Distributed Classification for Pocket Data Mining

Frederic Stahl[1], Mohamed Medhat Gaber[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. Distributed and collaborative data stream mining in a mobile computing environment is referred to as Pocket Data Mining *PDM*. Large amounts of available data streams to which smart phones can subscribe to or sense, coupled with the increasing computational power of handheld devices motivates the development of *PDM* as a decision making system. This emerging area of study has shown to be feasible in an earlier study using technological enablers of mobile software agents and stream mining techniques [1]. A typical *PDM* process would start by having mobile agents roam the network to discover relevant data streams and resources. Then other (mobile) agents encapsulating stream mining techniques visit the relevant nodes in the network in order to build evolving data mining models. Finally, a third type of mobile agents roam the network consulting the mining agents for a final collaborative decision, when required by one or more users. In this paper, we propose the use of distributed Hoeffding trees and Naive Bayes classifiers in the *PDM* framework over vertically partitioned data streams. Mobile policing, health monitoring and stock market analysis are among the possible applications of *PDM*. An extensive experimental study is reported showing the effectiveness of the collaborative data mining with the two classifiers.

1 Introduction

Recent and continuous advances in smart mobile devices have opened the door for running applications that were difficult or impossible to run in the past in such resource-constrained environments. The clear trend is to have more applications running locally on these devices given their computational and sensing capabilities. Recent important applications in the area of activity recognition [9, 14] have stimulated our recent research activities. Therefore, we have proposed the new area of pocket data mining in [1].

Pocket data mining has been first coined in [1] to describe the process of mining data streams collaboratively in a mobile computing environment. The *PDM* framework supports the process from resource discovery to the learning and usage phases. The core of the framework is the set of data stream mining

techniques that work together to synthesize a global outcome. The mining techniques are assigned to the mobile devices and are encapsulated as mobile software agents that are able to move and be cloned. This assignment is done based on the availability of resources and features of the data available to the mobile device. We have proved in [1] the computational feasibility of the *PDM* framework. However, we have not employed any stream mining technique in our previous implementation. Thus, in this paper, we propose the use of two data stream classification techniques. We have chosen Hoeffding trees and Naive Bayes classifiers due to the following reasons. First, Hoeffding trees classifiers have proved their efficiency as the state-of-the-art data stream classification technique as reported in [3]. Second, Naive Bayes is a lightweight naturally incremental classifier.

The paper is organised as follows. Section 2 enumerates the related work. The architecture of the PDM framework including the details of the used algorithms is given in Section 3. Extensive experimental study is presented in Section 4. Ongoing work and concluding remarks and be found in Section 5.

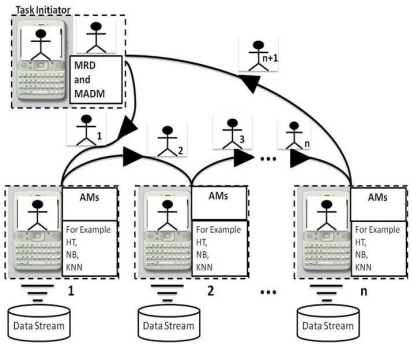
2 Related Work

Distributed data mining techniques have been thoroughly reviewed by Park and Kargupta in [16]. On the other hand, the field of data stream mining has been concisely reviewed in [11]. More recently, the utilisation of smart phones' sensing capabilities to be used to learn about the user's activities have been explored in [13, 9]. It has to be noted that none of the above techniques has explored the great potential of collaborative mining of data streams in the mobile ad hoc computing environments, including the work proposed by Miller et el [15], that only focused on recommender systems for any type of connected devices. Our work rather attempts to exploit data stream mining techniques for ad hoc analysis using smartphones in critical applications.

3 PDM: The Pocket Data Mining Architecture

The architecture of the *PDM* framework is highlighted in Figure 1. The basic scenario of *PDM* displayed in Figure 1 involves the following generic (mobile) software agents [1]: (a) (Mobile) **A**gent **M**iners (AM) are distributed over the network and located on the local mobile devices, they implement data mining algorithms; (b) **M**obile Agent **R**essource **D**iscoverers are used to explore the network and locate computational ressources, data sources and AMs; (c) **M**obile **A**gent **D**ecision **M**akers roam the network consulting AMs in order to retrieve information or partial results for the data mining task. It has to be noted that we use the terms *PDM architecture* and *PDM framework* interchangeably. Any smart phone in the network can own any kind of *PDM* agents. The smart phone from which a data mining task is initiated is called the task initiator. The AMs in *PDM* can implement any data mining algorithm such as Hoeffding Decision Trees (HT), Naive Bayes (NB) or K Nearest Neighbours (K-NN). As the left hand side of Figure 1 shows, these data mining algorithms are embedded in AMs which

run on-board a users smart phone. The smart phone may be subscribed to a data stream. The data mining model that is generated by the AMs is continuously updated in order to cope with possible concept drifts of the data stream. AMs may be stationary agents already installed for the use by the owner of the smart phone but may also be mobile agents distributed at the beginning of the data mining task. The right hand side of Figure 1 shows the pseudo code of the basic PDM workflow.



