From: AAAI-87 Proceedings. Copyright ©1987, AAAI (www.aaai.org). All rights reserved.

Complexity in Classificatory Reasoning

Ashok Goel, N. Soundararajan, and B. Chandrasekaran

Department of Computer and Information Science
The Ohio State University
Columbus. Ohio 43210

Abstract

Classificatory reasoning involves the tasks of concept evaluation and classification, which may be performed with use of the strategies of concept matching and concept activation, respectively. Different implementations of the strategies of concept matching and concept activation are possible, where an implementation is characterized by the organization of knowledge and the control of information processing it uses. In this paper we define the tasks of concept evaluation and classification, and describe the strategies of concept matching and concept activation. We then derive the computational complexity of the tasks using different implementations of the task-specific strategies. We show that the complexity of performing a task is determined by the organization of knowledge used in performing it. Further, we suggest that the implementation that is computationally the most efficient for performing a task may be cognitively the most plausible as well.

1. Introduction

Classificatory reasoning is a *type* of knowledge-using reasoning that deals with performance of the classification *task*, and has received significant attention in research on knowledge-using problem-solving systems [Clancey, 1985; Gomez and Chandrasekaran, 1984]. Given a taxonomy of concepts in a domain, and a set of data describing a situation in the domain, the classification task is to determine which concepts are *present* in the situation. In diagnosing a device in some situation for instance, the classification task is to determine which device malfunctions are present in the situation, while in assessing an event in some situation the classification task may be to find which threats to some system are present in the situation. A task may be performed with use of a knowledge-using *strategy* appropriate for the task. The classification task may be performed with use of the strategy of concept *activation*, which is to activate concepts in the taxonomy for *evaluation* of their presence in a given situation.

Concept evaluation is a task by itself since it may "occur" not only in classificatory reasoning but also in other types of knowledge-using reasoning such as plan selection. Given a concept in a domain and a set of data describing a situation in the domain, the task of concept evaluation is to determine whether the concept is *appropriate* for the situation. The sense in which a concept is appropriate for a situation depends on the *type* of the concept; if the concept is a device malfunction for instance, then the concept is appropriate for the situation if it is *present* in it, and if the concept is a plan to thwart a threat then the concept is appropriate for the situation if it is *applicable* to it. Concept evaluation may be performed with use of the strategy of concept matching, which is to match a *knowledge structure* for the concept with the description of the situation, and determine a *likelihood* that the concept is appropriate for the situation by the degree of the match [Berliner and Ackley, 1982; Bylander and Johnson, 1987]. A strategy might be *implemented* in more than one way, where an implementation may be characterized by the *organization of knowledge* and the *control of information processing* it uses.

An important issue in classificatory reasoning is the computational complexity of performing the tasks of classification and concept evaluation. The complexity of performing a task depends on the implementation of the strategy used to perform it. In this paper we derive the computational complexity of concept evaluation and classification for different implementations of concept matching and concept activation, respectively. We show that the complexity of performing a task is determined by the organization of knowledge used in performing it. Further, we suggest that the implementation that is computationally the most efficient for performing a task may be cognitively the most plausible as well.

2. Complexity of Concept Evaluation

2.1. Definitions

Let c be a concept in a given domain. Let U be a set of p discrete values, where p is some small integer. A value $u \in U$ represents the likelihood that the concept c is appropriate for a specific situation in the domain. A high likelihood value implies that c is appropriate for the situation; a low value implies that c is not appropriate for the situation; and middle-range values imply various levels of uncertainty.

Let D be a finite set of n data d_i, $i=1,2,...,n$ in the domain of c. Let Q be a set of q truth values. Let v be a map, $v : D \rightarrow Q$, which assigns a value from Q to each $d \in D$. A datum $d \in D$ corresponds to an *assertion* about some *feature* in a specific situation in the domain, and $v(d)$ is the truth value of the assertion in a q-valued truth system Q. If some $d \in D$ asserts that feature x has a discrete value y in some situation for example, then in the case q is three, $v(d)$ may be True, False, or Unknown depending on whether the assertion is known to be true, false, or it is not known whether the assertion is true or false.

We assume that for a given c, specification of D and v for any situation in a class of situations is a necessary and sufficient condition to determine $u \in U$ for c. We define concept evaluation as a five-tuple $\langle c, U, D, v, f_{ce} \rangle$, where c, U, D, and v are as defined above, and f_{ce} is a function that takes D and v as inputs, and outputs a $u \in U$.

