Classifying Numeric Information for Generalization

Michael Lebowitz

May, 1983

CUCS-53-83

Classifying Numeric Information for Generalization[1]

Michael Lebowitz
Department of Computer Science -- Columbia University
New York, NY 10027

Abstract

Learning programs that try to generalize from real-world examples may have to deal with many different kinds of data. Continuous numeric data may cause problems for algorithms that search for identical aspects of examples. This problem can be surmounted by *categorizing* the numeric data. However, this process has problems of its own. In this paper we look at the need for categorizing numeric data, and several methods for doing so. We concentrate on the use of a heuristic, *looking for gaps*, that has been implemented in the UNIMEM computer system. An example is presented of this algorithm categorizing data about states of the United States.

1 Introduction

Programs that learn by generalization from examples must be able to deal with many different kinds of data. Continuous numeric data, which is prevalent in many domains, can cause serious problems for such systems. This is simply because generalization may depend upon noticing identical components of data items, and numbers are rarely exactly the same from example to example. Numeric data necessitates the creation of discrete categories to allow generalization to take place. This categorization process creates interesting problems of its own, which we will look at in this paper. The categorization process described here is being used as part of a computer system, UNIMEM, designed to accept facts about a domain, store the information in long-term memory, and, most relevant to our purposes here, generalize from similar examples.

To illustrate the categorization problem, we will look at data from one domain UNIMEM

---

[1]This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-82-C-0427.

has been tested on information about states of the United States. In particular, consider the state population and area data in Figure 1.

STATE	POPULATION	AREA (squaremiles)
Alabama	3,800,000	51,000
Alaska	300,000	569,000
Arizona	2,700,000	113,000
Arkansas	2,200,000	53,000
California	23,600,000	158,000
Colorado	2,800,000	104,000
Connecticut	3,100,000	5,000
Delaware	600,000	2,000
Florida	9,700,000	58,000
Georgia	5,400,000	58,000
Hawaii	900,000	6,000

Figure 1: State population and area data

A typical generalization that we might wish UNIMEM to make from this data is that states with small areas usually have small populations (we are dealing here with pragmatic generalizations that describe situations that are usually, but not necessarily always, true). Clearly this cannot be done by looking for states with identical population or area values. One logical approach is to categorize numeric data, the population and area in this case, and then try to generalize. If we categorize the populations of Alaska, Delaware and Hawaii as "small" and do the same for the areas of Connecticut, Delaware and Hawaii, then we could hope to conclude that states with small areas often have small populations.

Most of the research done in learning from examples has not been concerned with categorizing numeric data. Such programs have either started out with discrete data ([Winston 72; Mitchell 82], among others), have had their processing of numbers built-in (for example, Meta-DENDRAL [Buchanan and Mitchell 78] apparently did not try and learn what values in a spectrogram constituted a peak), or have used clustering techniques (e.g., [Michalski 80]).

One program that is concerned with numeric data is Langley's BACON [Langley 81], which processes scientific data and develops hypotheses about mathematical relations among the values. BACON was able to derive several laws of nature, such as Ohm's law and Kepler's third law from appropriately selected input data.

Crucially from our point of view, BACON only derived continuous functions of the data fields. It did not deal with cases where generalizations about the data were dependent on categorization. Thus, while BACON might be able to deal a case where states' populations were directly proportional to their populations, it could not deal with the more likely possibility that large states have large populations, and small states small populations, but the relation is no more precise than this. In addition, it cannot deal with partial relations, where, for example, very small states might have small populations, but nothing can be said about larger states (some have small populations, some large). Each of these problems requires categorization of the data.

The goal of the categorization process is to derive categories that "make a difference" in generalization. We would like the categories to be such that different classes distinguish items for generalization. Simple methods, like categorizing by number of standard deviations from the mean often fail in this regard as they allow, for example, items with almost identical values to fall into different classes.

There are at least three different, and probably mutually applicable ways to categorize numeric data. These are: 1) *number heuristics*, rules we know about numbers, for example, "look for gaps"; 2) *domain information*, logical reasons, such as governmental laws, to expect items to have different behavior across given break points; 3) *consistency in generalizations*, ranges of numbers that are consistent across generalizations created from other data.

In this paper, we will concentrate on the use of number heuristics, after touching on the other methods briefly. First, we will present a brief description of the generalization framework within which we are operating.

2 Background -- Generalization-Based Memory

We will describe briefly here the generalization method used by UNIMEM. These methods are based on those used in IPP [Lebowitz 83] and RESEARCHER [Lebowitz 82a] and are further described in [Lebowitz 82b].

The generalization process used in UNIMEM begins by making tentative generalizations about a situation based on only a small amount of input data. Each input item is

referred to as an *instance*. UNIMEM then records specific items in memory in terms of the generalizations made, under *GEN-NODES*. It is also possible to make more specific generalizations and to record these, as well, under the more general cases.