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*;
for $i = 1$ to $i =$ *number of AMs* do
  repeat
    AM_i: mine streaming data;
  until Use of the model by *MADM*
end for

Fig. 1. PDM Architecture

For example in the context of classification, if the task initiator at any point in time decides to use the models of remotely located AMs to classify a set of unlabelled data instances, an MADM and an MRD can be used. The MRD is roaming the network to discover available AMs and data streams onboard the smart phones. The MADM then loads the unlabelled instances and visits relevant AMs, as determined by the MRD, in order to collect their predictions for the correct class labels. While the MADM agent visits different nodes in the network, it might decide to terminate its itinerary based on a stopping criterion such as confidence level of the already collected predictions, or a time limit.

The current implementation, which is evaluated in the paper, has two different AMs for classification tasks, namely Hoeffding Tree [3] and Naive Bayes classifiers. However, the *PDM* framework allows the use of any classification technique. The Hoeffding tree classifier from the *MOA* tool as illustrated by Bifet and Kirkby in [2] is shown in Figure 2. Hoeffding tree classifiers have been designed for high speed data streams. On the other hand, the Naive Bayes classifier has been developed for batch learning, however it is naturally incremental. The current implementation of *PDM* uses the Naive Bayes classifier from the *MOA* tool [2] which is based on the Bayes Theorem [14] stating that if $P(C)$ 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 $P(C|X) = \frac{P(X|C)P(C)}{P(X)}$. According to the Bayes Theorem, the Naive Bayes algorithm assigns a data instance to the class it belongs to with the highest probability. As Naive Bayes

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**

Fig. 2. Hoeffding Tree Algorithm

generally performs well [10, 12] and is naturally incremental, it is suitable for classifying data streams.

4 Implementation and Evaluation

As this paper examines PDM's applicability to classification rule induction on data streams, both Hoeffding tree and Naive Bayes classifiers have been thoroughly tested.

4.1 Experimental Setup

We use the implementation of both classifiers in *MOA* toolkit [2] which is based on the WEKA library [4]. We compare two *PDM* configurations, one where all AMs are based on Hoeffding trees and the other where all AMs are based on Naive Bayes. Owners of different AMs may have subscribed to overlapping subsets of the feature space of the same data stream, as there may be features that are particularly interesting for the owner of a local AM. However, the current subscription may be insufficient for classifying new data instances. Subscribing to more features may not be desirable for many reasons, such as that it may lead to higher subscription fees or confidentiality constraints. However the owner of the local AM sends an MADM that visits and consults further AMs that belong to different owners. The visited AMs are potentially subscribed to different features and the MADM consults the AMs to classify the unlabelled data instances. The classification results and accompanying information from the local AMs are collected and used by the MADM to decide for a final classification. The accompanying information of the AMs is an estimate of the AM's own confidence which is referred to as 'weight' in this paper. In *PDM*, each AM takes with a previously defined probability a labelled streamed data instance as training or as test instance. In the current setup the probability that an instance is selected

as test instance is 20% and as training instance is 80%. The test instances are used by the local AM to estimate its local classification accuracy/confidence or 'weight'. Concept drifts are also taken into account when the weight is calculated. This is done by defining a maximum number of test instances at the startup of an AM by the owner. For example, if the maximum number of test instances is 20, and already 20 test instances have been selected then the oldest test instance is replaced by the next newly selected test instance and the 'weight' is recalculated.

The MADM hops with the instances to be classified to each available AM, requesting to classify the instances and also retrieves the AM's weight. After the MADM's schedule is processed, the MADM derives the final classification for each data instance by 'weighted majority voting'. For example, if there are three AMs A, B and C and one data instance to classify, AM A predicts class X with a 'weight' of 0.55, AM B predicts class X with a 'weight' of 0.2 and AM C predicts class Y with a 'weight' of 0.8, then MADM's 'weighted' prediction would be for class X $0.55 + 0.2 = 0.75$ and for class Y 0.8 and as Y yielded the highest vote, the MADM would label the instance with class Y.

