Techniques for Estimating the Computation and Communication Costs of Distributed Data Mining·

Shonali Krishnaswamy[1], Arkady Zaslavsky[2], Seng Wai Loke[3],

[1]School of Network Computing, Monash University (Peninsula Campus)
McMahons Rd, Frankston, Victoria –3199, Australia.
Shonali.Krishnaswamy@Infotech.monash.edu.au
[2]School of Computer Science and Software Engineering
900 Dandenong Road, Monash University, Caulfield East, Victoria –3145, Australia.
arkady.zaslavsky@csse.monash.edu.au
[3]School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Melbourne, Victoria-3001, Australia.
swloke@cs.rmit.edu.au

Abstract. Distributed Data Mining (DDM) is the process of mining distributed and heterogeneous datasets. DDM is widely seen as a means of addressing the scalability issue of mining large data sets. Consequently, there is an emerging focus on optimisation of the DDM process. In this paper we present cost formulae for estimating the communication and computation time for different distributed data mining scenarios.

1 Introduction

Distributed data mining (DDM) addresses the specific issues associated with the application of data mining in distributed computing environments, which are typically characterised by the distribution of users, data, hardware and mining software. DDM is widely seen as a means of addressing the scalability issue of mining large data sets. There are predominantly two architectural models used in the development of DDM systems, namely, *client-server* (CS) and *software agents*. The "agents" category can be further divided on the basis of whether the agents have the ability to migrate in a self-directed manner or not (i.e. whether the agents exhibit the characteristic of *mobility* or not). There is an emerging focus on efficiency and optimisation of response time in distributed data mining [6][9]. A significant issue in the success of an optimisation model is the computation of the cost of different factors in the DDM process. In this paper, we present mathematical cost models that facilitate identification of the different cost components in the DDM process for various strategies such as client-server and mobile agents. These cost models form the basis

· THE WORK REPORTED IN THIS PAPER HAS BEEN FUNDED IN PART BY THE CO-OPERATIVE RESEARCH CENTRE PROGRAM THROUGH THE DEPARTMENT OF INDUSTRY, SCIENCE AND TOURISM OF THE COMMONWEALTH GOVERNMENT OF AUSTRALIA.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 603–612, 2002.

for apriori estimation of the response time for a given task. In conjunction with the cost models, we have developed a technique for estimating the run time of data mining tasks (which is one of the cost components in the DDM process). The experimental evaluation of this technique establishes its estimation accuracy and validity.

In the following sections of this paper, we present cost models for distributed data mining including cost formulae for estimating the communication time and our technique for apriori estimation of the run time of data mining algorithms. The paper is organised as follows. In section 2 we review related work in the field of distributed data mining with a particular focus on optimisation techniques. Section 3 describes the cost components in different DDM techniques and presents the cost formulae for estimating these components. In section 4, the experimental results of our data mining task run time estimation technique are analysed. Finally, in section 5, we conclude by discussing the current status of our project and the future directions of this work.

2 Related Work

Work on improving performance of DDM systems by using an optimal/cost-efficient strategy has been the focus of [6][9]. IntelliMiner [6] is a client-server distributed data mining system, which focuses on scheduling tasks between distributed processors by computing the cost of executing the task on a given server and then selecting the server with the minimum cost. The cost is computed based on the resources (i.e. number of data sets) needed to perform the task. While this model takes into account the overhead of communication that is increased by having to transfer more datasets it ignores several other cost considerations in the DDM process such as processing cost and size of datasets. In [9], the optimisation is motivated by the consideration that mining a dataset either locally or by moving the entire dataset into a different server is a naïve approach. They use a linear programming approach to partition datasets and allocate the partitions to different servers. The allocation is based on the cost of processing (which is assigned manually in terms dollars) and the cost of transferring data (which is also assigned manually in terms of dollars). The manual assignment of cost requires expert knowledge about different factors that affect the DDM process and quantification of their respective impact, which is a non-trivial task. Thus the computation of the cost is an important question for any optimisation model. In the models discussed, the cost is either assigned manually or computed using a simple technique, which only takes into account the availability of datasets. In the following sections of this paper, we present a technique for computing the cost based on the response time for different mining strategies. We use apriori estimates of the response time for factors such as communication and processing.

