Available online at www.worldscientificnews.com

World Scientific News

An International Scientific Journal

WSN 41 (2016) 247-252 EISSN 2392-2192

High Speed Data Streams Using Data Mining Techniques

R. Sangeetha

Department of Information and Computer Technology, Hindusthan College of Arts and Science, Coimbatore, Tamil Nadu, India

E-mail address: sangeethahicasit@yahoo.com

ABSTRACT

Many organizations today have more than very large data-bases; they have databases that grow without limit at a rate of several million records per day. Mining these continuous data streams brings unique opportunities, but also new challenges. This paper describes and evaluates VFDT, an anytime system that builds decision trees using constant memory and constant time per example. VFDT can in-corporate tens of thousands of examples per second using the-shelf hardware. It uses Hoeffding bounds to guar-antee that its output is asymptotically nearly identical to that of a conventional learner. The study VFDT's proper-ties and demonstrate its utility through an extensive set of experiments on synthetic data. To apply VFDT to mining the continuous stream of Web access data from the whole University of Washington main campus.

***Keywords*:** data-bases, VFDT, University of Washington

1. INTRODUCTION

Knowledge discovery systems are constrained by three main limited resources: time, memory and sample size. In traditional applications of machine learning and statistics, sample size tends to be the dominant limitation: the computational resources for a massive search are available, but carrying out such a search over the small samples available (typically less than

(Received 01 February 2016; Accepted 10 February 2016; Date of Publication 11 February 2016)

10,000 examples) often leads to over fitting or \data dredging" (e.g.,). Thus over fitting avoidance be-comes the main concern, and only a fraction of the available computational power is used [3]. In contrast, in many (if not most) present-day data mining applications, the bottleneck is time and memory, not examples. The latter are typically in over-supply, in the sense that it is impossible with cur-rent KDD systems to make use of all of them within the available computational resources. As a result, most of the available examples go unused, and under fitting may result: enough data to model very complex phenomena is available, but inappropriately simple models are produced because unable to take full advantage of the data. Thus the development of highly efficient algorithms becomes a priority.Currently, the most efficient algorithms available (e.g.,) concentrate on making it possible to mine databases that do not _t in main memory by only requiring sequential scans of the disk. But even these algorithms have only been tested on up to a few million examples. In many applications thesis less than a day's worth of data.

The expansion of the Internet continues and ubiquitous computing becomes a reality, it can expect that such data volumes will become the rule rather than the exception. Current data mining systems are not equipped to cope with them. When new examples arrive at a higher rate than they can be mined, the quantity of unused data grows without bounds as time progresses. Even simply preserving the examples for future use can be a problem when they need to be sent to tertiary storage, are easily lost or corrupted, or become unusable when the relevant contextual information is no longer available. When the source of examples is an open-ended data stream, the notion of mining a database of fixed size itself becomes questionable.

Ideally, it would like to have KDD systems that operate continuously and indefinitely, incorporating examples as they arrive, and never losing potentially valuable information. Such desiderata are fulfilled by incremental learning methods (also known as online, successive or sequential methods), on which a substantial literature exists. However, the available algorithms of this type (e.g.,) have significant shortcomings from the KDD point of view.

Some are reasonably efficient, but do not guarantee that the model learned will be similar to the one obtained by learning on the same data in batch mode. They are highly sensitive to example ordering, potentially never recovering from an unfavorable set of early examples. Others produce the same model as the batch version, but at a high cost in efficiency, often to the point of being slower than the batch algorithm.This paper proposes Hoeffding trees, a decision-tree learning method that overcomes this trade-o_. Hoeffding trees can be learned in constant time per example (more precisely, in time that is worst-case proportional to the number of attributes), while being nearly identical to the trees a conventional batch learner would produce, given enough examples. The probability that the Hoeffding and conventional tree learners will choose different tests at any given node decreases exponentially with the number of examples. Also describe and evaluate VFDT, a decision-tree learning system based on Hoeffding trees.

VFDT is I/O bound in the sense that it mines examples in less time than it takes to input them from disk. It does not store any examples (or parts thereof) in main memory, requiring only space proportional to the size of the tree and associated sufficient statistics. It can learn by seeing each example only once, and therefore does not require examples from an online stream to ever be stored. It is an anytime algorithm in the sense that a ready-to-use model is available at any time after the first few examples are seen, and its quality increases smoothly with time. The next section introduces Hoeffding trees and studies their properties. To describe the VFDT

system and its empirical evaluation. The paper concludes with a discussion of related and future work.

2. HOEFFDING TREES

The classification problem is generally defined as follows. A set of N training examples of the form (x; y) is given, where y is a discrete class label and x is a vector of d attributes, each of which may be symbolic or numeric. The goal is to produce from these examples a model y = f(x) that will predict the classes y of future examples x with high accuracy.

