Feature Selection and Error Tolerance for the Logical Analysis of Data

Endre Boros[*], Kathryn Davidson[†], Craig Bowles[‡], Dennis Fayneburg[§]

July 29, 2005

Abstract

We present a new process of analyzing data to determine critical attributes in a classification problem. Our method allows for maximal error tolerance in the original data while producing a choice of simple Boolean formulas for classifying the data. We implement the method and test it on the Wisconsin Breast Cancer Database with classification accuracy over ninety percent. This process utilizes an existing dualization algorithm and the method of binarizing data with missing bits.

[*]Rutgers University, RUTCOR
[†]University of Pennsylvania
[‡]Cornell University
[§]Rutgers University

1 Introduction and Definitions

The logical analysis of data is used to uncover hidden structural information in data sets [1]. The target of our analysis will be data sets which are classed, that is, divided into a small number of classes. The goal is to determine the most important features from the set while allowing a maximal tolerance for error in the data. These features and error tolerance can then be used to find a Boolean formula which will correctly classify a large percentage of the data. We give a procedure for binary classed data set, but this can be extended by separating the classes in halves and repeating the procedure until the halves all consists of a single class.

To allow error in our data, we use the method of partially defined Boolean functions with missing bits [2] so that the parts of the data that are not within our margin of error become missing bits in the Boolean function. The functions will be partially defined because our input is finite. To be clear, we present some definitions taken from Boros et al. [2]:

- A *Boolean function* is a mapping $f : B^n \to B$ where $B = \{0, 1\}$.
- A *partially defined Boolean function* (pdBf) is defined by sets (T, F) where $T, F \subseteq B^n$.
- A *pdBf with missing bits* (pBmb) is defined by sets $(\tilde{T}, \tilde{F})$ where $\tilde{T}, \tilde{F} \subseteq M^n$ and $M = \{0, 1, *\}$.

Our input data set will consist of a large number of vectors, each assigned a Boolean value "positive" or "negative." We call the set of positives S^+ and the set of negatives S^-. Besides this Boolean value, each vector will include values for k other attributes; call k the arity of the data vectors.

2 Error Vectors

Our final feature selection and formula will be based on our choice of error, so we begin by finding possible error vectors. Let Δ be the set of difference vectors $\delta = |a - b|$ where $a \in S^+$, $b \in S^-$. We would like a tolerance for error that is as large as possible while still distinguishing each positive from each negative. Therefore, each error vector must be less than each of the difference vectors in at least one coordinate. To find these maximal errors, we first find the minimal difference vectors and then find their complements to obtain the maximal difference vectors.

A vector δ_{min} is a *minimal difference vector* if for each $\delta \in \Delta$, δ_{min} is less than δ in some coordinate. We find the set Δ_{min} of δ_{min}'s by choosing any

$\delta_0 \in \Delta$ and removing all elements of Δ that are greater or equal to δ_0 in every coordinate. Then we choose another element δ_1 of the reduced set Δ and again remove all elements that are greater or equal to δ_1 in every coordinate. We repeat this process until there are no non-minimal vectors left in Δ. Now $\Delta = \Delta_{min}$.

In order to find maximal error vectors, we input Δ_{min} into an existing dualization process [3]. The dualization process outputs the complement dual vectors of Δ_{min} inside a box of dimension k and side length b.

The dualization output from these minimal difference vectors is also a minimal set, so since we're looking for maximal error we need to take the complement of each vector. These complements of the dual output are then divided in half to find our maximal error vectors to allow the error both above and below a certain threshold:

$$\forall i, j[\epsilon_i = \frac{1}{2}b - \delta_{i_j}]$$

The set E of maximal errors ϵ can be sorted in many ways: we chose the greatest coordinate sum, greatest number of high ($> m$) coordinates, least number of coordinates equal to zero, greatest number of coordinates equal to the maximum, and the maximum of the minimum coordinate value. An error $\epsilon \in E$ is chosen that appears near the top of many of these lists, and is used to guide the binarization process.

3 Thresholds for Binarization

The next step toward binarization is to choose thresholds τ for attributes $A_{1..k}$ so that we can classify each data vector $c \in S^+ \cup S^-$ as above or below τ for that attribute A_j with a confidence of ϵ. That is, we can classify c based on threshold τ (of attribute j) if $c_j > \tau + \epsilon_j$ or $c_j < \tau - \epsilon_j$. If neither is true, then we cannot classify c using threshold τ.

To be sure to classify every c we can take our set of thresholds to be the set of all midpoints $p = \frac{a_j + b_j}{2}$ for some $a \in S^+, b \in S^-, j \in 1...k$. The resulting pdBf is called the *master pdBf*. By definition, the master pdBf uses every $\tau = p$, but a goal of this work is to be efficient and to create a binary table that uses the smallest number of τ's. We could accomplish this by eliminating irrelevant τ's or by constructing a table with the most relevant τ's. We choose to proceed by way of construction.

To construct the set of relevant τ's for our selected error ϵ, we first look at our original set of difference vectors Δ. Each difference vector is associated with the pairs of original positive and negative vectors for which it is the difference.

For each difference vector that is greater than ϵ in some coordinate j, we look at the positive/negative pairs that are associated with it and add their midpoints in coordinate j to our set of relevant τ's.

We then have each $j \in 1...k$ associated with a list of τ's. Some j's may have more than one τ, and some may have none. Let this hash of keys j and values τ be called T. Let t equal the total number of τ thresholds over all k.

