Optimization Driven Data Mining and Credit Scoring

Robert L. Grossman* H. Vincent Poor†

1 Introduction

Assume we are given a large collection of objects, each with several hundred attributes, and we wish to assign scores and take appropriate actions for each object in such a way as to maximize a given objective function defined on the entire collection. In this paper, we describe a methodology that uses data mining to divide the objects into different clusters on the basis of their attributes so that individualized scores and appropriate actions can be assigned to the objects in each cluster.

This problem arises in a variety of applications: For example, we may wish to assign a credit score to a credit card prospect indicating the likelihood that the prospect will make credit card payments and then to set a credit limit for each prospect in such a way as to maximize the over-all expected revenue from the entire collection of prospects. In the terminology we will use in this paper, the credit score is called a statistical attribute and the credit limit a control attribute. The methodology we describe below uses data mining to better estimate the statistical attributes and to better set the control attributes.

As another example, consider the real-time detection of fraudulent credit card transactions. In this example, the goal is to assign a risk score to a real-time stream of credit card transactions indicating the probability that the transaction is fraudulent and to set a control attribute indicating a subsequent action to take, if any. The actions could include, for example, approval, denial, secondary automatic testing, or secondary testing by a human. The goal is to assign the control attributes so that the over-all loss of a collection of transactions is minimal, subject to a given level of resources.

As a non-financial application, consider a stream of documents that need to be sorted automatically, and, in addition, those which are most critical need to be assigned to individuals to read in detail. The goal in this example is to assign automatically a relevance score indicating that the document's relevance to a specified list of topics. In addition, a control attribute would be set indicating subsequent action to take. Actions could include archiving, dissemination, or forwarding to a specific individual. The goal is to set the control attributes so as to maximize information subject to a given level of resources.

We are interested in algorithms which scale 1) as the number of objects grows and 2) as the number of attributes for each object grows. To be specific, we have started early testing of algorithms and implementations for millions of objects and hundreds of attributes.

Broadly speaking the methodology we are proposing involves the following steps:

1. We assume that we are given a collection of objects from which we can learn. In an off-line computation we work with this collection and divide it into a number of sub-

*Magnify, Inc., 815 Garfield St., Oak Park,IL 60304 USA, 708 383 7002, 708 383 7084 fax,rlg@magnify.com. The author is also a faculty member at the University of Illinois at Chicago.

†Electrical Engineering Department, Princeton University, Princeton NJ 08544 USA,.

collections on the basis of the objects' attributes. This may be thought of a classification of the objects into different clusters on the basis of the data of the problem, that is on the basis of the data attributes.

2. For each sub-collection or cluster in the learning collection, we separately estimate certain random variables of the data attributes, which we call statistical attributes. We also estimate certain other parameters which we use in setting the control attributes in the step below.

3. In this step, we are given a different collection with the goal of estimating the statistical attributes of this collection and of setting the control attributes. We use the classification obtained in the Step 2 to estimate the statistical attributes of the objects in this collection. In addition we set the control attributes for the objects in the collection to optimize an objective function which is defined on the original collection as a whole.

Underlying our work our three key ideas:

First, we are interested in a class of problems with an over-all structure which we can exploit that broadly corresponds to dividing the attributes into data attributes which we are given, statistical attributes which we estimate, and control attributes which we set.

Second, our point of view is that *good data mining requires good data management.* Specifically, our data mining algorithms exploit parallelism and use an object warehouse designed for data mining applications that is capable of managing terabytes of data [9]. With this approach, sampling is not required, and it is in this sense that we refer to our approach as *data-driven.* A more traditional approach would sample the original data and use data mining on the sampled data.

Third, data mining has generally been used to look for patterns, associations, or anomalies in the data. We have also found these techniques useful for automatically dividing the data into different clusters and assigned different models to each cluster. With sampled data it is difficult to build a large number of different models since after sampling there is not enough data to build the separate models. This is one of the key ideas which allows us to set the control attributes appropriately.

This paper summarizes preliminary work. A full paper providing additional background, a more detailed description of the algorithm and experimental results is in preparation. The methodology described here is used in the PATTERN software system developed by Magnify, Inc.

2 Data Structure

Logically, we view the data as consisting of one or more collections of objects and their attributes. Physically, the object warehouse can manage the data, even if the data is distributed over different nodes and different physical media, such as disk and tape.

For example, in our running example there are several collections of objects, including a collection of customers and a collection of transactions. It is convenient to divide the attributes into several types:

Data attributes. Each customer and transaction has a variety of data attributes. By definition, data attributes are basic attributes, such as the name of a customer, the name of the merchant, or the amount of the transaction.

