An Algebraic Approach to Data Mining: Some Examples

Robert L. Grossman and Richard G. Larson
Laboratory for Advanced Computing, University of Illinois at Chicago
851 S. Morgan St M/C 249, Chicago IL 60607
{grossman, rgl}@uic.edu

September 15, 2002

This is a draft of the paper R. L. Grossman and R. G. Larson, An Algebraic Approach to Data Mining: Some Examples, Proceedings of the 2002 IEEE International Conference on Data Mining, IEEE Computer Society, Los Alamitos, California, 2002, pages 613-616.

Abstract

In this paper, we introduce an algebraic approach to the foundations of data mining. Our approach is based upon two algebras of functions defined over a common state space X and a pairing between them.

One algebra is an algebra of state space observations, and the other is an algebra of labeled sets of states.

We interpret H as the algebraic encoding of the data and the pairing as the misclassification rate when the classifer f is applied to the set of states χ.

In this paper, we give a realization theorem giving conditions on formal series of data sets built from D that imply there is a realization involving a state space X, a classifier $f \in R$ and a set of labeled states $\chi \in R_0$ that yield this series.

1 Introduction

Let R denote an algebra of functions formed by state space observations, that is, maps $f : X \longrightarrow k$. Classifiers will be elements of $f \in R$.

The second algebra of functions R_0 consists of state space observations with finite support, that is, maps $\chi : X \longrightarrow k$ with finite support. Labeled sets of states will be elements of $\chi \in R_0$.

Note that R and R_0 are k-algebras. We also assume that there is a pairing (the misclassification rate)

$$\ll f, \chi \gg \in \mathbf{R}, \qquad f \in R, \quad \chi \in R_0.$$

Fix a space D of labeled data elements. We define a labeled learning set to be an element of D^*, the set of words $d_1 \cdots d_k$ of elements $d_i \in D$. If k is a field, the $H = kD^*$ is a k-algebra with basis D^*. In this paper we study formal series of the form

$$p = \sum_{h \in D^*} p_h h.$$

By a formal series, we mean simply a map $H \longrightarrow k$, associating each element of H with the series coefficient p_h. The coefficient p_h is the classification (or misclassification) rate for the learning set h. We make no assumptions about convergence. Formal series occur in the formal theory of languages, automata theory, control theory, and a variety of other areas.

Fix a formal series $p \in H^*$. Note that this is an object associated not with a single data set but with a family of data sets. For our applications, these may be thought of as associated with a series of experiments involving different, but related data sets. We address a standard question: given a formal series p, built from the data D, is there a state space X, a classifier $f : X \longrightarrow k$, and a set of initial states that yield p? This is called a realization theorem. The state space captures the essential data which are implicit in the series p. The formal definition is given below.

We now give two examples of formal series.

Example 1. The first example is motivated by the problem of learning a model to predict credit card fraud given transaction data, where each transaction is labeled fraudulent or not fraudulent. Let the set D consist of labeled data elements which are triples (p, a, f). Here p is an account ID, a bounded integer. The second component a is an integer between \$1 and \$1000 representing the transaction amount. The third component is a label 0 or 1, the first labeling a good transaction and the second a bad or fraudulent one. We assume that the first two components are distributed according to the uniform distribution, while 98% of the labels of the third component are 0, so that fraud occurs for approximately 2% of the transactions.

For this example, we let the state space $X = \mathbf{Z}^2$ which we think of as embedded in $\mathbf{R}^2$. A labeled set of states is a map $\chi : X \longrightarrow k$, with finite support. We assume that each state $x = (x_1, x_2)$ with $\chi(x) \neq 0$ is associated with a (unique) profile id or account id. The component x_2 is the transaction amount a of the last transaction associated with the account. There must be at least one such transaction or $x = 0$, which we assume is not the case. The component x_1 is the transaction amount a of the second to the last transaction associated with with the account. If there is no such transaction, then x_2 is 0.

Given a data set $d \in D^*$, consisting of words built from transaction triples (p, a, f), there is a natural action $\chi \cdot d$, which we think of the result of a data set updating a set of states. The profile in χ corresponding to p is left shifted by a and the resulting profile takes the label f.

