コンピュータと教育５－５
（１９８９. ５. １１）

バグ生成モデルと効率の良い Dempster-Shaferによる推論計算法

福田　章夫、徳田　尚之

宇都宮大学

修復理論で生成された代数的演算即ち減算での３６個のバグが階層構造を持つことを利用して、我々は Gordon & Shortliffe のアルゴリズムを使うと Dempster-Shafer 理論による推論計算が非常に効率良く行われることを示した。

実際、ある生徒が持っているバグがどのバグになるのか推定するのにバグの数が n 個あるとすると、従来の Dempster-Shafer理論では、2^nの計算量が必要であるし、Gordon & Shortliffe のアルゴリズムではこれが $O(n^3)$に減らせ、我々の方法ではこれが $O(n^{3/2})$にまで減らす可能性があることを示した。

An Efficient Evidential Reasoning Algorithm Using Dempster-Shafer Theory

A.Fukuda and N.Tokuda

Utunomiya University
350 Minemachi,Utunomiya-shi ,Tochigi 321 ,Japan

Taking an advantage of the hierarchical structure of 36 bugs generated by repair theory,we demonstrated that the Gordon-Shortliffe's method of computation of the Dempster-Shafer theory is remarkably effective in narrowing down and finally identifying any and all of the 36 bugs. We suggest that the computations can be reduced to $O(n^{3/2})$ rather than 2^n of the original Dempster-Shafer theory or $O(n^3)$ of Gordon-Shortliffe,if the parenal node and then the bug belonging to the node are computed,with n being the number of hypotheses involved.

§1 Introduction

The judicial decision making process in the present trial system is a most successfully implemented evidential reasoning system whereby a prosecutor presents confirmatory evidence to prove the defendant guilty while a lawyer presents disconfirmatory evidence to prove his/her innocence. A judge or a jury must assess the validity of evidence by combining confirmatory as well as disconfirmatory hypotheses carefully to make a final decision. An evidential reasoning system is a most popular inference system adopted widely in areas of artificial intelligence including expert systems. Unlike predicate or propositional logic which are designed to treat complete knowledge, evidence or information presented in most of AI application including expert systems are almost always uncertain, incomplete or ambiguous. Many prohabilistic reasoning methods are proposed for dealing with uncertain evidence; they include Certainty Factor Theory [14] implemented in Mycin, subjective Bayes' theory [6], Bundy's Incidence Calculus [4], the prohabilistic logic of Nilsson [11], Zadeh's fuzzy theory [16], Dempster-Shafer's theory [5,13] and Pearl's Bayesian belief network [12] to mention a few. Since these methods are fully reviewed in references [9,10], I will not dwell on here today. None of them seems readily adaptable to the inference system of a practical expert system as they are. The computations involved often explode in exponential time, are too tedious to implement or involve too crude approximations for effecting a valid reasoning.

Dempster-Shafer's theory (abbreviated as D-S theory hereafter) has two distinct advantages over other methods including the classical Bayesian approach [10]. First the theory is capable of representing and dealing with, uncertainty and/or total ignorance. Secondly the theory has an updating function of narrowing the hypothetical space with accumulation of evidence whereby aggregated evidence is successively combined by Dempster's rule, transforming once uncertain evidence into supporting definite hypothesis.

The purpose of the present paper is two-fold;

1. We would like to show that the G-S method of §3 is remarkably effective in computations of evidential reasoning, thus justifying the approximation of a smallest superset

2. Rather than $N = 2^{\Theta} = 2^n$, the G-S method requires computations of $O(n^3)$. We show a possibility that this can be improved to $O(n^{3/2})$, a marked improvement from the original 2^n.

We have chosen as a domain of inference the bug model as simulated by Brown & VanLehn's repair theory [3]. We implemented the repair theory in LISP and have generated 36 bugs in algebraic subtractions. But to our surprise, the 36 bugs form a strict hierarchical structure. We used these bugs as hypothetical space and successfully identified any and all 36 bugs using 3 - 7 problems.

§2 Repair Theory and Hierarchically Structured Bugs

The bug model is initiated by Brown & Burton [2]. Children are assumed to develop certain fixed bugs in algebraic subtractions and to always demonstrate the bugs when they face a right problem. Repair theory is proposed by Brown & VanLehn [3] to simulate a bug generation process. A remarkably simple GAO graph representing a bug-free algebraic procedure of subtraction constitutes the starting point for the bug simulations. One of 12 rules imbedded in 6 nodes of the GAO graph is deleted whereby an impasse occurs immediately because the rule deletion forces the firing of other rules whose preconditions are often violated. For example, if a rule for processing borrowing is deleted and a problem having smaller top cell than a bottom cell jumps into the next rule of normal subtraction whose precondition is top cell $\geq$ bottom cell, an impasse occurs immediately.

A total of 36 bugs are generated and most importantly they are found to form a tree structure with their parental nodes p_1, p_2, and p_3. p_1 node involves a blank in subtracters while p_2 node involves the borrowing process while p_3 involves the borrowing across columns.