For the implementation of *PDM* the well known JADE framework has been used [5], 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' [6]. As JADE works on standard PCs as well as on mobile devices, it is possible to develop and evaluate the *PDM* framework on a test LAN. The LAN consists of 8 PCs with different hardware configurations, which are connected using a standard CISCO System switch of the catalyst 2950 series. In our experimental setup, we used 8 AMs each running a Hoeffding tree induction algorithm or Naive Bayes, and one MADM collecting classification results. The AMs in the current implementation can be configured so that they only take specific features into account or a particular percentage of randomly selected features out of the total number of features. The latter configuration is for our experimental purposes as in the real application we may not know which features a certain AM is subscribed to.

Table 1. Evaluation Datasets

Test Number	Dataset	Number of Attributes
1	kn-vs-kr	36
2	spambase	57
3	waveform-500	40
4	mushroom	22
5	infobiotics 1	20
6	infobiotics 2	30

The data streams were simulated using the datasets described in Table 1. Datasets for tests, 1, 2, 3 and 4 were retrieved from the UCI data repository [7] and datasets for tests 5 and 6 were retrieved from the Infobiotics benchmark

data repository [8]. The simulated data stream takes a random data instance from the dataset and feeds it to the AM. 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 of the AM's model and calculate the 'weight' on test instances.

4.2 Distributed Hoeffding Trees

The fact that the datasets in Table 1 are batch files allows us to compare PDM's accuracy with Hoeffding trees to batch learning classification algorithms, in particular PDM's accuracy is compared to accuracy of C4.5 and the accuracy of Naive Bayes. Both batch implementations were retrieved from WEKA [4]. 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, the more features an AM has available and the more AMs visited, the more likely it is to have a better accuracy of the global classification. However, some features may not be relevant to the classification task and introduce unnecessary noise. For all experiments in this section, 30% of the data is taken as test instances for the MADM and the remaining 70% for training the AMs. All experiments have been conducted 5 times and the achieved local accuracies on the AMs and the achieved accuracy of the MADM has been recorded and averaged.

The left hand side of Figure 3 shows the accuracy of *PDM* 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. Looking at the left hand side of Figure 3, it can be seen that the batch versions of C4.5 and Naive Bayes achieve a comparable accuracy on all datasets. The largest discrepancy between both algorithms is for Test 2 where Naive Bayes's accuracy was 80% and C4.5 91%. Regarding PDM's classification accuracy, it can be seen that in all cases the achieved accuracy is no less than 50%. In general, it can be observed that for configurations of *PDM* that use AMs with only 20% of the attributes, PDM's classification accuracy is low compared with configurations that use 30% or 40% of the attributes. Also there does not seem to be a large discrepancy between configurations that use 30% or 40% of the attributes, which may be due to the fact that it is more likely with 40% attributes that irrelevant attributes have already been selected. In general, it can be seen that in many cases PDM's classification accuracy is close to the batch classification accuracy of C4.5 and Naive Bayes, especially for tests 3 and 5, however also for the remaining tests *PDM* often achieves close accuracies compared to those achieved from the batch learning algorithms. In general PDM with Hoeffding Trees achieves an acceptable classification accuracy. The right hand side of Figure 3 shows the

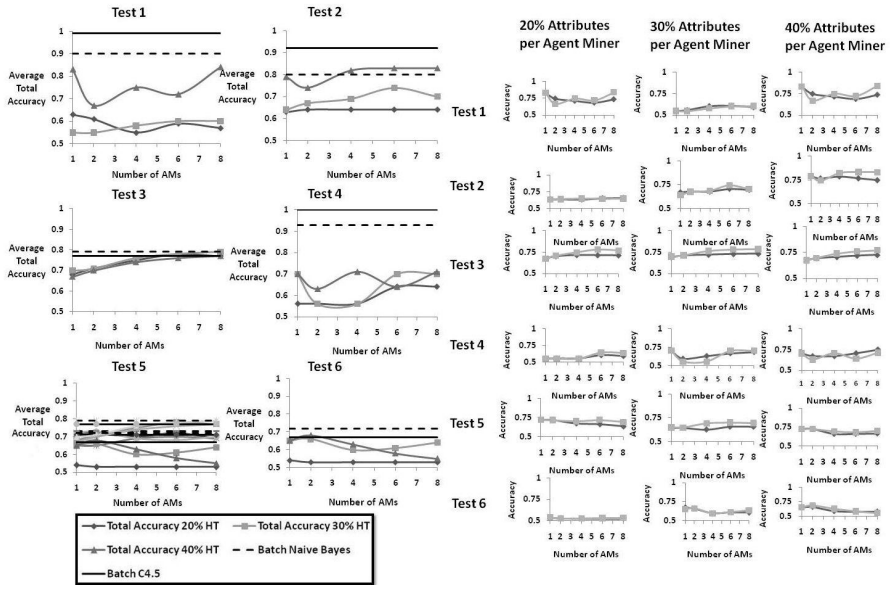


Fig. 3. The left hand side of the figure shows PDM's classification accuracy based on Hoeffding Trees and the right hand side of the figure the average accuracy of the MADM (using 'weighted' majority voting) versus the average accuracy of the AMs based on PDM with Hoeffding Trees.