For example, x could be a description of a client's recent purchases, and y the decision to send that customer a catalog or not; or x could be a record of a cellular-telephone call, and y the decision whether it is fraudulent or not. One of the most effective and widely-used classification methods is decision tree learning [1]. Learners of this type induce models in the form of decision trees, where each node contains a test on an attribute, each branch from a node corresponds to a possible outcome of the test, and each leaf contains a class prediction. The label y = DT(x) for an example x is obtained by passing the example down from the root to a leaf, testing the appropriate attribute at each node and following the branch corresponding to the attribute's value in the example. A decision tree is learned by recursively replacing leaves by test nodes, starting at the root.

The attribute to test at a node is chosen by comparing all the available attributes and choosing the best one according to some heuristic measure. Classic decision tree learners like ID3, C4.5 and CART assume that all training examples can be stored simultaneously in main memory, and are thus severely limited in the number of examples they can learn from. Disk-based decision tree learners like SLIQ [10] and SPRINT assume the examples are stored on disk, and learn by repeatedly reading them in sequentially (effectively once per level in the tree). While this greatly increases the size of usable training sets, it can become prohibitively expensive when learning complex trees (i.e., trees with many levels), and fails when datasets are too large to _t in the available disk space. The goal is to design a decision tree learner for extremely large (potentially infinite) datasets. This learner should require each example to be read at most once, and only a small constant time to process it. This will make it possible to directly mine online data sources (i.e., without ever storing the examples), and to build potentially very complex trees with acceptable computational cost. To achieve this by noting with Catlett [2] and others that, in order to find the best attribute to test at a given node, it may be sufficient to consider only a small subset of the training examples that pass through that node. Thus, given a stream of examples, the _first ones will be used to choose the root test; once the root attribute is chosen, the succeeding examples will be passed down to the corresponding leaves and used to choose the appropriate attributes there, and so on recursively. The difficult problem of deciding exactly how many examples are necessary at each node by using a statistical result known as the Hoeffding bound (or additive Cherno_ bound).

Consider a real-valued random variable r whose range is R (e.g., for a probability the range is one, and for an information gain the range is log c, where c is the number of classes). Suppose we have made n independent observations of this variable, and computed their meanr. The Hoeffding bound has the very attractive property that it is independent of the probability distribution generating the observations. The price of this generality is that the bound is more

conservative than distribution-dependent ones will take more observations to reach the same _ and _).

Let G(Xi) be the heuristic measure used to choose test attributes (e.g., the measure could be information gain as in C4.5, or the Gini index as in CART). Our goal is to ensure that, with high probability, the attribute chosen using n examples (where n is as small as possible) is the same that would be chosen using infinite examples. Assume G is to be maximized, and let Xa be the attribute with highest observed G after seeing n examples, and Xb be the Ideally, it would like to have KDD systems that operate continuously and indefinitely, incorporating examples as they arrive, and never losing potentially valuable information.

Such desiderata are fulfilled by incremental learning methods (also known as online, successive or sequential methods), on which a substantial literature exists. However, the available algorithms of this type (e.g.,) have significant shortcomings from the KDD point of view. Some are reasonably efficient, but do not guarantee that the model learned will be similar to the one obtained by learning on the same data in batch mode.

3. THE VFDT SYSTEM

The VFDT System implemented a decision-tree learning system based on the Hoeffding tree algorithm, which call VFDT (Very Fast Decision Tree learner). VFDT allows the use of either information gain or the Gini index as the attribute evaluation measure. It includes a number of refinements.

Category 1: Ties. When two or more attributes have very similar G's, potentially many examples will be required to decide between them with high confidence.