Summary attributes. These are derived attributes computed from the data attributes. Examples include the number of transactions for a customer during the past six months, the average amount of the transactions over the same period, the highest value of any single transaction during the period, etc. Thought of from a different point of view, summary attributes aggregate the underlying data attributes.

Statistical attributes. Not only is it useful to summarize the data attributes, but it is also useful to use the data

attributes to make predictions. Statistical attributes are random variables which are functions of the data attributes. They may be estimated using a variety of statistical methods, such as regression trees. Examples include the expected number of transactions during a given period, the likelihood that a customer will purchase an item significantly more expensive than any one that has been purchased during the past three months and the probability that a customer will purchase an item of a given type.

Control attributes. These are attributes which we set. An example might be the amount of credit offered a customer or a special customer plan provided to a customer.

To summarize, an object such as a customer has a variety of attributes attached to it. There are two basic types of attributes derived from the basic data attributes associated with an object: summary attributes which reflect or aggregate past behavior and statistical attributes which are random variables and which can be used to predict future behavior.

3 Tree-Based Stochastic Optimization

An optimization tree (OT) is broadly analogous to a classification or regression tree [8]. In each case, we start with all of the data assigned to the root and assign the data to children of the root by selecting an attribute and a threshold. Those objects in which the value of the specified attribute is less than the threshold are assigned to the left node; the remainder to the right node. The tree is then grown recursively. Nodes with more than two children can can easily be handled by using more than one threshold.

Classification, regression and optimization trees use different rules for growing and interpreting the tree. In an optimization tree, the control attributes are assigned differently for the objects assigned to each leaf node. The advantage of this approach is that large heterogeneous collections of objects typically can be divided using statistical methods into smaller more homogeneous collections in such a way that the control attributes can be set more advantageously for each sub-collection.

Recall that we assume that the attributes of the objects are divided into several types: data attributes, statistical attributes, and control attributes. We assume that we are given an objective function which is a function of the data attributes, summary attributes, statistical attributes, and control attributes. Our approach involves the following steps:

1. The first step is to use a training set of objects and build one or more regression trees to estimate the required statistical attributes in terms of the underlying data and summary attributes.

2. The next step is to a build tree to estimate the objective function in terms of the other attributes, including the statistical attributes. This is the optimization tree.

3. The final step is to use an iterative algorithm to set the control attributes for each object in order to maximize the objective function. This is done separately for each leaf of the optimization tree.

We now describe computing the statistical and control attributes in more detail. The first step is to use standard techniques from [3] to build a regression tree that allows us to estimate the statistical attributes from the data attributes and summary attributes.

In the second step, we are given the data and summary attributes and estimates of the statistical attributes. We are also given a training set of objects which contain all of the attributes including values of the control attributes and the corresponding values of the objective function. Our goal is to estimate the performance attributes and other relevant attributes. The nature of these relevant parameters will be discussed below.

In the third step, we are given the data, summary, and statistical attributes, but not the con-

trol attributes. Our goal is to choose control attributes that optimize the expected performance that will result from the application of those controls to the given case. Since we have from Step 2 a tree that estimates performance (objective function) from a full set of attributes (i.e., observed attributes plus control attributes), an obvious, but totally impractical, way of choosing the control attributes is to search exhaustively through the set of possible of control values for those that lead to a maximum. Clearly some other search strategy is needed, and for computational efficiency some type of gradient ascent is desirable.

Since the performance surface has been estimated in Step 2, nonparametric estimates of gradient (i.e., numerical derivatives) are suspect. Thus, we consider instead a parametric approach, in which a local model for the performance surface is assumed from the phenomenology of the application. This is a reasonable approach for credit scoring, risk-management and related financial problems, since local effects of control variables are often known. By locally parametrizing the surface, the gradient with respect to control attributes can be computed at a given point in the attribute space if the local surface parameters can be estimated at that point. These are the aforementioned parameters that the regression tree must be trained to estimate. Given a tree trained in this way, an algorithm for choosing optimizing controls for a given case is clear. Namely, an initial set of control attributes is selected, and the regression tree is used to estimate the performance and surface parameters corresponding to the combined set of observed attributes and selected control attributes. The estimated surface parameters then determine the gradient vector at that point in attribute space The regression tree thus presents us with a search direction in which to move toward a higher revenue-producing line and rate, and we can iterate in this manner until we find a desirable point in the control space.

This combination of parametric gradient description with regression tree estimation allows for effective stochastic optimization of control variables in a much more efficient manner than could otherwise be accomplished.

4 Distributed CART

The general philosophy proposed in the sections above calls for comprehensive mining of very large databases as an alternative to more traditional techniques based on statistical sampling. In order to apply the algorithms described in Section 3 for the examples considered in the introduction, it is necessary to develop methods for implementing classification and regression trees on very large databases. Of course, the key issue here is that of *growing* a classification or regression tree on a large training database, since the actual application of an established tree requires relatively minor computational load. In this section, we describe a method for distributing the growth of a classification or regression tree onto a cluster of workstations.