The storage of instances and sub-GEN-NODEs under a GEN-NODE is done with discrimination networks (D-NETs) [Charniak, et al. 80]. D-NETs provide an efficient way to retrieve objects stored with a given set of indices. In the UNIMEM memory model, every feature of an instance or sub-GEN-NODE is initially used as an index, resulting in shallow, bushy D-NETs that allow retrieval of an object given any of its features.

The UNIMEM generalization process itself is relatively simple. Given an input event, as a set of features, UNIMEM searches through memory, using the hierarchy of D-NETs to search efficiently, to find the GEN-NODE that best describes the new information. Then it checks to see if any of the instances stored under that GEN-NODE have additional similarities to the new instance. If so, a new GEN-NODE is made, in effect creating a new concept.

3 Methods of Categorization

3.1 Domain Information

As in most areas of Artificial Intelligence, domain-specific knowledge can be applied to the problem of categorizing numeric data. In particular, we may know factors of a domain likely to cause values across a breakpoint to generalize differently. For example, if a particular Federal law takes effect only for states with populations above 5,000,000, then we would expect that to be a logical point to break categories apart. Normally, information of this sort will be used to initially categorize data. However, in many cases, we will not have such relevant facts, and have to look for other methods.

3.2 Consistency in Generalizations

As mentioned earlier, the goal in categorizing numeric data is to allow the making of generalizations based on the categories created. It seems logical, then, that if we have made generalizations based on other information, either non-numeric or previously categorized data, then examining the values of a new field in these generalizations will

help in categorization. If a field takes on one range of values in all (or most) instances stored with a generalization, and another range under another generalization, then these ranges make logical categories, as the new field would then participate in the generalizations, and hopefully others.

As an example of the process we have in mind, if, using our state data, we had made one generalization about Alabama, Florida and Georgia, and another about Arkansas, Arizona and Colorado, then we might break population categories between 2,800,000 and 3,800,000 so that population would be constant across these generalizations.

While we expect this method of categorization to be very useful in the future development of UNIMEM, it is not adequate by itself. It cannot handle cases where *all* the data is numeric (at least until an initial start-up period of generalization has occurred), or even cases where numeric data only generalizes with other numeric data. Since domains of this sort seem common, we must look for another categorization method to allow an initial set of generalizations to be made.

3.3 Number Heuristics -- Looking for Gaps

Despite the virtues of the methods of categorization mentioned so far, often these methods will not be applicable, and we will simply have a group of numbers that must be categorized. It is possible to do this using general heuristics that we have available about numeric data. In particular, the heuristic we will concentrate on in this section is one familiar to anyone who has ever "curved" an exam. We will refer to it as *looking for gaps*. We will also mention a secondary heuristic involving the number of categories that is created.[2]

Looking for gaps is based on the idea that values that are close together are unlikely to be fundamentally different. If close values are not to be in different categories, then the only place to put category boundaries is in gaps devoid of values. Thus, when UNIMEM categorizes numeric data, it uses a method similar to that used by an instructor looking for the breakpoints between grades on an exam -- it sort the values of a field and looks for the largest possible gaps.

---

[2] In a rather different context, [Riesbeck 81] suggests another form of number heuristic, "check scale" that has a similar flavor to those mentioned here.

In order to employ a gap-finding method, we must employ another heuristic, that the number of categories we derive should not be too large or too small. From experience with generalization, we know that if we create too many categories, not enough instances will involve each, and that if we create too few, then the instances in each category will not be similar enough to warrant generalization. A rule of thumb seems to be that about 5 to 10 categories per field are useful. It would be possible to formulate a similar rule that looked at the number of elements in each class. The reason for our decision to look at the number of classes is related to the advisability of looking at only a sample of a field's values, which we will discuss shortly.

One additional relevant fact about numbers (too basic to even be called a heuristic) is that the absolute magnitude of a piece of numeric data is virtually meaningless. A gap of 100,000 might be large when looking at state area in square miles, but insignificant when dealing with state population. We adopt the natural solution of looking for large gaps relative to the sizes of the data items that bound each gap, rather than for any absolute size gaps. (A normalization of data strategy would have much the same effect.)

These various heuristics lead to the UNIMEM categorization algorithm illustrated in Figure 2. The basic idea is that we begin with an optimistic view towards the size gaps we might find (25% was the initial gap size used in analyzing state data), and see how many gaps that large can be found. If the number found is over a threshold (5 was used), then we stop. Otherwise, the gap size is gradually decreased (2% at a time) until enough gaps are found, or the gap size becomes too small (under 5%). (The latter condition would imply the data is too nearly continuous to be analyzed in this fashion). All the parameters in this algorithm can be easily modified and are open to experimentation.

Figure 3 illustrates the results of applying this algorithm to the populations of the fifty states. The resulting six categories seem to reflect a division that works quite adequately in making generalizations.