Concept evaluation may be performed with use of the strategy of concept matching, which is to match a conceptual structure with the description of the situation, and determine a likelihood that the concept is appropriate for the situation by the degree of the match. Concept matching may be viewed as an instantiation of f_{ce}. We now describe different implementations of concept matching, and derive the computational complexity of concept evaluation using these implementations. We assume an oracle for testing the value of one datum. We express the time complexity as the number of calls to the oracle, and the space complexity as the number of tests which have to be *encoded* in the knowledge-base.

2.2. Table Look-up

A first implementation of concept matching is table look-up. Knowledge is organized as a $q^n \times 2$ table. The first column of each row in the table contains a different entry from the q^n possible combinations of $v(d_i)$, $i=1,2,...,n$, and the second column contains the corresponding value of u. The control of information processing is row by row. Starting with the first row in the table, the entry in the first column of the row is matched with the input; if the match succeeds then the entry in the second column of the row is the output, else the entry in the first column of the next row is matched with the input, and so on. The time and space complexities T_{ce1} and S_{ce1} respectively, are given by

$$T_{ce1} = O(n.q^n)$$

$$S_{ce1} = O(n.q^n)$$

2.3. Tree Traversal

A second implementation of concept matching is tree traversal. Knowledge is organized as nodes in a discrimination tree. The top node in the tree corresponds to d_1 and has q branches coming out of it, one for each of the q possible values that d_1 may take. The branches lead to q different nodes, each of which corresponds to d_2 and has q branches coming out of it. This organization of knowledge is repeated until d_i, $i=1,2,...,n$ have been represented on the tree. Thus, there is one node at the first level, q nodes at the second, q^2 nodes at the third level, and so on. There are q^n branches coming out of the q^{n-1} nodes at the n^{th} level, each of which leads to a value of u. The control of information processing is top-down. Starting with the root node the branch that matches $v(d_1)$ in the input is taken, and the next node is reached, where the branch that matches $v(d_2)$ in the input is taken, and so on until $v(d_i)$, $i=1,2,..,n$ in the input have been matched. The match of $v(d_n)$ leads to the value of u which is the output. The time and space complexities T_{ce2} and S_{ce2} respectively, are given by

$$T_{ce2} = O(n)$$

$$S_{ce2} = O(q^n)$$

The space complexity is the sum of the geometric series of q^i from $i=0$ to $i=n-1$.

2.4. Structured Matching

We may view D and c as characterizing two different levels of abstraction in a given domain. Let us introduce $l-2$ intermediate levels of abstraction G_j, $j=1,2,...,l-2$ between the D and c levels. Let us consider n_1 features at the G_1 level, n_2 features at the G_2 level, and so on, with $n_1 \approx n/k$, $n_2 \approx n_1/k$, and so on, where n is the number of data in D, and k is some small constant greater than one. The number of intermediate levels of abstraction depends on k; $l \approx log_k(n)$. Let us assume that it is possible to form n_1 disjoint subsets of values $v(d_i)$, $i=1,2,...n$, with no more than k values in any subset, such that each such subset may be used to abstract the value of some feature at G_1. Let us assume also that it is possible to form n_2 disjoint subsets of the values of features at G_1, with no more than k features in any subset, such that each such subset may be used to abstract the value of some feature at G_2. This process may be repeated until the value u for c is abstracted from the values of features at the G_{l-2} level. We may call the hierarchy thus formed a feature hierarchy. The idea of hierarchical feature abstractions was first developed by Samuel in his work on game playing programs [Samuel, 1967].

We now describe a third implementation of concept matching called structured matching [Bylander and

Johnson, 1987]. Knowledge is organized in a feature hierarchy as above. At any level in the hierarchy a small number of strongly interrelated features are grouped, evaluated, and abstracted together to a higher level feature, and weakly interrelated features are evaluated and abstracted in different groups. The interactions between two groups of features at some level are taken into account at a higher level in the hierarchy. k represents the upper bound on the number of features that may be grouped together at any level in the hierarchy. The task of abstracting the value of a feature at some level from the values of features at the lower level in the hierarchy may be performed by a simple matcher that uses table look-up. Notice that in going from one level of abstraction to another it is not important if the range of likelihood values p, does not equal q. In the case q is three for instance, if the likelihood value for a feature is high then the truth value of the feature may be taken as True, if the likelihood value for a feature is low then the truth value may be False, and if the likelihood value is in middle range then the truth value of the feature may be Unknown.