3 Cost Components in the Distributed Data Mining Process

In general, optimisation models in DDM attempt to reduce the response time by choosing strategies that facilitate faster communication and/or processing. We

propose apriori estimates of the response time as the basis for computing the cost of distributed data mining. In this section we describe the different cost components of the DDM process and present cost formulae for estimating their response times. The response time of a DDM task broadly consists of three components:

1. *Communication:* The communication time is largely dependent on the DDM model and varies depending on whether the task is performed using a client server approach or using mobile agents
2. *Computation.* This is the time taken to perform data mining on the distributed data sets and is a core factor irrespective of the DDM model.
3. *Knowledge Integration.* This is the time taken to integrate the results from the distributed datasets.

The response time for distributed data mining is as follows:

$$T = t_{ddm} + t_{ki} \quad (1)$$

In (1), T is the response time, t_{ddm} is the time taken to perform mining in a distributed environment and t_{ki} is the time taken to perform knowledge integration. The response time for t_{ddm} in turn can be represented as:

$$t_{ddm} = t_{dm} + t_{com} \quad (2)$$

In (2), t_{dm} is the time taken to perform data mining and t_{com} is the time involved in the communication. Depending on the model used for distributed data mining (i.e. mobile agents or client server) and the different scenarios within each model, the factors such as communication that determine t_{ddm} will change. This results in a consequent change in the actual cost function that determines t_{ddm}. The modelling and estimation of the knowledge integration (t_{ki}) variable in Eq. (1) is non-trivial as it depends on several unknown factors such as the size and type of results produced by data mining. In fact, one of the primary reasons or rationales for data mining is the discovery of hidden and unknown patterns in the data. Thus, we do not present cost formulae or estimation techniques for the response time of knowledge integration.

3.1 Estimating the Communication Cost

The communication cost in the DDM process varies depending on whether the client-server strategy is followed or the mobile agent model is used.

3.1.1 Mobile Agent Model

This case is characterised by a given distributed data mining task being executed in its entirety using the mobile agent paradigm. The core steps involved include: submission of a task by a user, dispatching of mobile agent (or agents) to the respective data server (or servers), data mining and the return of mobile agent(s) from the data resource(s) with mining results. This model is characterised by a set of mobile agents traversing the relevant data servers to perform mining. In general, this can be expressed as m mobile agents traversing n data sources. There are three possible alternatives within this scenario. The first possibility is $m = n$, where the number of mobile agents is equal to the number of data servers. This implies that one data mining agent is sent to each data source involved in the distributed data mining task. The second option is $m < n$, where the number of mobile agents is less than the number of data servers. The implication of having fewer agents than servers is that

some agents may be required to traverse more than one server. We do not consider the third case of $m > n$ since this is in effect equivalent to the case 1 above where there is a mobile agent available per data server. Each of the above alternatives has its own cost function. These cost models are described as follows.

Case 1. Equal number of mobile agents and data servers (m=n).

This is a case where data mining from different distributed data servers is performed in parallel. The algorithm used across the different data servers can be uniform or varied. The system dispatches a mobile agent encapsulating the data mining algorithm (with the relevant parameters) to each of the data servers participating in the distributed data mining activity.

Let n be the number of data servers. Therefore, the number of mobile agents is n (since $m=n$). In order to derive the cost function for the general case involving n data servers and n data mining agents, we first formulate the cost function for the case where there is one data server and one data mining agent.

Let us consider the case where data mining has to be performed at the i^{th} data server (i.e $1 \leq i \leq n$). The cost function for the response time to perform distributed data mining involving the i^{th} data server is as follows:

$$t_{ddm} = t_{dm}(i) + t_{dmAgent}(AC, i) + t_{resultAgent}(i,AC)$$

The communication terms in the above cost estimate are are $t_{dmAgent}(AC, i)$ and $t_{resultAgent}(i,AC)$. That is,

$$t_{com} = t_{dmAgent}(AC, i) + t_{resultAgent}(i,AC) \quad (3)$$

$t_{dmAgent}(AC, i)$. In our cost model, the representation **$t_{mobileAgent}(x, y)$** refers to the time taken by the agent mobileAgent to travel from node x to node y. Therefore $t_{dmAgent}(AC, i)$ is the time taken by the mobile agent dmAgent (which is the agent encapsulating the mining algorithm and the relevant parameters) to travel from the agent centre (AC) to the data server (i). In general, the time taken for a mobile agent to travel depends on the following factors: the size of the agent and the bandwidth between nodes (e.g. in kilobits per second). The travel time is proportional to the size of the agent and is inversely proportional to the bandwidth. This can be expressed as follows:

$$t_{dmAgent}(AC, i) \propto \text{size of dmAgent} \quad (4)$$

$$t_{dmAgent}(AC, i) \propto 1 / \text{bandwidth} \quad (5)$$

From (4) and (5):

$$t_{dmAgent}(AC, i) = (k * \text{size of dmAgent}) / (\text{bandwidth between AC and i})$$

In the above expression for the time taken by the data mining agent to travel from the agent centre to the data server, k is a constant. On adapting the representation used by [8] to model the size of a mobile agent, we express the size of the data mining agent (dmAgent) as:

size of an dmAgent = <dmAgent state, data mining algorithm, input parameters>

$t_{resultAgent}(i, AC)$. This is the time taken for the data mining results to be transferred from the data server (i) to the agent centre (AC). However, estimating this component apriori is not feasible as the size of results obtained from a data mining task is usually unknown.

Since the mining is performed at the distributed locations concurrently, the total communication cost is equal to the time taken by the mobile agent that takes the longest time to travel to its respective remote location. Therefore,

$$t_{com} = \max(t_{dmAgent}(AC, i)), \text{ where } i = 1..n \quad (6)$$

Case 2. Fewer mobile agents than data servers (m<n).

This is the second case, where the number of mobile agents (m) available for distributed data mining is less than the number of data sources (n) participating in a distributed data mining task (i.e. $m < n$). Thus the i^{th} agent (where $1 \le i \le m$) travel to and perform mining at j sites labelled specified in ds_1, ds_2,…,ds_j (where $1 \le j \le n$). We make an assumption that the mining agent returns to the agent centre only after it has accomplished its task in all the respective data sources assigned to it and that results are sent to the central server directly. This allows us to estimate the communication cost more effectively.

The total time taken to mine is therefore the time taken by the agent, which takes the maximum time interval to complete its task. The communication cost estimate for the i^{th} agent's response time is as follows in equation (7):

$$t_{com}(i) = t_{dmAgent}(AC, ds_1) + \sum_{j=2}^{j-1} t_{dmAgent}(j, j+1) + t_{dmAgent}(ds_j, AC)$$

In the above expression, the first term is the time taken by the data mining agent to travel from the agent centre to the first data server in its path. The term involving the summation is the time taken for the agent to travel to the respective data sites within the set assigned to it (excluding the final site). The second last term is the time taken to mine at the last data site in its path. The final term in the expression is the time taken for the agent to travel from the last site on its path to the agent centre.

Since there are m agents operating concurrently, the time t_{com} is the time taken by the agent requiring the longest travel time. Thus,

$$t_{com} = \max(t_{com}(i)),\ i = 1..m \quad (8)$$

In (8), $t_{com}(i)$ is estimated from equation (7).

3.1.2 Client Server Model

The communication cost formulae for the response time in DDM systems that use the traditional client-server paradigm is presented in this section. Typically, data from distributed sources is brought to the data mining server – a fast, parallel server - and then mined. Let there be n data sites from which data has to be mined. Let s_i be the data set obtained from the i^{th} site (where $1 \le i \le n$). The communication time for DDM for the data set s_i from the i^{th} site is as expressed in equation (9) as follows:

$$t_{com}(i) = t_{dataTransfer}(i, DMS, s_i),\quad 1 \le i \le n \quad (9)$$

The term $t_{dataTransfer}(i, DMS, s_i)$ is the time taken to transfer the data set (s_i) from the i^{th} site to the DDM server (DMS) and is estimated as follows:

$$t_{dataTransfer}(i, DMS, s_i) = \text{size of } s_i \,/\, (\text{bandwidth between i and DMS}) \quad (10)$$

The data transfer can be a significant addition when the data volumes are large and/or the bandwidth is low. In this section, we have presented the cost formulae for the communication component of the DDM process for different mining strategies. We now present our technique for estimating the cost of performing data mining.

