Keith B. Olson

An NP-Complete Problem Arising from a Brute Force Search for Limits of Subsets of B^N

Abstract

A brute force search for limits of a subset of $B^n = \{0,1\}^n$ is shown to be NP-Complete. This is accomplished by demonstrating equivalence to an extension of the known NP-complete problem SATISFACTION. No results are postulated concerning sufficient conditions for the existence of such a limit.

Key words: NP-Complete, limits, binary vector spaces, computational complexity

Motivation

In a fundamental paper in 1967, E. M. Gold proposes a paradigm for learning processes entitled *Identification In the Limit (IIL)*. This branch of learnability has developed the following model over the years since Gold's work (See, for example, Ben-David (1992)). Suppose there is a finite or countable class of 'concepts' C that is known to both a teacher and a student. The teacher chooses a concept T from C, and then repeatedly passes chunks of information about T to the student. The student, based on the information passed to him, successively modifies his conjecture concerning the identity of T. The student is successful if the infinite sequence of his conjectures stabilizes on the correct concept T. One way to mathematically represent this process is to treat the concepts as elements of the vector space $\{0,1\}^n = B^n$. Information from the teacher is presented in the form of other elements of B^n which are known to agree with T for at least k of its n elements.

Keith B. Olson, Montana Tech of the University of Montana, Butte, Montana 59701

Consider the problem of transmitting data to Earth from a very remote spacecraft. If the transmitter is weak, or if there is a lot of interference between the spacecraft and Earth, there will be a great deal of noise that will obscure the correct data. Many error-detecting and error-correcting codes are currently in use, but in severe cases, they can require the transmission of more bits than the data itself. We can consider the spacecraft as the teacher in the first example above, and the Earth station as the student. How many elements of B^n (retransmissions of the raw data) must be sent to guarantee that we can correctly determine the correct data value? If the spacecraft is far away (say at the orbit of Pluto), the time required to request a retransmission of the data and then receive it is on the order of 10.5 hours or more for each request. If 10 retransmissions are needed, this approach would occupy over 100 hours. If, on the other hand, one request can be made for the 10 retransmissions, they could all be received in something on the order of 11 hours. But we must know how many retransmissions would be required to guarantee that we could determine the correct data value.

These problems give rise to the following definition and the question that follows it. The question was first posed by Shai Ben-David (1992).

> ***Definition:*** *Let B^n be the set of all n-dimensional vectors over {0,1}, and A be a subset of B^n. The element **x** in B^n is said to be a **K-LIMIT** of the set A, $1 \le k \le n$, if for any choice of k subscripts, there is an element **y** in A, **y** not equal to **x**, such that **x** and **y** agree on these k subscripts.*
>
> ***Question K-LIMIT:*** *Is there an integer K(k,n), such that for **A** a subset of B^n, $|\mathbf{A}| > K(k,n)$ implies that **A** has a k-limit?*

One approach that can be taken to solving this problem is to look at all subsets of B^n, determine which have limits, and then find the largest subset that does not have a limit. If this is done for a number of different values of *n*, then perhaps a pattern can be determined and a theorem would follow. The difficulty here is that the number of elements in B^n is $m = 2^n$. This means that the number of possible subsets of B^n is 2^m. Even for small values of n, this number is very large. If n = 5, for example, there are 32 elements in B^n, and over 4 *billion* subsets that would have to be considered. A more common value for n in the spacecraft problem is 32, and the number of subsets is in excess of $10^{1,260,000,000}$, an almost unimaginably large number. Problems of this type are said to be of *exponential* complexity, because the time necessary to work through the problem increases exponentially with the size of the problem (n in this case). We would like to find an algorithm that would solve the problem in a time period that is at most a polynomial function of the size of the problem.

Background

A problem is said to be of class *NP* if it is only solvable in polynomial time on a non- deterministic computer. A non-deterministic computer is one that has the capability of pursuing all of its possible sequences of action in parallel (See, for example, Gersting 1982:362). Since all contemporary computers are deterministic, this is essentially equivalent to saying that the problem cannot be solved in an amount of time that is polynomial in the size of the problem. In a fundamental paper, Cook (1971) showed that there is one particular problem in the class NP that has the property that every other NP problem can be reduced to it in polynomial time. That problem is referred to as SATISFIABILITY, and is stated by Garey and Johnson in their book *Computers and Intractability* as follows: (The format used here is the traditional one for stating NP-Complete problems. It consists of a statement of the elements of the problem, and a question that is to be answered either yes or no.)

> INSTANCE: Set U of logic variables, collection C of clauses over U.
>
> QUESTION: Is there a satisfying truth assignment for C?

A *logic variable* is one that can assume one of two values: True or False. For a given set $U = \{u_i \mid 1 \le i \le n\}$, the set V consisting of all the elements of U and the logical negations of those elements is referred to as the set of *literals* over U. A *clause* is the disjunction of a subset of the set V; i.e., an expression of the form

$$c = v_i + v_j + \ldots + v_k.$$

A *satisfying truth assignment* is an assignment of True or False to each of the logic variables in such a manner that at least one of the terms in each clause is True.

