, pp. 791-807.

[33] Zaslavsky A., "Mobile Agents: Can They Assist with Context Awareness?", IEEE MDM, Jan. 2004 , California.

[34] JADE-LEAP: http://jade.tilab.com/.

[35] Witten I. and Frank E., DATA MINING: PRACTICAL MACHINE LEARNING TOOLS AND TECHNIQUES WITH JAVA IMPLEMENTATIONS, Morgan Kaufmann, Second Edition, 2005.