3.2 Estimating the Data Mining Cost

In this section, we present application run time estimation techniques as an approach to estimating the time taken to perform data mining. We have developed a novel rough sets based approach to estimating the run time of applications [5]. The motivation for our work in this area comes from application run time estimation techniques proposed by [7]. Application run time prediction algorithms including [1][2][7] operate on the principal that applications that have similar characteristics have similar run times. Thus, a history of applications that executed in the past along with their respective run times is maintained in order to estimate the task run time.

Early work in this area by [1][2] proposed the use of "similarity templates" of application characteristics to identify similar tasks in the history. A similarity template is a set of attributes that are used as the basis for comparing applications in order to determine if they are similar or not. It was proposed by [7] that manual selection of similarity templates was limited and they proposed automated definition and search for templates and used genetic algorithms and greedy search techniques. They were able to obtain improved prediction accuracy using these techniques. For a detailed description of the template identification and search algorithms readers are referred to [7].

We have developed a rough sets based algorithm to address the problem of automatic selection of characteristics that best define similarity to estimate application run times. Rough sets provide an intuitively appropriate theory for identifying good "similarity templates" (or sets of characteristics on the basis of which applications can be compared for similarity). For a detailed explanation of the theoretical soundness of a rough sets based approach and its improved prediction accuracy to identify similarity templates readers are referred to [5]. In this paper, we present a brief overview of our algorithm for use as a means to costing the data mining component in the DDM process.

3.2.1 Rough Sets Algorithm for Estimating t_{dm}

Zdislaw Pawlak introduced the theory of Rough Sets in 1981 as a mathematical tool to deal with uncertainty in data. For a good overview of rough sets concepts, readers are referred to [4]. A data set in rough sets is represented as a table called *Information System*, where each row is an object and each column is an attribute. The attributes are partitioned into condition attributes and decision attributes. The condition attributes determine the decision attribute. The history, as shown in figure 1, is a rough information system, where the objects are the previous applications whose run times (and other properties) have been recorded. The attributes in the information system are the properties about the applications that have been recorded. The decision attribute is the application run time that has been recorded. The other properties that have been recorded constitute the condition attributes. This model of a history intuitively facilitates reasoning about the recorded properties so as to identify the dependency between the recorded attributes and the run time. Thus, it is possible to concretise similarity in terms of the condition attributes that are relevant/significant in determining the decision attribute (i.e. the run time). Thus, the set of attributes that have a strong dependency relation with the run time can form a good similarity

template. Having cast the problem of application run time as a rough information system, we now examine the fundamental concepts that are applicable in determining the similarity template.

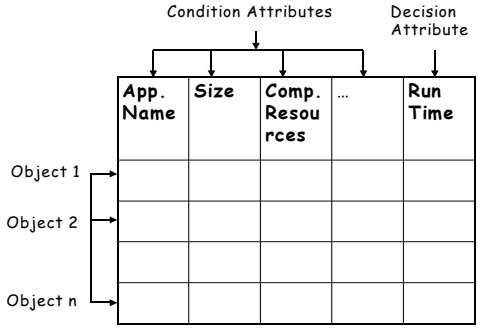

Figure 1: A Task History Modelled as a Rough Information System

Degree of Dependency. Using rough sets it is possible to measure the degree of dependency between two sets of attributes. The measure takes values [0,1] and higher values represent stronger degrees of dependency. It is evident that the problem of identifying a similarity template can be stated as identifying a set of condition attributes in the history that have a strong degree of dependency with the run time.
Significance of Attributes. The significance of an attribute is the extent by which the attribute alters the degree of dependency between a set of condition and decision attributes. If an attribute is "important" in discerning/determining the decision attribute, then its significance value, which is measured in the range [0,1], will be closer to 1. The similarity template should consist of a set of properties that are important for determining the run time.
Reduct. A reduct consists of the minimal set of condition attributes that have the same discerning power as the entire information system. All superfluous attributes are eliminated from a reduct. According to [4], while it is relatively simple to compute a single reduct, the general solution for finding all reducts is NP-hard. A similarity template should consist of the most important set of attributes that determine the run time without any superfluous attributes. In other words, the similarity template is equivalent to a reduct which has the most significant attributes included.