To establish that another problem is NP-Complete, one must show that it is either an *extension* of a known problem, or that it is *equivalent* to a known problem. A problem **X** is equivalent to problem **Y** if a solution to either one gives rise to a solution of the other with a transformation time that is a polynomial in the size of the problem. **X** is an extension of **Y** if a solution of **X** always provides a solution to **Y**, but not vice versa. We will use both methods in the sequel.

In what follows, we will work exclusively with 3-limits. The extension to k-limits for arbitrary k is not conceptually difficult, but it is a notational nightmare. We leave it to the reader to see that the extensions do work.

We begin with an extension of the problem SATISFACTION, proceeding as follows. We begin with a set of logic variables $U = \{u_1, u_2, \ldots u_n\}$. We generate a new set of variables $V = \{v_1, v_2, \ldots v_n\}$ by assigning v_i to be either u_i or $-u_i$, $1 \leq i \leq n$. As will be shown, there is usually a rule to be used in making the choice between u_i and $-u_i$, and once this assignment has been made, it will remain fixed through the remainder of the problem. Now, there are $m = [n(n - 1)(n - 2)]/6$ selections of (i,j,k), with $1 \leq i < j < k \leq n$. Number these 1, 2, 3, . . . m. If the p^{th} such triple is (r,s,t), set $x_p = v_r v_s v_t$ (the juxtaposition of logic variables here implies a logical *and*). This defines a new set of logic variables that we will call X. We are now concerned about the relationship between truth assignments for U and truth assignments for X.

It is clear that any truth assignment for U will generate truth values for each element of X. Once each u_i is assigned to be either True or False, then the values of the v_i's are also determined; if $v_i = u_i$, then the values are the same; otherwise, they are opposites. With the v_i's determined, it is a simple matter to apply the rules of logic to determine the value for each of the x's.

Now, consider the problem of assigning values to the x_i's and determining from those values what would be the appropriate values for the u_i's. If all the x_j's are false, then the problem is straightforward — simply assign all the v_i's to be false also, and from that determine the values for the u_i's. Similarly, if all the x_j's are true, then all the v_i's can be assigned to be true, and then the u_i's are again easily determined.

This leaves us with the case that we have some x_j's being true and some false. For a given x_j to be true, the three v's involved in its definition must also be true. We thus begin by taking all the x_j's that are true, and assign all the v_i's that appear in these terms to be true also. Now consider any x_j that is false. We must find a v_i in the definition of this x_j that is not in the set already defined to be true, and assign it to be false. If such a v_i cannot be found, there is no truth assignment for the u's to produce the result in the x_j's; otherwise we proceed to the next x_j that is false. The transition from the v_i's to the u_i's is again straightforward. Thus we will either find a satisfying truth assignment or a contradiction as above; in either case the question has been answered in polynomial time.

To summarize, we have the following relationship. A given truth assignment for the u_i's will always produce an assignment for the x_j's, but the converse is not always true. In some cases a truth assignment for the u_i's may be deduced from one for the x_j's, but not always.

For the set X described above, define the expression

$$s = x_1 + x_2 + x_3 + \ldots + x_m$$

to be a ***complete clause*** over the set X. It is complete in the sense that it contains every element of the set X. Note that the nature of the set X, relative to U, depends on the selection of the literals in the set V. If U has *n* elements, then there are 2^n ways that the set V can be chosen, and each such choice gives rise to a corresponding set X. We should properly refer to X as X(V) to clearly express this dependency. If we have a collection of V's, say $\{V_1, V_2, \ldots V_k\}$, then we would also have a corresponding set of X's, $\{X(V_1), X(V_2), \ldots X(V_k)\}$. Let us denote this set by Z. For each set of X's in Z, we can construct a complete clause; the set consisting of these complete clauses will be called S. In the construction of these complete clauses, we will assume that the terms of each are numbered in a consistent manner. By this we mean that if the pth term in $X(V_1)$ is $v_{1,i}v_{1,j}v_{1,m}$, then the pth term in $X(V_r)$ is $v_{r,i}v_{r,j}v_{r,m}$, $2 < r < k$. We are now concerned about satisfying truth assignments for the set S. The problem that we wish to consider is as follows:

INSTANCE: A set **S** *of clauses in the logic variables of Z, which are in turn expressions in literals over a set of logic variables U.*

QUESTION: Is there a truth assignment for U that implies a truth assignment for **Z** *that in turn satisfies S?*

This problem is broken into two steps. First, one must find a satisfying truth assignment for **S** in terms of the logic variables of **Z**. This is essentially the problem SATISFACTION referenced above. To get from there to the variables of U is a polynomial time problem, as described above. This problem is therefore NP-Complete.

We will extend this problem by imposing an additional condition on the satisfying truth assignments. Inasmuch as the terms in each complete clause are numbered in the same manner, it does make sense to compare the ith term of one clause with the ith term of another clause. We will refer to the following problem as SEQUENTIAL SATISFACTION:

INSTANCE: A set **S** of clauses in the logic variables of **Z**, which are in turn expressions in literals over a set of logic variables U.