We define a classifier $f : X \longrightarrow k$ as follows: $f(x) = 1$, if $x_1 = 1$ and $x_2 > 250$ and zero otherwise. This is motivated by the standard practice of testing a stolen credit card with a one dollar transaction and then buying an expensive item. Finally, we assume that 90% of credit card transactions for one dollar are fraudulent. Given these assumptions, the formal series $\sum_{d \in D^*} \ll f, \chi \cdot d \gg \cdot\, d$ is realizable by construction and the distribution of the coefficients can be computed easily as an exercise. Note that the series doesn't converge and though the coefficients may be arbitrarily close to 1, on average, they tend to be less than 0.2.

Example 2. For the second example, take the same data set D consisting of triples (p, a, f) as above and form words by as in Example 1.

Define a formal series

$$\sum_{d \in D^*} c_d \cdot d,$$

where this time the coefficient c_d is assumed to be a random variable on $[0, 1]$ which we assume to be *independent* of d. Assume that this formal series has a realization, say on a state space $\mathbf{R}^n$ so that there are functions f, $\chi : X \longrightarrow k$ with $c_d = \ll f, \chi \cdot d \gg$. Then either the action $\chi \cdot d$ is independent of d in which case the coefficients c_d are constant, which is a contradiction, or the coefficients $c_d = \ll f, \chi \cdot d \gg$ do in fact depend upon d, which violates our assumption that the c_d are a random variable independent of d. We conclude that this series doesn't have a finite dimensional state space realization.

Formal series elegantly capture the structure of a variety of infinite objects that arise in computation. Realization theorems use a finiteness condition to imply that the infinite object can be represented by a finite state space. One of the most familar realization theorems is the Myhill–Nerode theorem. In this case, the infinite object is a formal series of words forming a language; the finiteness condition is the finiteness of a right invariant equivalence relation, and the state space is a finite automaton. In our case of data mining, the infinite object is a formal series of learning sets comprising a series of experiments, the finiteness condition is described by the finite dimensionality of a span of vectors, and the state space is $\mathbf{R}^n$. The Myhill–Nerode theorem, and, more generally, languages, formal series, state space representations (such as provided by automata), and finiteness conditions play a fundamental role in the foundations of computer science. Our goal is to introduce analogous structures into data mining. We now briefly recall the Myhill–Nerode theorem following [4], page 65.

Let D be an alphabet. D^* is the set of words in D, and $L \subset D^*$ is a language. A language L defines an equivalence relation $\sim$ as follows: for $u, v \in D^*$, $u \sim v$ if and only if for all $w \in D^*$ either both or neither of uw and vw are in L. An equivalence class $\sim$ is called right invariant with respect to concatenation in case $u \sim v$ implies $uw \sim vw$ for all $w \in D^*$.

Theorem 1.1 (Myhill–Nerode) *The following are equivalent:*

1. *L is the union of a finite number of equivalence classes generated by a right invariant equivalence relation.*

2. *The language $L \subset D^*$ is accepted by some finite automaton.*

In the sections below, we point out further analogies between the Myhill–Nerode thereom and the data mining realization we prove below in Theorem 5.2. For now, we point out that a language $L \subset D^*$ naturally defines a formal series. Fix a field k and the k-algebra $H = kD^*$. Given a language L, define the formal series $p \in H^*$ as follows:

$$p(h) = \begin{cases} 1 & \text{if } h \in L \\ 0 & \text{otherwise} \end{cases}$$

In this paper, we prove a Myhill–Nerode type theorem for data mining. We have two innovations in this paper:

1. We introduce a realization theorem for data mining. In particular, we introduce a natural finiteness condition associated with an infinite series of data sets that comprise a series of experiments. As far as we are aware, realization theorems and these types of finiteness conditions in data mining have not been studied previously.

2. The distinction between data, states, data updates, data attributes, and derived attributes is usually ignored in alternative approaches. Rather one works with a classifier f on a data space D. In our approach, we clearly distinguish the data $d \in D$, states $x \in X$ formed from the data using derived attributes, and the action of new data d' updating the states $d' \cdot x$.

2 Data, States, and State Space Observations

Let D denote a space of labeled data elements and D^* the set of words $d_1 \cdots d_k$ formed from data elements $d_i \in D$. We emphasize that each d_i is labeled.

Fix a field k. Let $H = kD^*$ denote the vector space witb basis D^*. Then H is an algebra whose multiplication is induced by the semigroup structure of D^*, which is simply concatination.

