IDENTIFICATION OF NON-NUMERIC SYSTEMS

Robert M. Haralick, University of Kansas
Gary J. Minden, University of Kansas

ABSTRACT

In this paper we discuss a technique for identifying non-numeric systems based on an observed input output sequence of the system. The time invariant relationship among the input variables and output variables is determined by clustering using the measured association and similarity between the variables. The system function is found by modeling the system with a chain sequential machine and then determining a reduced input/output equivalent sequential machine. The reduction algorithm has memory requirements which vary linearly with the length of the original observation sequence.

Such identification problems arise when we desire to model the way people behave as they interact with one another, or the way they behave in a small group setting based on a sequence of observations of the interacting people.

Introduction

In order to understand our world, we try to explain the organizations and interactions we perceive. When these explanations are tightly organized we think of the world as a system and our explanations are really descriptions of the system structure and function. In this paper we discuss a method for determining the structure of a class of non-numeric systems.

A system is a structured device which produces an output related to its past and present inputs according to some deterministic rule. The rule is called the structure of the system. At any particular time, if we have access to a system, we can, so to speak, go inside it and completely observe its state. The state of the system at time t is the minimum description which completely specifies what any observation of the system would tell at time t. The system is identifiable if the history of observable variables has enough information to specify the present state of the system and the system function.

The structure of systems that are non-numeric by nature cannot be represented by linear equations, differential equations, or equations requiring a metric space. Instead, such systems must be represented as a general sequential machine. The sequential machine has a set of states, S, a set of inputs Σ, a set of outputs Δ, a function relating the present state and input to the next state; and δ, a function relating the present state and input to the output, λ.

We represent δ and λ as ternary relations:

\[ \delta \subseteq S \times \Sigma \times S \]

\[ \lambda \subseteq S \times \Sigma \times \Delta, \text{ where } \delta \subseteq \Sigma \]

satisfy that there exists a \( s \in S \) such that

\[ (s, a, s') \in \delta \] if and only if there exists a \( s \in S \) such that \( (s', a, s') \in \lambda \).

To synthesize a system we must determine or define each of the elements of the sequential machine.

We begin by observing the system to be modeled for some time. At each time unit the system is observed an n-tuple of measurements becomes available. If the system is observed for m time units, then the raw data consists of a sequence of m n-tuples. We assume that k variables are inputs to the system and the remaining n-k variables are outputs. From this raw data the input and output sets, S and Δ, can be determined.

Using this straightforward approach to defining the input and output sets is computationally inefficient for many of the possibilities or combinations of values the different variables can take on never occur. Any extra complexity that is introduced at this stage makes the sequential machine reduction procedure take much more computer resource to complete. We suggest that the input and output variables be clustered independently using association and similarity measures [6].

Tight clusters mean that there is high interrelationship between the values the variables take on so that within a cluster of variables only a few combinations of values from the entire set of possibilities ever occur. The few combinations which do occur can be recorded in a new variable set. The recorded data can then be used to define a chain sequential machine which is much easier to reduce than the one defined by the raw data.

Clustering

In order to cluster variables, a measure of associations, indicating how related two variables are, must be defined. Depending on the situation and the type of variables, there are a variety of measures which could be used. For example, the following measure of association based on co-occurrence can be used for clustering binary variables:

\[ a_{ij} = \frac{P(x_i=1, x_j=1)P(x_i=0,x_j=0) - P(x_i=1, x_j=0)P(x_i=0,x_j=1)}{\sqrt{P(x_i=1)P(x_i=0)P(x_j=1)P(x_j=0)}} \]

(3)

where \( P \) is a probability measure. A binary relation pairing highly associated variables can be defined as:

\[ R_a = \{ (x_i, x_j) \mid a_{ij} > \theta_a \} \]

(4)

where \( \theta_a \) is a threshold value. This method of pairing variables is insufficient when some variables may be measuring characteristics which are indirectly related to several variables. To improve the distinction between variables we add a measure of similarity. Similarity is a relationship of commonness. The extent to which two things have common parts, elements, or relations is the
extent to which they are similar. The formal definition of similarity between the i-th and j-th variables is:

\[ s_{ij} = 1 - \frac{1}{2(N-2)} \sum_{n=1}^{N} |a_{in} - a_{jn}| \]  