accuracy achieved by the MADM (using 'weighted' majority voting) and the average of the local accuracies achieved by the AMs versus the number of AMs that have been visited. Each row of plots on the right hand side in Figure 3 corresponds to one of the datastets listed in Table 1 and each column of plots corresponds to a different percentage of features subscribed to by the AMs. The darker lines in the plots correspond to the average accuracy of the AMs and the lighter lines correspond to the accuracy the MADM derived using the local AM's 'weights' and classifications. It can be observed that in most cases the 'weighted' majority voting either achieves a similar or better accuracy compared with simply taking the average of the predictions from all AMs.

4.3 Distributed Naive Bayes

Similarly, *PDM* using Naive Bayes has been evaluated the same way as Hoeffding Trees described in Section 4.2. Similar results compared with Section 4.2 are expected with *PDM* using Naive Bayes classifiers.

Figure 4 illustrates the data obtained with *PDM* using Naive Bayes the same way as in Figure 3. The left hand side of Figure 4 shows the total accuracy of *PDM* plotted versus the number of AMs visited by the MADM. Regarding *PDM*'s classification accuracy, again it can be seen that in all cases the achieved

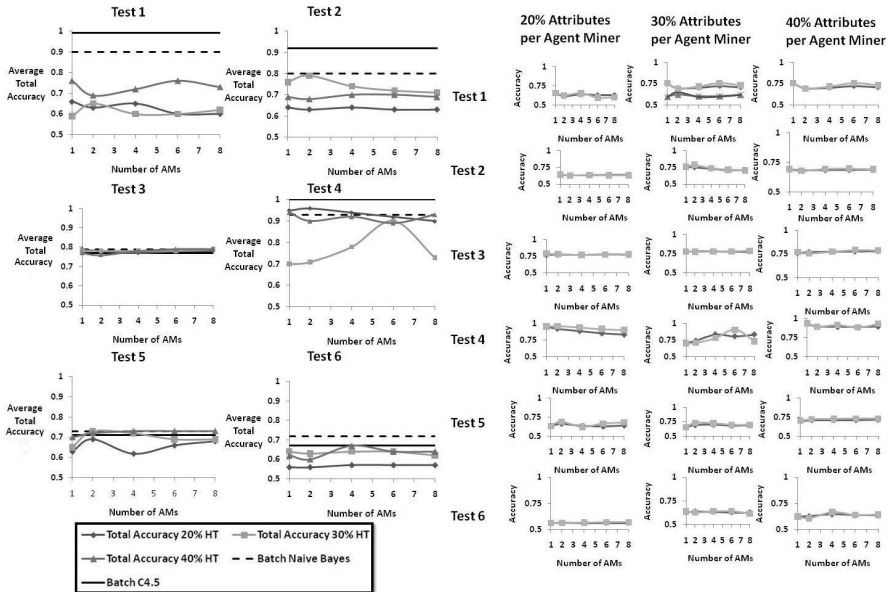


Fig. 4. The left hand side of the figure shows PDM's classification accuracy based on Naive Bayes and the right hand side of the figure shows the average accuracy of the MADM (using 'weighted' majority voting) versus the average accuracy of the AMs based on PDM with Naive Bayes.

accuracy is not less than 50%. In general, it can be observed that for configurations of PDM that use AMs with only 20% of the attributes PDM's classification accuracy is low compared with configurations that use 30% or 40% of the attributes. In general, it can be seen that in most cases that PDM's classification accuracy is close to the batch classification accuracy of C4.5 and Naive Bayes, especially for tests 3, 4, 5 and 6, however also for the remaining tests *PDM* often achieves acceptable accuracies. 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. In general *PDM* with Naive Bayes AMs achieves an acceptable classification accuracy. The right hand side of Figure 4 shows the accuracy achieved by the MADM (using 'weighted' majority voting) and the average of the local accuracies achieved by the AMs versus the number of AMs that have been visited. Each row of plots in Figure 4 corresponds to one of the datasets listed in Table 1 and each column of plots corresponds to a different percentage of features subscribed to by each AM. The darker lines in the plots correspond to the average accuracy of the AMs and the lighter lines correspond to the accuracy the MADM derived using the local AM's 'weights' and classifications. Similar to the Hoeffding tree results, It can be observed that in most cases the 'weighted' majority voting either achieves a similar or better accuracy compared with simply taking the average of the predictions from all AMs.