Fix a space X. Elements $x \in X$ are called states and X is called the state space. Fundamental to our approach is the introduction of two algebras of functions defined over a common state space X and a pairing between them. Let R denote an algebra of functions formed by state space observations, that is, maps $f : X \longrightarrow k$. Classifiers will be elements of $f \in R$, that is, a classifier associates a value or label to each state.

The second algebra of functions R_0 consists of state space observations with finite support, that is, maps $\chi : X \longrightarrow k$ with finite support. By finite support we mean that the set $\{x : \chi(x) \neq 0\}$ is finite. Labeled sets of states will be elements of $\chi \in R_0$.

The other fundamental assumption is that there is an action of the data $h \in H$ on the functions $f \in R$ which satisfies

$$h \cdot (f + g) = h \cdot f + h \cdot g$$

$$h \cdot (\alpha f) = \alpha(h \cdot f) = (\alpha h) \cdot f,$$

for $f, g \in R$, $\alpha \in k$. That is, R is an H-module. Since the state space X and the function space R are closely connected, this is roughly equivalent to having an action of H on X. In addition, the action also satisifies the identity

$$h \cdot (fg) = \sum_{(h)} (h_{(1)} \cdot f)(h_{(2)} \cdot g),$$

where the map $H \longrightarrow H \otimes H$, $h \mapsto \sum_{(h)} h_{(1)} \otimes h_{(2)}$ is called a comultiplication. In this case R is called a H-module algebra. An algebra H with a comultiplication and units for both the multiplication and comultiplication, all of which satisfy certain compatibility conditions is called a bialgebra. See [1] and [3] for details.

We assume that

1. R and R_0 are H-module algebras.
2. There is a pairing

$$\ll f, \chi \gg \in \mathbf{R}, \qquad f \in R, \quad \chi \in R_0.$$

This is our setup for the analysis of data mining from an algebraic point of view. To summarize: we are given a bialgebra H, two function algebras R and R_0, and a pairing between them. We interpret H as the algebraic encapsulation of the data and the pairing as the misclassification rate when the classifer f is applied to the set of states χ. The process of updating and computing derived attributes is encapsulated in the action $\chi \cdot h$, for $h \in H$ and $\chi \in R_0$.

We now show how the same algebraic structure can be used to describe automata following [3]. As in the description of automata above, let D denote a finite alphabet, and D^* the set of finite strings of letters of D. Then D^* is a semigroup with operation concatination, and with identity the empty string ϵ. $H = kD^*$ is a bialgebra. Let $L \subset D^*$ be a language, and let p be the characteristic function of L. Let M denote a finite automaton accepting the language L, let S be the set of states of the automaton, let s_0 be the initial state, and $F \subseteq S$ the set of accepting states. Then a word $w \in D^*$ is accepted by the automaton if and only if $s_0 \cdot w \in F$.

We now re-interpret this structure using the algebras R and R_0 introduced above. Let R denote the algebra of k-valued functions on the state space S. Then R is a commutative k-algebra. Let R_0 denote the set of characteristic functions on the set S which are 0 everywhere except at a single point where they are 1. Note that R_0 is an H-module defined by $h \cdot f(s) = f(s \cdot h)$ if $h \in D^*$.

Define

$$f(s) = \begin{cases} 1 & \text{if } s \in F \\ 0 & \text{otherwise} \end{cases}$$

Note that $w \in L$ if and only if $s_0 \cdot w \in F$ if and only if $f(s_0 \cdot w) = 1$ if and only if $p(w) = (w \cdot f)(s_0) = 1$. Define

$$\ll f, \chi \gg = f(s'), \qquad f \in R, \quad \chi \in R_0,$$

where s' is the point of S where $\chi(s') = 1$. Now if χ is the characteristic function of the initial state s_0,

$$\chi(s) = \begin{cases} 1 & \text{if } s = s_0 \\ 0 & \text{otherwise} \end{cases}$$

Then

$$p(w) = \ll f, w \cdot \chi \gg, \tag{1}$$

in case the language defined by $p \in H^*$ is accepted by some finite automaton. Equation (1) is the fundamental equation defining a realization. The left hand side contains the coefficients of a formal series, while the right hand side is based on a state space and functions defined on it. We give an analogous theorem (Theorem 3.4) for the formal series of learning sets arising in data mining in Section 5 below.