§3 Gordon & Shortliffe's Method of D-S Theory

The G-S method of computations for the D-S theory proceeds along the first scene of judicial process described in §1 introduction of this paper. We will show convincingly that the G-S method provides a most efficient reasoning scheme when applied to evidential reasoning in identifying 36 bugs of §2. It is essential that the hypotheses (bugs in this case) form a strict hierarchy as we have shown in §2. We construct two trees; one for confirmatory evidence namely T and the other for disconfirmatory evidence T^c. We write,

$$T = \{P_1, P_2, \dots P_k, B_1, B_2, \dots B_{n-k}\} \qquad (1)$$

$$T^c = \{P_1^c, P_2^c, \dots P_k^c, B_1^c, B_2^c, \dots B^c_{n-k}\} \qquad (2)$$

P_1^c etc. denotes a complementary set to P_1. $P_1 \dots P_k$ are parental nodes directly below the root of the tree. Note that T nor T^c do not contain θ in their elements. The G-S method proceeds in three steps.

Step 1

As in Barnett [1], we use a simplified Dempster's rule of combination $\longrightarrow$ as applied to singletons. We apply the rule to combine all confirmatory evidence in T and then all disconfirmatory evidence in T^c separately. Let $s_1, s_2, \ldots s_\ell$ denote degrees of probability to support the singleton hypothesis B_i, then we have

$$m(B_i) = 1 - (1-s_1)(1-s_2) \ldots (1-s_\ell) \quad (3)$$
$$m(\Theta) = 1 - m(B_i) \quad (3)'$$
$$m(B_i^{\,c}) = 1 - (1-s_1^{\,c})(1-s_2^{\,c}) \ldots (1-s_\ell^{\,c}) \quad (4)$$
$$m(\Theta) = 1 - m(B_i^{\,c}) \quad (4)'$$

Here $m(B_i)$ etc denotes the bpa associated with the set B_i after all evidence of ℓ observations has been combined.

Step 2

We now compute the single aggregate bpa that assigns net belief to all elements in {T} namely $\{P_1, \ldots P_k\}$ and $\{B_1, \ldots B_{n-k}\}$ of eq. (1).

$$m_T = m_{P_1} \oplus m_{P_2} \oplus \ldots \oplus m_{P_k} \oplus m_{B_1} \oplus m_{B_2} \oplus \ldots \oplus m_{B_{n-k}} \quad (5)$$

$\oplus$ denotes an operation based on the Dempster's combination rule. $\longrightarrow$ We easily see that since $m_{P_i} \ldots m_{B_j}$ are non zero only on P_i, B_j and Θ, the combination rule will not generate any new subsets not in T by the set-theoretic approach. We will not manipulate the negated hypotheses at this stage and leave the combination of negated evidence to Step 3.

Step 3

We now combine disconfirmatory evidence with m_T of eq. (5). We choose any $B_i^{\,c}$ in tree T^c and try to compute $m_T \oplus m_{B_i^c}$ step by step. But we immediately see that the combination rule $\longrightarrow$ now produces many new subsets not in tree T. Suppose that $m_T \oplus m_{B_i^c}$ has produced a new subset C_i not in T. Because of the tree structure, we can always find a unique, smallest superset, B_ℓ say, of C_i in T. We write the new combination rule by choosing a smallest superset as

$$m_F = m_T \ominus m_{P_1}{}^c \ominus m_{P_2}{}^c \ominus \cdots \ominus m_{P_k}{}^c \ominus m_{B_1}{}^c \ominus \cdots \ominus m_{B_{n-k}}{}^c \quad (6)$$

Here m_F is a final bpa after combining the confirmatory as well as disconfirmatory evidence.

$4 Discussions and Conclusions

Typical results of evidential reasoning by G-S scheme of $3 are shown in Tables 1 and 2. Table 1 shows that the parental nodes to which a bug for identification belongs are more than convincingly identified with perhaps 5 problems. Once the parental node is identified, Table 2 shows that any of the bugs in the node are easily identified. In all cases, we should note that results improve markedly when disconfirmatory results are combined by step 3 of G-S method.

We have shown that, in spite of an approximation made in choosing a smallest superset rather than adding a new subset, the G-S method has a remarkable convergence in narrowing down the hypothetical space. We have in fact demonstrated convincingly that the approximation involved is indeed an excellent approximation and we did not have any difficulty in identifying bugs we seek through parental nodes with 3 to 7 problems. In fact, 5 problems seem quite adequate.

The approach we took differs from that of the G-S method. We first pinned down the parental node and then identified the bug from among bugs belonging to the particular parental node. Our approach suggests a marked improvement in the inference computations.

Suppose we have n hypotheses in a strict hierarchical structure including parental nodes.

The G-S method of computations suggest that

Step 1 requires computations of $O(n)$ while
Step 2 requires computations of $O(n^2)$ and
Step 3 requires computations of $O(n^3)$.