4 Binarization

Our process of binarization is a function from our original vectors $C = S^+ \cup S^-$ to a set of binary vectors $B \subset \{0, 1, *\}^t$. This function depends on our threshold hash T and an error vector ϵ. For each $c \in C$, we check each coordinate k, and for each k we check each $\tau \in T$. For each τ, if $c_k \geq \tau + \epsilon_k$, then $\beta_{c,k} = 1$. If $c_k \leq \tau - \epsilon_k$, then $\beta_{c,k} = 0$. Finally, if $\tau - \epsilon_k < c_k < \tau + \epsilon_k$, then we let $\beta_{c,k} = *$, a missing bit.

5 Formula

This process produces a pdBf $\beta(c)$ where $c \in S^+ \cup S^-$. For each c, if $c \in S^+$, let $\beta(c) \in B^+$. If $c \in S^-$, let $\beta(c) \in B^-$. This is well defined because $S^+ \cap S^- = \emptyset$. We can picture β as a table of vectors $\beta(c)$ for some $c \in C$ called the *binary table*. We want to see what distinguishes B^+ from B^-.

The binary table defines and produces a Boolean formula that will distinguish B^+ from B^-, but it is generally extremely long. Instead, we try to reduce the table so that a more lucid formula is available. First, we eliminate each column t with a missing bit "*", because this column does not convey any information. We also condense the table so that for each pair in B, if $\beta(i) = \beta(j)$ then we eliminate $\beta(j)$ and keep track of how many time a $\beta(i)$ is repeated.

The binary table is now greatly reduced in columns t, and we know how many times a row vector $\beta(i)$ appears in the table. If one vector appears much more frequently than any other, we can roughly use it as our formula. This was the case in our test data. If this were not the case, the few most common β vectors can be run through any Boolean formula finder (such as [6]) which may produce a slightly more complicated formula. If there are many widely varying most frequent Boolean vectors, one may doubt the possibility to find a simple formula to classify the data.

6 Implementation and Testing

The algorithm described above was implemented with Perl and tested on the Wisconsin Breast Cancer Database [5]. We used 950 vectors from the data set, each of arity $k = 9$ and for each k the maximum value was $b = 10$.

The running time is roughly 4 hours, most of the computationally intensive part spent on finding the minimal difference vectors Δ. We ran the program on the entire WBCD vector set, and by choosing an error vector that appeared near the top of most of our lists, we arrived at the following formula:

$$(c_3 < 7) \wedge (c_5 < 5) \wedge (c_6 < 8) \Longrightarrow c \in S^+$$

In this database, column 3 rates uniformity of cell shape, column 5 rates single epithelial cell size, and column 8 rates the normal nucleoli. This formula correctly classified 432/444 (97 percent) of the positive examples, and no negative examples. The WBCD in particular is characterized by widely varying values for the negative vectors, so the formula was most easily phrased in terms of the positive samples.

Next, we ran the procedure on roughly 10 percent of the WBCD data. The resulting formula that held true for 30/33 of these positive training examples and only 2 negatives also held true for 372/406 of the positive testing data and no negatives. While these numbers are difficult to stake a patient's life on, they are still remarkably accurate for such a simple formula. Since our program is designed to allow a user interaction in the choices of error and formula, we believe the accuracy can only improve if a specialist in the field (in this case, a doctor or medical scientist), were able to use the program. However, more of this type of training and testing should be done on other databases to verify the accuracy of this procedure.

7 Applications and Further Research

As we have seen above for the Wisconsin Breast Cancer Database, this process can be used to select important features in medical databases. The strength of this process is that it is flexible enough to allow for human input in the choices of error vectors, important attributes, and end formulas. Since often patterns arise in such data that humans know to be irrelevant, there is the ability to disregard these and use more relevant choices. However, we think it would be beneficial to investigate heuristic methods of determining thresholds instead of only choosing them from among the set of midpoints to provide even more choices to the user. In general, our process is sure to be improved by finding more flexible error and threshold choices.

In applications of machine learning to natural language processing, we might like to exercise our choice by introducing or restricting the data by certain rules, such as those learned through previous processing. We may also like to include error, which in the case of language processing can be used along with weights to model the likelihood of a certain structure or definition, building on the work of Collins and others [4]. Another application is genetic testing, where a result of a DNA test may involve thousands of pieces of information, about which we are only familiar with a small number. Our program can provide a choice of error vectors that would allow most of the information we cannot meaningfully identify to be included in the error. The procedure then presents the tester with a choice of important features, from which she is free to choose those which are most useful. Any other large classed database where there is uncertainty in the data and it is desirable to have human input in choosing formulas can benefit from this feature selection process.

8 References

[1] Boros, E., Hammer, P.L., Ibaraki, T., and Kogan, A. "Logical Analysis of Numerical Data," Math. Programming, 79, (1997), 163-190

[2] Boros, E., Ibaraki, T., and Makino, K., "Variations on Extending Partially Defined Boolean Functions with Missing Bits," June 6, 2000

[3] Boros, E., Gurvich, V. and Elbassioni K. "Dualization in Integer Boxes." http://rutcor.rutgers.edu/ boros/IDM/DualizationCode.html July 1, 2005

[4] Collins, Michael. lecture, Institute for Research in Cognitive Science. Philadelphia, PA. November 4, 2005

[5] Mangasarian, O.L. and W. H. Wolberg. "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, p 1,18

[6] Rudell, Richard. "Espresso Boolean Minimization" http://www.csc.uvic.ca/ csc485c/espresso/instructions.html July 18, 2005