3 Realizations

We first define an algebraic finiteness condition on formal series of learning sets $p \in H^*$.

Definition 3.1 *If H is a bialgebra, its* primitive *elements are defined by*

$$P(H) = \{ h \in H \mid \Delta(h) = 1 \otimes h + h \otimes 1 \},$$

where the map $\Delta : H \longrightarrow H \otimes H$, $h \mapsto \sum_{(h)} h_{(1)} \otimes h_{(2)}$ is the comultiplication.

Definition 3.2 *The algebra H^* has a left H-module algebra structure given by $(h \rightharpoonup p)(k) = p(kh)$, for h, $k \in H$, $p \in H^*$. We say that the formal series of learning sets $p \in H^*$ has* finite Lie rank *if $\dim P(H) \rightharpoonup p$ is finite.*

Finite rank is a naturally occuring condition and occurs in the Fliess theorem from control theory, in the Myhill–Nerode theorem from automata theory, and in hybrid systems [3].

In this section, we state and prove a simple realization theorem.

Let D denote a *data space*. More precisely an element of D is a triple whose first element is a Profile IDentifier (PID) chosen from a finite set $\mathcal{I}$ and used to keep track of the various states, whose second element is a label chosen from a finite set of labels $\mathcal{L}$, and whose third element is an element of S, a set of data associated with PIDs. In short, $D = \mathcal{I} \times \mathcal{L} \times S$, where $\mathcal{I}$ is the set of PIDs and $\mathcal{L}$ is the set of labels. We use heavily the facts that $\mathcal{I}$ and $\mathcal{L}$ are finite.

Recall that $H = kD^*$ denote the vector space with basis D^*. Let $U = kS$ denote the vector space with basis S. Then H is an algebra whose multiplication is induced by the semigroup structure of D^*, which is simply concatination. Also $U = kS$ is an algebra whose structure is induced by the semigroup structure of S. We use the mappings from H^* to U^* induced by adjoiniing labels and PIDs to elements of S.

A *simple formal learning series* is an element $p \in U^*$. We can think of a simple learning series p as an infinite series $\sum_{s \in S} c_s s$. Essentially, a simple formal learning series is a formal labeled learning series without the labels and PIDs.

Definition 3.3 *Let R be a commutative algebra with augmentation ϵ. We say that $p \in U^*$ is* differentially produced by the pair (R, f) *if*

1. *there is right U-module algebra structure on R;*
2. *$p(u) = \epsilon(f \cdot u)$ for $u \in U$.*

The basic theorem on the existence of the state space is the following, in which the state space is the vector space with basis $\{x_1, \ldots, x_n\}$.

What Theorem 3.4 gives us is the existence of a finite state space based on a finiteness condition on the series p.

Theorem 3.4 *Let $p \in U^*$. Then 1) implies 2).*

1. *p has finite Lie rank;*
2. *there is a subalgebra R of U^* which is isomorphic to $k[[x_1, \ldots, x_n]]$, the algebra of formal power series in n variables; there is $f \in R$ such that p is differentially produced by the pair (R, f).*

Proof: See [2].

We end by revisiting the Myhill–Nerode Theorm:

Theorem 3.5 (Myhill–Nerode) *Let D be a finite alphabet and $H = kD^*$. Let $p \in H^*$. Then 1) implies 2):*

1. *$\dim(H \rightharpoonup p)$ is finite and p takes on the values 0 and 1.*
2. *there is a finite state space S with k-algebra of functions R on S, and a function $f \in R$, such that $p_h = (h \rightharpoonup f)(s_0)$.*

We see that the Data Mining Realization theorem (Theorem 3.4 above) is a generalization of the Myhill–Nerode theorem using a more general finiteness condition, a finite dimensional vector space for a state space, and a bit of added complexity because of the labels and the presence of a finite number of initial conditions.

References

[1] R. Grossman and R. G. Larson. The realization of input-output maps using bialgebras. *Forum Mathematicum*, 4:109–121, 1992.

[2] R. L. Grossman and R. G. Larson. An algebraic state space realization theorem for data mining. submitted for publication.

[3] R. L. Grossman and R. G. Larson. An algebraic approach to hybrid systems. *Journal of Theoretical Computer Science*, 138:101–112, 1995.

[4] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley, Reading, Massachusetts, 1979.