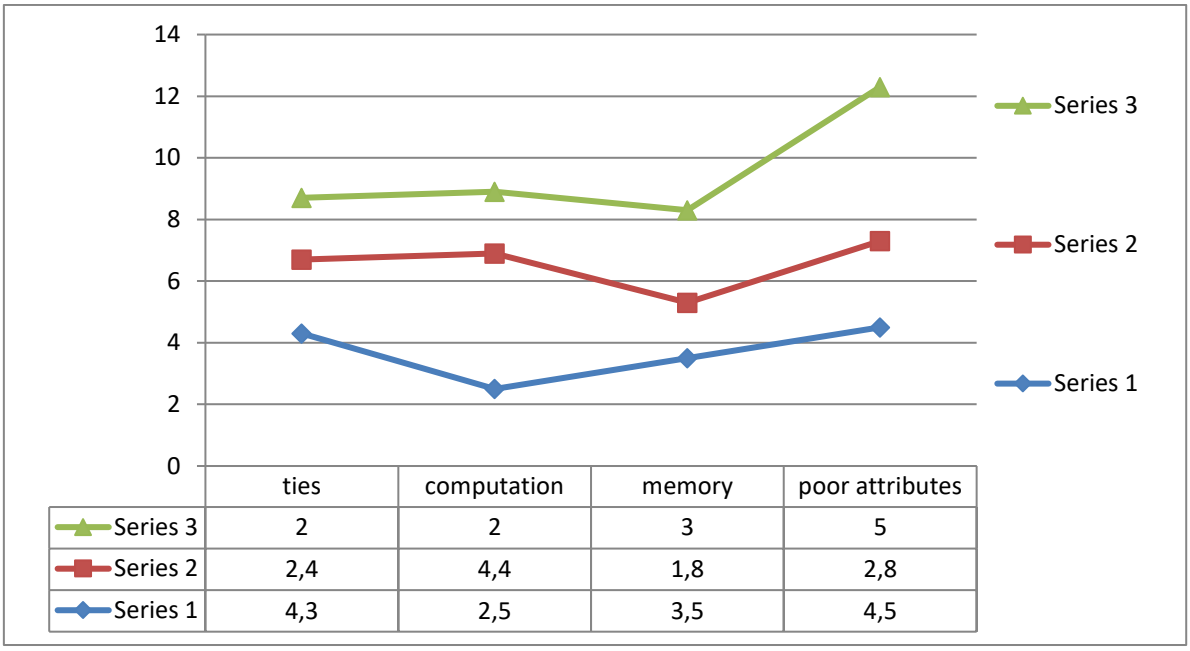
Category 2: G computation. The most significant part of the time cost per example is recomputing G. It is inefficient to recomputed G for every new example, because it is unlikely that the decision to split will be made at that specific point.

Category 3: Memory. As long as VFDT processes examples faster than they arrive, which will be the case in all but the most demanding applications, the sole obstacle to learning arbitrarily complex models will be the finite RAM available.

Category 4: Poor attributes. Memory usage is also minimized by dropping early on attributes that do not look promising. Initialization VFDT can be initialized with the tree produced by a conventional RAM-based learner on a small subset of the data.

4. VFDT TREE STRUCTURE

A system like VFDT is only useful if it is able to learn more accurate trees than a conventional system, given similar computational resources.



	ties	computation	memory	poor attributes
Series 3	2	2	3	5
Series 2	2,4	4,4	1,8	2,8
Series 1	4,3	2,5	3,5	4,5

In particular, it should be able to use to advantage the examples that are beyond a conventional system's ability to process. Figure 1 shows the accuracy of the learners averaged over all the runs. VFDT was run, no leaf reactivation, and no rescans. VFDT boot is VFDT bootstrapped with an over-pruned version of the tree produced by C4.5. C4.5 is more accurate than VFDT up to 25k examples, and the accuracies of the two systems are similar in the range from 25k to 100k.

5. WEB DATA

Currently applying VFDT to mining the stream of Web page requests emanating from the whole University of Washington main campus. The nature of the data is described in detail in. In our experiments so far it’s have used a one-week anonymized trace of all the external web accesses made from the university campus. There were 23,000 active clients during this one-week trace period, and the entire university population is estimated at 50,000 people (students, faculty and sta_). The trace contains 82.8 million requests, which arrive at a peak rate of 17,400 perminute. The size of the compressed traceable is about 20GB.

6. RELATED WORK

This work on mining large databases using sub sampling methods includes the following. Catlett [2] proposed several heuristic methods for extending RAM-based batch decision-tree learners to datasets with up to hundreds of thousands of examples. Musick, Catlett and Russell proposed and tested (but did not implement in a learner) a theoretical model for choosing the size of subsamples to use in comparing attributes.

7. CONCLUSION

This paper introduced Hoeffding trees, a method for learning online from the high-volume data streams that are increasingly common. Hoeffding trees allow learning in very small constant time per example, and have strong guarantees of high asymptotic similarity to the corresponding batch trees. VFDT is a high-performance data mining system based on Hoeffding trees. Empirical studies show its effectiveness in taking advantage of massive numbers of examples. Application to a high-speed stream of Web log data is under way.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J.Stone. Classication and Regression Trees.Wadsworth, Belmont, CA, 1984.

[2] J. Catlett. Megainduction: Machine Learning on Very Large Databases. PhD thesis, Basser Department of Computer Science, University of Sydney, Sydney, Australia, 1991.

[3] T. G. Dietterich. Over-tting and undercomputing in machine learning. *Computing Surveys*, 27: 326{327}, 1995

[4] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and. Xu, Incremental clustering for mining in a datawarehousing environment. In *Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases*, pages 323{333}, New York, NY, 1998.