We now present our rough sets algorithm for identifying similarity templates in figure 2. It is evident that rough sets theory has highly suitable and appropriate constructs for identifying the properties that best define similarity for estimating application run time estimation. Our technique for applying rough sets to identify similarity templates centres round the concept of a reduct. We use a variation of the reduct generation algorithm proposed by [3]. The algorithm proposed by [3] was intended to produce reducts that included user specified attributes. The modified algorithm we use to compute the reduct for use as a similarity template is shown in figure 2. It computes reducts by iteratively adding the most significant attribute to the D-core (note: the *core* of a rough information system is the intersection of all reducts and the *D-core* is the core with respect to the set of decision attributes). An

improvement we have included in the algorithm is the identification of any reduct of size |D-core + 1|. From our experiments we found that it is not unusual for the D-core to combine with the a single attribute (typically the most significant attribute) to form a reduct. This iteration in Step 6, is computationally inexpensive as it involves only simple additions and no analysis. Thus, if a reduct of size |D-core + 1| exists, we find it without further computation.

```
1.  Let A={a1, a2,….,an}be the set of condition attributes and D be the set of decision
    attributes.
2.  Let C be the D-Core
3.  REDUCT = C
4.  A1 = A – REDUCT
5.  Compute the Significances of the Attributes (SGF) in A1 and sort them in ascending
    order
6.  For i = |A1| to 0
        K(REDUCT, D) =  K(REDUCT, D) + SGF(ai) /* K(X,Y) is the degree of
        dependency between attribute sets X and Y) */
        If K(REDUCT,D) = K(A,D)
            REDUCT = REDUCT ∪ ai
            Exit
        End If
      K(REDUCT, D) =  K(REDUCT, D) - SGF(ai)
    End For
7.  K(REDUCT, D) =  K(REDUCT, D) + SGF(a|A1|)
8.  While K(REDUCT,D) is not equal to K(A,D)
    Do
        REDUCT = REDUCT ∪ ai (where SGF(ai) is the highest of the attributes in
        A1)
        A1 = A1 - ai
        Compute the degree of dependency K(REDUCT,D)
    End
9.  |REDUCT| -> N
10. For i = 0 to N
        If ai is not in C (that is the original set of attributes of the REDUCT at the start
        and SGF(ai) is the least)
            Remove ai from REDUCT
        End If
        Compute degree of dependency K(REDUCT, D)
        If  K(REDUCT, D) not equal to K(A,D)
            REDUCT ∪ ai -> REDUCT
        End If
11. End For
12. End if
```

Figure 2: Reduct Generating Algorithm

This process of re-computation of the dependency (to determine if an attribute is significant or not) is expensive as it involves finding equivalence classes and the iteration we initiated was principally an attempt to avoid it. We have also introduced the concept of "incremental evaluation of equivalence classes" to improve the performance of the algorithm. We have presented our rough sets approach to application run time estimation techniques as a way of estimating the cost of processing as it allows prediction of the time taken to perform mining at a given location. Obviously, the success of such a technique is dependent on the prediction accuracy.

4 Experimental Results and Analysis

The viability of using these cost models for DDM optimisation depends on the accuracy of the estimation techniques. Thus, the estimated communication times and predicted data mining task run times must be close to actual run time for these cost formulae to form the basis for effective optimisation. We are implementing mobile agents to perform distributed data mining and do not have experimental results to validate the communication cost models. However, we have implemented our rough sets algorithm and in this section present results from experiments on estimating the run times of data mining tasks. We compiled a history of data mining tasks by running several data mining algorithms on a network of distributed machines running Windows 2000 and Sun OS 5.8 and recording information about the tasks and the environment. We executed several runs of data mining jobs by varying the parameters of the jobs such as the mining algorithm, the data sets, the sizes of the data sets (from 1MB to 20MB) and the machines on which the tasks were run. The algorithms used were from the WEKA package of data mining algorithms [10]. For each data mining job, the following information was recorded in the history: the algorithm, the file name, the file size, the operating system, the version of the operating system, the IP address of the local host on which the job was run, the processor speed, the memory, the start and end times of the job. Currently we record only static information about the machines, however we are implementing a feature to enable recording dynamic information such as memory usage and CPU usage. We used histories with 100 and 150 records and each experimental run consisted of 20 tests. The mean error we recorded was 0.34 minutes and the mean error as a percentage of the mean run time was 8.29. The mean error is less than a minute and the error as a percentage of the actual run times is also very low, which indicates that we obtained very good estimation accuracy for data mining tasks. This good performance accuracy is illustrated in figure 3, which presents the actual and estimated run times from one of our experimental runs.