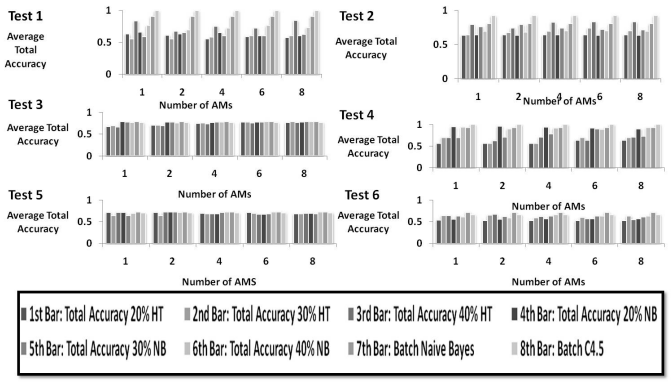


Fig. 5. Classification Accuracies achieved by both, PDM with Hoeffding Trees and PDM with Naive Bayes.

The bar charts in Figure 5 show for each number of used AMs the accuracies of *PDM* 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 Hoeffding 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. For tests 3 and 5, PDM with Naive Bayes AMs and PDM with Hoeffding tree AMs seem to achieve an equal performance concerning the classification accuracy. In the remaining tests 1, 2, 4 and 6, there seems to be no general bias towards one of the two approaches, sometimes PDM with Hoeffding tree AMs is slightly better than PDM with Naive Bayes AMs and vice versa. The fact that there doesn't seem to be a bias towards one of the approaches suggest that heterogeneous PDM configurations with some AMs implementing Naive Bayes and some implementing Hoeffding trees would generate a similar performance compared with PDM systems solely based on Naive Bayes or Hoeffding trees.

5 Conclusions

This paper outlines the Pocked Data Mining (PDM) architecture, a framework for collaborative data mining on data streams in a mobile environment. *PDM* uses mobile agents technology in order to facilitate mining of data streams collaboratively. *PDM* has been evaluated concerning its achieved classification accuracy for two different configurations, one with Hoeffding Tree AMs and one with Naive Bayes AMs. It has been observed that both configurations achieve an acceptable classification accuracy. Often PDM even achieves close accuracies compared to the ideal case, where all instances and attributes are available in

a batch file, so that a batch learning algorithm such as C4.5 or Naive Bayes can be applied. Also it does not seem that in *PDM*, one of the used classifiers (Hoeffding Tree or Naive Bayes) is superior to the other, both setups achieve very similar results. Also it has been observed that PDM's weighted majority voting achieves a better classification accuracy compared with simply the taking the local average accuracies of all AMs.

PDM opens a powerful yet so far widely unexplored distributed data mining niche. The particular *PDM* implementation outlined in this paper for classification just scratches the surface of possibilities of collaborative data mining on mobile devices.

References

1. 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.
2. Bifet A. and Kirkby R., Data Stream Mining: A Practical Approach, Center for Open Source Innovation, August 2009.
3. Domingos P. and Hulten G., Mining high-speed data streams, In International Conference on Knowledge Discovery and Data Mining, pages 71-80, 2000.
4. Witten I. and Frank E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, Second Edition, 2005.
5. 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.
6. JADE-LEAP: http://jade.tilab.com/.
7. 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.
8. Bacardit J. and Krasnogor N., *The Infobiotics PSP benchmarks repository.*, http://www.infobiotic.net/PSPbenchmarks, 2008.
9. Choudhury T., Borriello G., et al. The Mobile Sensing Platform: An Embedded System for Activity Recognition. Appears in the IEEE Pervasive Magazine - Special Issue on Activity-Based Computing, April 2008
10. Domingos P. and Pazzani M. J.. On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning, 29(2/3):103-130, 1997.
11. 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.
12. J. Hilden. Statistical diagnosis based on conditional independence does not require it. Computers in Biology and Medicine, 14(4):429-435, 1984
13. N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A Survey of Mobile Phone Sensing, IEEE Communications, Sep. 2010.
14. Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of bayesian classifiers. In National Conference on Artificial Intelligence, pages 223–228, 1992.
15. B. N. Miller, J. Konstan, and J. Riedl, PocketLens: Toward a personal recommender system, ACM Transactions on Information Systems Vol 22(3), 2004
16. B. Park and H. Kargupta, Distributed Data Mining: Algorithms, Systems, and Applications, Data Mining Handbook. Editor: Nong Ye, 2002.