QUESTION: Is there a truth assignment for U that satisfies **S**, with the additional condition that for some numbering of the elements of **S** , the ith term of the ith clause is satisfied?

Any solution to SEQUENTIAL SATISFACTION is a solution to the unnamed problem posed above, but the converse is not true. The additional restrictions only make solutions harder to find, and this problem is also NP-Complete.

Results

We are now prepared to treat the fundamental question of the paper as stated in Section II. The elements of B^n are related to logic variables and truth assignments in the following way. For a given element b in B^n, we generate a V_b as follows: if the ith component of b is 1, then $v_{b,i} = u_i$; if the ith component is a 0, then $v_{b,i} = -u_i$. In this manner, we can obtain a one-to-one correspondence between the elements of B^n and the sets of literals V_b. If we have a subset *A* of elements of B^n, then each element will correspond to a V in this manner, and we generate a set V_A. From this we generate the set of logic variables **Z**, and the corresponding set of complete clauses S. Similarly, there is a one-to-one correspondence between elements of B^n and truth assignments for the variables U. Again, if *b* is an arbitrary element of B^n, and if the ith component of B is 1, then set u_i to be *True*; if that component is 0, then set u_i to be *false*. We can also proceed in the other direction: given a truth assignment for U, it can be transformed in a straightforward manner into an element of B^n. We are now prepared to prove the following theorem:

Theorem: If T is a subset of B^n, *b* an element of B^n, then *b* is a limit of T if and only if there is a subset A of T such that the set V_A and the truth assignments generated by *b* constitute a solution to SEQUENTIAL SATISFACTION.

Proof: Suppose first that *b* is a limit of T. Let A be the subset of T consisting of those elements that are used to establish the limit. Then for any choice of 3 subscripts, there is an element of A that agrees with *b* on those 3 subscripts. Number the possible arrangements of the 3 subscripts from 1 to m. Now, consider the element of A that agrees with *b* on the elements defined by the ith triple. If we label this element *a(i)*, then the ith term in $X(V_{a(i)})$ will be true. Since this is true for any set of three subscripts, it follows that the set V_A and the truth assignments generated by *b* form a solution to the sequential satisfaction problem.

Now, suppose that we have a set V_A which together with the truth assignment of *b* yields a solution to the sequential satisfaction problem. That is, for each i, $1 \le i \le m$, there is an element of A, call it *a(i)*, such that the ith term in the complete clause $X(V_{a(i)})$ is true. This says simply that *a(i)* and *b* agree on those three subscripts. Since i was general, there is an element of A that agrees with *b* on any set of 3 subscripts; in other words, *b* is a limit of A. Note that if T is a superset of A, *b* will be a limit of T also.

We now propose the following decision problem, which we will call 3-LIMIT:

INSTANCE: A set A of elements from B^n, $|A| = k$.

QUESTION: Does A have a 3-limit?

Corollary: 3-LIMIT is NP-Complete

Proof: A brute force approach to this problem would involve finding all subsets of B^n that have k elements, and then trying all elements of B^n to find if a limit exists. By the previous theorem, this amounts to finding a solution to SEQUENTIAL SATISFACTION. Hence, this problem is also NP-Complete.

Conclusions

The establishment of a problem as being NP-Complete does not in itself imply that the problem is insolvable and should be abandoned. Zadeh (1973) and others established in the early 1970's that the Simplex method of solving linear programming problems is in general of exponential complexity, and yet it is one of the most widely used methods in linear programming today. Complexity here only establishes a worst-case scenario. Garey and Johnson (1979) summarize the significance of establishing a result like the one above:

> . . . . the primary application of the theory of NP-Completeness is to assist algorithm designers in directing their problem-solving efforts toward those approaches that have the greatest likelihood of leading to useful algorithms.

In light of this statement, one should note carefully that the original problem *k*-LIMIT is still open: that is, is there a *k* such that $|A| > k$ will guarantee that A has a limit in B^n? However, the search for such a *k* by brute force is not likely to succeed for even moderate values of *n* because of the complexity of such a search. It is possible, however, that some of the work on the All-Nearest-Neighbor problem will yield results that will apply to this problem.

Acknowledgment

This research was sponsored by the NASA-University Joint Venture (JOVE) program.

Literature Cited

Ben-David, S. 1992. Personal communication following presentation of his paper "Can finite samples detect singularities of real-valued functions?" Proc. ACM Symp. on the Theory of Computing, Victoria, B.C., Canada. 24:390-399.

Cook, S. A. 1971. The Complexity of Theorem-Proving Procedures. Proc. ACM Symp. on the Theory of Computing, New York, NY. 3:151-158.

Garey, M. R. and D. S. Johnson. 1979. Computers and Intractability, W. H. Freeman and Co., San Francisco. p. 259.

Gersting, J. L. 1982. Mathematical Structures for Computer Science. W. H. Freeman and Co., San Francisco, CA.

Gold, E. M. 1967. Language Identification in the Limit. Information and Control 10:447-474.

Zadeh, N. 1973. A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms. Math Programming 5:255-266.