There are several problems with the "gap finding" algorithm as presented so far. The most obvious is that it cannot be directly applied to any domain with an unbounded

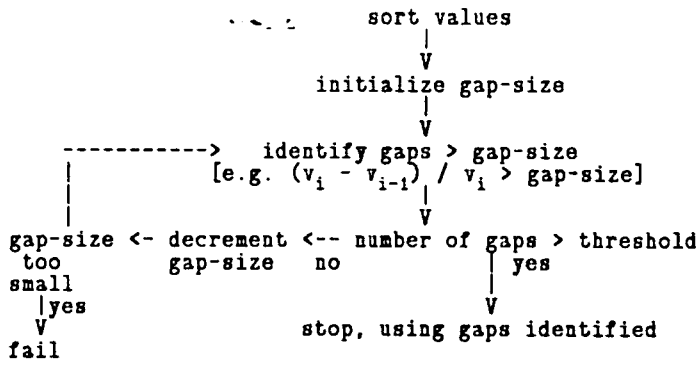

Figure 2: Gap-finding categorization algorithm

pop < 1,650,000

ALASKA	DELAWARE	HAWAII	IDAHO
MAINE	MONTANA	NEBRASKA	NEVADA
NEW HAMP	NEW MEX	NORTH DAK	RHODE ISLAND
SOUTH DAK	UTAH	VERMONT	WYOMING

1,650,000 < pop < 2,250,000

ARKANSAS	WEST VIRGINIA

2,250,000 < pop < 3,600,000

ARIZONA	COLORADO	CONN	IOWA
KANSAS	KENTUCKY	MISSISSIPPI	OKLAHOMA
OREGON	SOUTH CAROLINA		

3,600,000 < pop < 6,550,000

ALABAMA	GEORGIA	INDIANA	LOUISIANA
MARYLAND	MASS	MINNESOTA	MISSOURI
N CAROLINA	TENNESSEE	VIRGINIA	WASHINGTON
WISCONSIN			

6,550,000 < pop < 8,250,000

NEW JERSEY

8,250,000 < pop

CALIFORNIA	FLORIDA	ILLINOIS	MICHIGAN
NEW YORK	OHIO	PENN	TEXAS

Figure 3: Categorized state population data

number of instances, or even a finite, but very large number of instances (which are exactly the kinds of domains generalization-based memory is best suited for). In neither case could all the values of a field be accumulated, much less sorted. In addition, as more and more values are examined over a limited range, it becomes quite unlikely that

there will be any perfect gaps. Instead, there will simply be "high density" and "low density" ranges.

Fortunately, these problems can be solved with one modification of our algorithm. Instead of looking at *all* the values for a field, we simply pick randomly a *sample* of the values. Then we apply the gap finding algorithm. Statistical reasoning indicates that a modest sample will be adequate to capture the main properties of the field. We expect, by and large, the "low density" ranges of values for a field not to contribute values to the sample, and hence leave gaps where categories can be delimited. The state population example in Figure 3 was actually accomplished by processing a sample of 25 of the 50 states. Note that for statistical reasons, the sample size needed is effectively constant, and does not grow proportionally to the number of instances in the domain.

4 Conclusion

We have presented here the problem of dealing with numeric data in in the context of generalizing from examples. We have also shown one method for categorizing data, based on finding gaps in the data, that has been implemented in the UNIMEM computer system. Several other potential solutions were also suggested. The problems concerning numeric data that we have begun to attack are ones that must be dealt with in programs that hope to learn from complex, real-world data.

REFERENCES

[Buchanan and Mitchell 78] Buchanan, B. G. and Mitchell, T. M. Model-directed learning of production rules. In D. A. Waterman and F. Hayes-Roth, Ed., *Pattern-Directed Inference*. Academic Press, New York, 1978, pp. 297 - 312.

[Charniak, et al. 80] Charniak E., Riesbeck, C. K., and McDermott, D. V. *Artificial Intelligence Programming*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1980.

[Langley 81] Langley, P. "Data-driven discovery of natural laws." *Cognitive Science 5*, 1 (1981), 31 - 54.

[Lebowitz 82a] Lebowitz, M. Intelligent information systems. Columbia University Department of Computer Science, 1982.

[Lebowitz 82b] Lebowitz, M. "Correcting erroneous generalizations." *Cognition and Brain Theory 5*, 4 (1982), 367 - 381.

[Lebowitz 83] Lebowitz, M. "Generalization from natural language text." *Cognitive Science 7*, 1 (1983), 1 - 40.

[Michalski 80] Michalski, R. S. "Pattern recognition as rule-guided inductive inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1980).

[Mitchell 82] Mitchell, T. M. "Generalization as search." *Artificial Intelligence 18* (1982), 203 - 226.

[Riesbeck 81] Riesbeck, C. K. Failure-driven reminding for incremental learning. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981.

[Winston 72] Winston, P. H. Learning structural descriptions from examples. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1972.