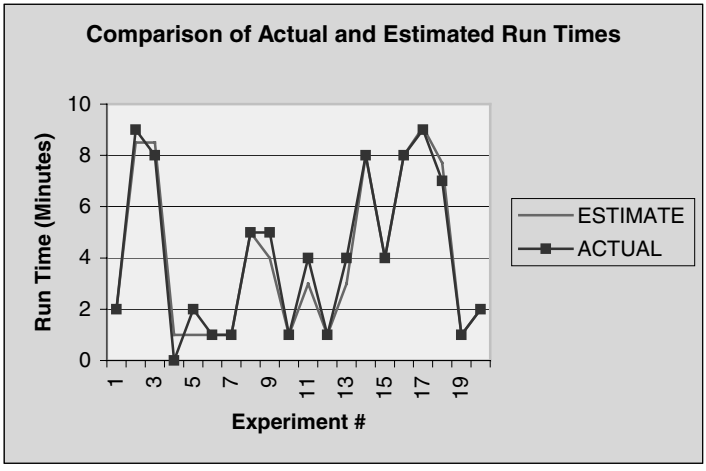

Figure 3: Actual Vs. Estimated Run Times

5 Conclusions and Future Work

This paper has focussed on computing the cost of the distributed data mining process by estimating the response time of the communication and computation components. Accurate costing is vital for optimisation of the DDM process and we address the need for good cost estimation techniques. We have developed cost formulae for estimating the response time of the communication for different strategies (including client server and mobile agent techniques) and a rough sets based application run estimation algorithm for predicting the run times of data mining tasks. We have experimentally validated the prediction accuracy of our rough sets algorithm, which is shown to have low mean errors and high accuracy. We are currently implementing the mobile agent model for DDM so as to experimentally validate the cost formulae for estimating the communication time. Future directions of this work include the development of a DDM optimiser based on the cost models presented in this paper.

References

1. Downey,A,B., (1997), "Predicting Queue Times on Space-Sharing Parallel Computers", *Proc. of the 11th Intl. Parallel Processing Symposium (IPPS)*, Geneva, Switzerland, April.
2. Gibbons,R., (1997), "A Historical Application Profiler for Use by Parallel Schedulers", *LNCS* 1291, Springer-Verlag, pp.58-75.
3. Hu,X., (1995), "Knowledge Discovery in Databases: An Attribute-Oriented Rough sets Approach", PhD Thesis, University of Regina, Canada.
4. Komorowski,J., Pawlak,Z., Polkowski,L., and Skowron, A., (1998), "Rough sets: A Tutorial", in *Rough-Fuzzy Hybridization: A New Trend in Decision Making*, (eds) S.K.Pal and A.Skowron, Springer-Verlag, pp. 3-98.
5. Krishnaswamy,S., Loke,S,W., & Zaslavsky,A, (2002), "Application Run Time Estimation: A Quality of Service Metric for Web-based Data Mining Services", *To Appear in ACM Symposium on Applied Computing (SAC 2002),* Madrid, March.
6. Parthasarathy,S., and Subramonian,R., (2001), "*An Interactive Resource-Aware Framework for Distributed Data Mining",* in Newsletter of the IEEE Technical Committee on Distributed Processing, Spring 2001, pp.24-32.
7. Smith,W., Taylor,V., and Foster,I.,(1999), *"Using run-time predictions to estimate queue wait times and improve scheduler performance"*, *LNCS* 1659, Springer-Verlag, pp.202-229.
8. Straßer,M., and Schwehm,M., (1997), "A Performance Model for Mobile Agent Systems", in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, (eds) H. Arabnia, Vol II, CSREA, pp. 1132-1140.
9. Turinsky,A., and Grossman,R., (2000), "A Framework for Finding Distributed Data Mining Strategies that are Intermediate between centralized Strategies and In-place Strategies", *Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000,* Boston, pp.1-7.
10. Witten,I,H., and Eibe,F., (1999), "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Morgan Kauffman.