Our methodology is based on ideas developed for the problem of distributed detection/classification (e.g., [1, 14, 15]), which is concerned primarily with the optimization of classification algorithms in which the observations are distributed among a set of processors that are connected to a central processor via communication-constrained channels. Typically, the distributed processors communicate only with the central processor, which is known as a *fusion center.* The fusion center is controlling, in the sense that it takes the decisions (i.e., classifications) of the distributed processors and combines them into a single, final global decision.

Distributed classification can be applied to the development of a distributed CART algorithm by adopting a fusion feedback mechanism at each level of tree growth. In particular, the basic idea is to use a cluster of workstations to grow a single classification or regression tree, while each workstation works on only part of the data. Thus, a large database (i.e., a large sample of objects) is distributed uniformly among a group of processors (e.g., workstations), one of which serves also as a fusion center. Each processor grows classification and regression trees in a standard manner (e.g., [3]) to determine the first branch in a classification or regression tree for its portion of the

database. (The processors share a common objective and algorithm for local tree growth.)

When each processor has determined the attribute on which it should branch first, this information is sent to the fusion center. (The two-class version of this problem is related to a distributed sequential detection problem of [13, 17] - see also [4].) The role of the fusion center at this point is to combine the first-branch information from the distributed processors into a global decision as to which variable should be used in the first branch of the global tree, and what threshold should be applied to that variable.

A variety of fusion rules can be applied to make these decisions. Assuming that the data contains a sufficient degree of substantialism (i.e., that the data is truly informative for the classification/regression problem at hand), there should be some agreement among the processors regarding the first-branch variable. Thus, a simple plurality vote serves as a reasonable fusion rule for determining an appropriate variable on which to begin branching. (In this scheme, randomization is necessary if there are ties.) Alternatively, the processors can compute a confidence score for their first-branch decisions, and these scores can then be aggregated to form a global decision.[1]

Soft decisions can also be made if no attribute is a clear winner for the first branch, although this increases the complexity of the tree-growing process. Having thus determined the attribute to use for the first branch of the global tree, the global threshold can be determined by simply averaging the thresholds of those workstations that have selected the winning attribute, or by computing a score-weighted average.

Once the fusion center has determined the first branch of the global tree, each processor then divides its data into two classes based on this global first branch. The same procedure of computing, then fusing, distributed branching procedures is then repeated for each branch emanating from this first branch. By recursively applying this procedure, a global tree is grown. Pruning, when necessary, proceeds in the same manner - that is, by fusing distributed pruning decisions into global pruning decisions.

It is noteworthy that, while the procedure outlined above draws its inspiration from the distributed detection paradigm, there are some key differences between distributed detection and distributed classification and regression trees. An obvious and overriding difference is that distributed detection problems typically involve two or, perhaps, a few classes among which the detection network is trying to choose, whereas the distributed classification and regression network is trying to choose a tree structure from among a very large number of possible tree structures. That is, the dimensionality of the decision space is much, much larger in distributed CART than in more standard distributed classification problems; and so the goals of the tree-growth algorithm must be much more heuristic than those applied in problems with smaller dimensional decision spaces. (Of course, this high-dimensionality is also a feature of centralized CART, so this comment applies more generally.) Moreover, this also means that the amount of data available at each processor still must be quite significant, since there must be enough substantialism for the distributed sensors to arrive at some form of consensus at each level of the tree.

A further distinction between distributed CART and distributed detection is the recursive nature of the former. Although there are some distributed detection algorithms that involve feedback from the fusion center to the distributed processors (e.g., [16]), this is not a typical feature of such systems. That is, in most distributed detection algorithms, the distributed processors make their decision independently of one another, and these are combined by the fusion center to make a global decision. The presence of feedback in distributed CART strongly correlates the computations in the distributed processors, and makes analytical characterization of this process very difficult. Thus, the pri-

[1] Decision trees have also been used to implement fusion rules in distributed classification networks (cf. [5]); so tree-based techniques might also be applied in the fusion center. However, it would seem that the amount of data at this level would be too small to make this a practical alternative.

mary mode of performance evaluation is simulation.

5 Object Warehouses

By data-driven we mean that we exploit all of the data without having to sample. This requires both that our algorithms scale and that our data management system scale. One of the key ideas behind scaling our algorithms is to grow our classification, regression and optimization trees in a distributed fashion as we have described in the section above.

The key idea required to scale the data management is to exploit a data management system which is explicitly designed for data intensive computing, that is for making numerically and statistically intensive queries on large data sets [7].