(5)

where \( a_{in} \) is the association between the i-th and n-th variables and \(-1 \leq a_{in} \leq +1\). A similarity relation \( R_s \) can be defined as:

\[ R_s = \{(x_i, x_j) | s_{ij} > \theta_s \} \]  

(6)

A combined relation involving both association and similarity can be defined as follows:

\[ R = R_d \cap R_s = \{(x_i, x_j) | a_{ij} > \theta_d \text{ and } s_{ij} > \theta_s \} \]  

(7)

This relation identifies those variables that are both highly associated and highly similar. Values of \( \theta_d \) and \( \theta_s \) typically select the upper decile of \( a \) associations and similarities.

A Graph Theoretic Method for Determining Clusters

In this section we discuss a general clustering algorithm which can be used to compute clusters of the \( R \) relation. The graph theoretic clustering method is motivated by [5] and [3] who determined the clusters of a graph by finding all the cliques of the graph and iteratively merging those cliques having high enough overlap. We begin with some definitions. Let S be a set and \( R \subseteq S \times S \) be a binary relation on S. The pair \((S, R)\) is called a digraph, or when \( R \) is symmetric, a graph. The elements of \( S \) are called nodes. If a pair of nodes \((x, y)\) belongs to \( R \), then we say that \( y \) is a neighbor of \( x \). The set of all nodes \( y \) such that \( y \) is a neighbor of \( x \) is called the neighborhood of \( x \). (Note that unless \( R \) is symmetric, \( y \) being a neighbor of \( x \) does not imply that \( x \) is a neighbor of \( y \).)

In the spirit of [3], we define a cluster as a maximal chain of highly compact or connected neighborhoods. [3] used cliques, the most highly compact neighborhoods possible. However, in order to reduce the computational burden of finding all cliques, we will use neighborhoods which are less dense but easier to compute. To do this we must define a density function and use it to determine the compact neighborhood around each node. The conditional density \( D(x|y) \) of node \( x \) given node \( y \) is defined as the number of nodes in the neighborhood of \( y \) which have \( x \) as a neighbor.

\[ D(x|y) = \#\{n \in S | (n, x) \in R \text{ and } (y, n) \in R \} \]  

(8)

Notice that if \( R \) is symmetric then \( D(x|y) \) is just the number of neighbors common to both node \( x \) and node \( y \). In this case

\[ D(x|y) = \#\{\text{neighborhood } (x) \cap \text{neighborhood } (y) \} \]  

(9)

Another measure needed in determining clusters is a measure of compactness for a set of nodes. For any subset \( B \) of \( S \), we define the compactness \( C(B) \) as the ratio of the number of pairs of nodes in \( B \times B \) which belong to \( R \) to the total number of pairs in \( B \times B \).

\[ C(B) = \frac{\#(B \times B \cap R)}{\#B \times B} \]  

(10)

If \( M \) is a maximal subset of \( S \) such that \( C(M) = 1 \), then \( M \) is a clique of the relation \( R \). If \( x \) is a node of \( S \), then there are some number \( N(x) \geq 1 \) of cliques of \( R \) containing \( x \), and of these cliques there are one or more that are larger in size. These larger cliques we call major cliques.

We will suggest a way to compute dense regions which are almost major cliques. We call these regions near-cliques. For each node \( x \) in \( S \) and integer \( k \) define the region \( Z(x, k) \) by

\[ Z(x, k) = \{ y \in S | D(y|x) \geq k \} \]  

(11)

For small values of \( k \), the region \( Z(x, k) \) around node \( x \) is likely to be large and loose. As \( k \) becomes larger, the region \( Z(x, k) \) becomes smaller and more tightly interconnected and compact.