This alone is a big improvement over 2^n of the D-S theory when the tree structure is not utilized. We suggest that the computations can be improved and reduced to $O(n^{3/2})$. This can be achieved if we are able to subdivide the parental nodes to $n^{1/2}$ partitions. A simple calculation then shows that computations of step 3 reduces to $(n/n^{1/2})^3 = n^{3/2}$. We are looking for a method of devising the partitioning in such a way and would like to report on this later on.

References

1. Barnett,J.A. Computational methods for a mathematical theory of evidence, in Proc.Seventh International Joint Conference on Artificial Intelligence, Vancouver,BC,pp.868-875 (1982)

2. Brown,J.S. & Burton,R.B. Diagnostic models for procedural bugs in basic mathematical skills, Cognitive Science 2,pp.155-192 (1978)

3. Brown,J.S. & VanLehn ,K. A generative theory of bugs in procedural skills ,Cognitive Science 5,pp.379-426 (1980)

4. Bundy,A. Incidence calculus. A mechanism for probabilistic reasoning,Journal of Automated Reasoning 1,pp.263-288 (1985)

5. Dempster,A.P. Upper and lower probabilities induced by a multi-valued mapping,Annuals of Mathematical Statistics 38,pp.325-334 (1967)

6. Duda,R.O.,Hart,P.E. & Nilsson,N.J. Subjective Baysian methods for rule-based inference system, In Proc.of AFID 1970 National Computer Conf. 45,pp.1075-1082 (1976)

7. Fukuda,A. Buggy model and Dempster-Shafer calculations,M.S.thesis,Utunomiya University, (1989) (in Japanese)

8. Gordon,J & Shortliffe ,E.H. A mehtod for managing evidential reasoning in a hierarchical hypothesis space, Artificial Intelligence 26,pp. 323-357 (1985)

9. Ishizawa,H. Dealing with uncertain knowledge,M.S.thesis,Utunomiya University , (1987) (in Japanese)

10. Ishizawa,H & Tokuda,N ICAI system based on Dempster-Shafer theory, Symposium on Computer ans Educatin,Information Processing Society of Japan, pp.9-18 (1986) (in Japanese)

11. Nilsson,N.J. Probabilistic logic,Artificial Intelligence 28,pp.71-87 (1986)

12. Pearl,J. Fusion,propagation and structuring in belief networks,Artificial Intelligence 29,pp.241-289 (1986)

13. Shafer,G. A mathematical theory of evidence,Princeton University Press,(1976)

14. Shortliffe,E.H. Computer-based medical consultaton;MYCYN,New York,American Elseviers (1976)

15. Toyama,K. What is city-water method?,Collected work of Toyama,Sokosha Press (1980) (in Japanese)

16. Zadeh,L.A. Fuzzy logic and approximate reasoning,Synthesis 30,pp.407-428 (1975)

Table / Identifying Parental Nodes P_1, P_2, P_3

Bugs	Problems used \ Parental nodes		Θ	P1	P2	P3
B_1 (P_1 node)	TEST 2 (3 problems)	A	29	70		
		B	29	70		
	TEST 3 (4 problems)	A	7	75	17	
		B	8	90		
	TEST 4 (5 problems)	A	7	75	17	
		B	8	90		
B_{11} (P_2 node)	TEST 2 (3 problems)	A	7		75	17
		B	8		85	5
	TEST 3 (4 problems)	A	2	5	86	5
		B	2		96	
	TEST 4 (5 problems)	A		1	90	7
		B			98	
B_{35} (P_3 node)	TEST 2 (3 problems)	A	17		41	41
		B	24		17	57
	TEST 3 (4 problems)	A	17		41	41
		B	28		5	65
	TEST 4 (5 problems)	A	4		47	47
		B			3	84

Numbers in the table denote percentages truncated. Blanks denote zeros.

Column A: based on combination of confirmatory evidence only

Column B: based on combined results of both confirmatory as well as disconfirmatory evidence

Table 2 Identifying Bugs B_{24}, B_{32}, B_{35}, B_{16} of P_3 Node (7 problems are tried)

Student's bug \ Bugs			θ	p1	p2	p3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
B24	①	A																												90	2	2	2						2			
		B																												99												
	②	A													2															88	2	2	2						2			
		B																												99												
B32	①	A																																				94				
		B																																				99				
	②	A																																				94				
		B																																				99				
B35	①	A																										5						5	5	5	5	5	5	5	58	
		B																										7													87	
	②	A																										5						5	5	5	5	5	5	5	58	
		B																										7													87	
B16	①	A																				99																				
		B																				99																				
	②	A							49													49																				
		B							49													49																				

Numbers in the table denote percentages after truncating fractions.
Blanks denotes zeroes.

Column A: based on combined results of confirmatory evidence

Column B: based on combined results of confirmatory as well as disconfirmatory evidence

Column ①: The computational space is limited to P_3 node only

Column ②: The computational space is expanded to the entire bug space (as in Gordon-Shortliffe).