Traditional relational databases are designed to support simple data types and simple operations on the data. Object-oriented databases are designed to support complex objects but with the access patterns typical of databases: frequent additions and updates. On the other hand, an object warehouse [9] is designed for numerically and statistically intensive queries on large, distributed collections of objects.

To achieve this, an object warehouse is optimized for frequent reads, occasional appends, and infrequent updates. In addition, object warehouses must support very large collections of objects. To do this, some type of physical hierarchy of objects must be used. For example, a standard object-oriented database manages objects and physical collections of objects called segments which are moved as necessary between disk and memory. There are simply too many segments in a terabyte of objects to manage efficiently. For this reason, in the system we used for this study [9], segments are gathered into larger collections called folios which are separately managed. In other words, there are separate object, segment, and folio managers in order to provide the scalability required for working with terabytes of objects.

6 Summary

In this paper we have considered a class of problems involving estimating statistical attributes and setting control attributes for large collections of objects on the basis of the objects' attributes. These types of problems are difficult because of the large numbers of objects and the large numbers of attributes for each object.

Classification and regression trees are standard techniques used in data mining to deal with collections of objects in which the objects have large numbers of attributes. We propose using optimization trees as a tool for setting control attributes for collections of objects so as to optimize a specified objective function defined on the entire collection.

We briefly describe how to compute distributed classification, regression and optimization trees. We also briefly comment on some of data management issues which arise for these types of problems in practice. We propose using object warehouses to provide low overhead, high performance access to large collections of objects as an underlying foundation for our data mining algorithms.

The work described in this paper is preliminary. A paper providing additional background, more details, and experimental results is in preparation. The algorithms described here are implemented by the PATTERN system developed by Magnify, Inc.

References

[1] R. Ahlswede and I. Csiszar, "Hypothesis testing with communication contraints," *IEEE Trans. Inform. Theory,* Vol. IT-32, No. 4, pp. 533 - 542, 1986.

[2] T. Berger and Z. Ye, "Entropic Aspects of Random Fields on Trees, IEEE Transactions on Information Theory, Volume 36, pp. 1006–1018, 1990.

[3] L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* (Wadsworth: Belmont, CA, 1984)

[4] W. Chang and M. Kam, "Asynchronous distributed detection," *IEEE Trans. Aerospace and Electr. Syst.*, Vol. 30, pp. 818 - 826, 1994.

[5] K. Demirbas, "Distributed sensor data fusion with binary decision trees," *IEEE Trans. Aerospace and Electr. Syst.*, Vol. AES-25, No. 5, pp. 643 - 649, 1989.

[6] Yu. Ermoliev and R. J-B Wets, Eds. *Numerical Techniques for Stochastic Optimization.* (Springer-Verlag: Berlin, 1988)

[7] R. Grossman, X. Qin, and W. Xu, and H. Hulen, and T. Tyler, "An Architecture for Scalable Digital Libraries," *Proceedings of the 14th IEEE Computer Society Mass Storage Systems Symposium,* S. Coleman, editor, IEEE, 1995, pages 89–98.

[8] R. L. Grossman, "Optimization Driven Data Mining," *Magnify Technical Report,* Number 96-R2, January, 1996.

[9] R. L. Grossman, "Early Experience with a System for Mining, Estimating, and Optimizing Large Collections of Objects Managed Using an Object Warehouse," *Magnify Technical Report,* Number 96-R4, February, 1996.

[10] P. Kall and S. W. Wallace, *Stochastic Programming.* (Wiley-Interscience: Chichester, UK, 1994)

[11] H. V. Poor, *An Introduction to Signal Detection and Estimation - Second Edition.* (Springer-Verlag: New York, 1994)

[12] A. Prékopa, *Stochastic Programming.* (Kluwer: Dordrecht, The Netherlands, 1995)

[13] D. Teneketzis and Y.-C. Ho, "The Decentralized Wald Problem," *Information and Control,* Vo. 73, pp. 23 - 44, 1987.

[14] J. N. Tsitsiklis, "Distributed Detection," in *Advances in Statistical Signal Processsing - Vol. 2: Signal Detection,* H. V. Poor and J. B. Thomas, Eds. (JAI Press: Greenwich, CT, 1993)

[15] P. K. Varshney, *Distributed Detection and Data Fusion.* (Springer-Verlag: New York, in press)

[16] V. V. Veeravalli, T. Basar and H. V. Poor, "Decentralized sequential detection with a fusion center performing the sequential test," *IEEE Trans. Inform. Theory,* Vol. 39, No. 2, pp. 433 - 442, 1993.

[17] V. V. Veeravalli, T. Basar and H. V. Poor, "Decentralized sequential detection with sensors performing sequential tests," *Mathematics of Control, Signals and Systems,* Vol. 7, pp. 292 - 305, 1994.