If \( C \) is a major clique of size \( M \), then \( x, y \in C \) implies that \( D(y|x) \geq M \). Thus \( C \subseteq Z(x, k) \). Therefore, the largest-sized clique associated with each node \( x \) must be contained in \( Z(x, k) \) for any \( k \leq M \) and we have \( k \leq M \leq \#Z(x, k) \). Then any value of \( k \) which does not satisfy \( k \leq \#Z(x, k) \) could not be the size of a major clique for node \( x \). Hence it is only natural to consider

\[ Z(x) = Z(x, N) \text{ where } N = \max(k \#Z(x, k) \geq k) \]  

(12)

as a good compact region around \( x \).

Because each \( Z(x) \) contains only those nodes with highest conditional densities, there is a reasonable basis for expecting that each \( Z(x) \) will contain much more than the nodes of the major cliques of node \( x \). To guarantee this, we want to consider only those sets \( Z(x) \) whose size and compactness are high enough. Given a compactness threshold \( \theta \) and a minimum size threshold \( \text{MINSIZE} \), we define a near-clique of \( R \) to be a subset \( D \subseteq C \) satisfying

(13) \( D = Z(x) \) for some \( x \in S \).

(14) \( C(D) > \theta \).

(15) \( \#D \geq \text{MINSIZE} \).

The clusters of \( R \) can be determined by taking the union of those near-cliques of \( R \) which overlap one another to a high enough degree. Let \( a \) be the specified degree of overlap. We define the near-clique relation \( F \) as the set of pairs of near-cliques \((D_1, D_2)\) such that the fraction of nodes in \( D_1 \) which are also in \( D_2 \) is greater than \( a \), or the fraction of nodes in \( D_2 \) which are also in \( D_1 \) is greater than \( a \).

\[ F = \{(D_1, D_2) | D_1, D_2 \text{ are near-cliques of } R, \quad \#D_1 \cap D_2 / \#D_1 \geq a \text{ or } \#D_1 \cap D_2 / \#D_2 \geq a \} \]  

(16)

(17) \( \#D_1 \cap D_2 / \#D_1 \geq a \) or \( \#D_1 \cap D_2 / \#D_2 \geq a \)

The relation \( F \) is symmetric and reflexive. Its transitive closure is an equivalence relation. We would like a cluster to be the union of the set of near-cliques in each equivalence class. Although the equivalence classes are naturally disjoint, it is possible for the union of the near-cliques in one class to be contained in the union of the near-cliques in another class. The simplest way to handle this is to iterate by merging clusters which overlap much in the same way that the near-cliques were merged. The iteration can proceed until the clusters can no longer be merged.

Recording of the Raw Data

The results of the clustering can be used to define
the input and output sets of the sequential machine. The clustering not only can be used to decrease the number of degrees of freedom mentioned previously, but also can give insight into the structure of the system. For example, if three clusters of output variables were found, we might assume that three, rather than one, sequential machines were operating in parallel.

Reencoding the raw data can reduce the degrees of freedom. The reencoding process works with one cluster at a time. If there are k variables in a cluster and each variable can take on v values, then there are \( v^k \) possible codes in the raw data. From the raw data we can determine that only p codes occur for the k variables in the cluster.

Then we can reencode the \( v^k \) possible codes into p new ones. When each cluster is handled in this manner, the number of degrees of freedom in the data is reduced. The reencoded data defines an input and output sequence for the sequential machine.

Modeling the System Function with a Sequential Machine

To find the system function we use a chain sequential machine model. A chain sequential machine has simple \( \delta \) and \( \lambda \) functions: if \( \text{in} = t_1, t_2, \ldots, t_m \)

and \( \text{out} = o_1, o_2, \ldots, o_m \) are the input and output sequences, let \( T = t_1, t_2, \ldots, t_m \) be a sequence of states for a chain sequential machine. Then,

\[
\delta = \{ (t_j, i_j, t_{j+1}) \} \quad j = 1, \ldots, m-1 \quad (18)
\]

\[
\lambda = \{ (t_j, o_j) \} \quad j = 1, \ldots, m-1 \quad (19)
\]

![Figure 1. An example of a deterministic chain sequential machine.](image)

While the chain sequential machine completely describes the observed input-output variables, it gives little information about the structure of the system, does not help explain the system, is large and cumbersome to use, and cannot be used for simulating the system. To determine a functionally input-output equivalent but reduced machine, one need only apply the state minimization procedures for incompletely specified machines [2], [1], [9], [11], [12], [7], [10], [13]. Unfortunately, the techniques which have been published involve too much memory for a machine with 1000 states and 40 to 50 input variables.

Sequential Machine Reduction Algorithm

Reducing the number of states involves merging states of the chain sequential machine into non-overlapping subsets called equivalence classes which do not alter the input-output behavior. Each equivalence class then becomes a state in the reduced machine. In this section we briefly discuss one easily implementable state merging process for sequential machine reduction.

Two states, \( x \) and \( y \), are equivalent if the outputs of \( x \) and \( y \) are the same for the same input and the next states of \( x \) and \( y \) are equivalent. Let \( E = \{ S \} \), a binary equivalence relation on the set of states. The pair \((x, y) \in E \) if the following conditions hold:

1. \((x, y) \in E \) and \((x, q) \in E \) imply \((y, q) \in E \) \( (20) \)
2. \((x, y) \in E \), \((x, s) \in E \) and \((y, s') \in E \) imply \((x', y') \in E \) \( (21) \)
3. For each \( x \in S \), \((x, x) \in E \) \( (22) \)

A consistent relation \( F \) is generated by a pair of states if the pair of states is equivalent and the successors of the states are equivalent. The relation \( F \) contains the original pair and succeeding pairs of consistent states.

To find the equivalence relation for a chain sequential machine we use the following algorithm. Define a sequence of equivalence relations \( E_0, E_1, \ldots, E_n \) which determine a partition of the state set in the following iterative way:

Step 1. \( E_0 = \{ (S, S) \in S \times S \} \), \( n = 0 \).

Step 2. If there is a pair of states not yet considered, go to Step 3; otherwise stop.

Step 3. Determine the consistent relation \( F \) generated by the next pair of states to be considered.

Step 4. Determine the equivalence relation \( F^\top \), the transitive closure of \( F \).

Step 5. If for every \( x \in S \)

\[
E_n(x) \subset F^\top(x) \quad \text{or} \quad F^\top(x) \subset E_n(x).
\]

Then go to Step 6; otherwise go to Step 2.

Step 6. \( E_{n+1} = E_n \cup F^\top \)

\( n = n+1 \)

Go to Step 2.

In the example of Figure 1 states \( s_1 \) and \( s_5 \) are equivalent, the succeeding states; \( s_2 \) and \( s_6 \), and \( s_3 \) and \( s_7 \) are also equivalent. Since \( s_4 \) and \( s_8 \) do not accept the same input the sequence of equivalent states ends with \( s_3 \) and \( s_7 \). A consistent relation, \( F \), is formed by the pairs of equivalent states. In this example, \( F = \{ (s_1, s_5), (s_2, s_6), (s_3, s_7) \} \).

The transitive closure of \( F \) is the same as \( F \).

Since this is the first non-empty \( F \), \( E_1 = \emptyset \), and \( E_1 \cup F^\top = F \). Steps 2 through 6 are repeated for each pair of states.

While this algorithm does not produce optimum results in the sense that the reduced machine may not be minimal, the memory requirements are of the order \( n \), where \( n \) is the number of initial states. Other algorithms, which produce minimal machines, require on the order of \( n^2 \) memory locations. This becomes prohibitive when \( n \) is much larger than 100.

Conclusion

In this paper we have briefly sketched an approach
of modeling non-numeric systems with a chain sequential machine. We have suggested clustering the input and output variables in order to reduce the complexity of the input and output sets. We have formulated a sequential machine reduction algorithm to simplify the initial chain sequential machine to a smaller one with the same input/output behavior. We are currently in the process of applying this modeling approach to mother-infant interaction analysis and will report on the results soon.

REFERENCES