The control of information processing is top-down. The information processing starts by invocation of the simple matcher corresponding to the concept c which is at the top node in the feature hierarchy. Since the simple matcher requires the values of the features input to it, it invokes the simple matchers at the next lower level in the hierarchy. The invocations of the simple matchers proceed downwards through the hierarchy, until the level of abstraction just above D level is reached. Since the values of input features at this level are known, the feature abstractions may begin. The feature abstractions flow upwards in the hierarchy until u is computed at the top node. Since each simple matcher in the hierarchy uses the strategy of table look-up with no more than k values the time and space complexities for each simple matcher are both $O(k.q^k)$, which is a constant. There is one simple matcher on the top level, k simple matchers on the second level, k^2 on the third level, and so on for the l levels. The time and space complexities are the sum of the geometric series of k^i from $i=1$ to $i=l-1$, where $l \approx log_k(n)$. Thus, the time and space complexities of concept matching using the strategy of structured matching T_{ces} and S_{ces} respectively, are given by

$$T_{ces} = O(n)$$

$$S_{ces} = O(n)$$

3. Complexity of Classification

3.1. Definitions

Let C be a finite set of m concepts, c_j, $j=1,2,..,m$ in a given domain. C is a taxonomy of m concepts in the domain. Let U_j, $j=1,2,...,m$ be m finite sets of p discrete values each. Let U be a set composed of sets U_j, $j=1,2,...m$. A value $u_j \in U_j$ represents the likelihood that the concept $c_j \in C$ for $j=1,2,...,m$, is present in a specific situation in the domain. Let D, Q, q, and v be defined as for concept evaluation. We note that the number of data $d \in D$ for classification would typically be much larger than for concept evaluation.

We assume that for a given C, specification of D and v for any situation in a class of situations is a necessary and sufficient condition to determine $u_j \in U_j$ for $c_j \in C$, $j=1,2,...,m$. The m concepts in C are equivalence classes of different subsets of D. We assume that the concepts in C are independent of one another; if c_{j1} and c_{j2} are two concepts in C, then the subset of D that may be classified into c_{j1} and c_{j2}, is the union of the two subsets of D that may be classified into c_{j1} and c_{j2} separately. We define classification as a five-tuple $\langle C, U, D, v, f_c \rangle$ where C, U, D, and v are as defined above, and f_c is a function that takes D and v as the input, and outputs $u_j \in U_j$, $j=1,2,...,m$.

The classification task may be performed with use of the strategy of concept activation, which is to activate concepts in the taxonomy for evaluation. Concept activation may be viewed as an instantiation of f_c. We now describe different implementations of concept activation, and derive the computational complexity of classification for these implementations. We assume an oracle for concept evaluation. We express the time complexity of classification as the number of calls to the oracle for concept evaluation, and the space complexity as the number of concepts in the knowledge-base.

3.2. Direct Activation

One implementation of concept activation is direct activation. Knowledge is distributed among the m conceptual structures, and the control of problem-solving is activation of concepts for evaluation, one by one. Concept evaluation is performed with a call to the oracle. The time and space complexities T_c and S_c respectively, are given by

$$T_c = O(m)$$

$$S_c = O(m)$$

3.3. Hierarchical Activation

Let C, U, D, and v be as before. Let C' be a finite set of m' concepts, c'_j, $j=1,2,...,m'$. Let U_j, $j=1,2,...,m'$ be m' finite sets of p discrete values each. Let U' be a set composed of sets U_j, $j=1,2,...m'$. A value $u_j \in U_j$ represents the likelihood that the concept $c_j \in C'$ is present in a specific situation in the domain. Let D' be a finite set of n' data, d_i, $i=1,2,...,n'$. Let $m' \geq m$, and $n' \geq n$. C', U', and D', are supersets of C, U, and

D, respectively. We redefine the task of classification as a five-tuple $\langle C', U', D', v, f'_c \rangle$, where C', U', and D', are as defined above, v is as defined earlier, and f'_c is a function that takes D' and v as inputs, and outputs $u_j \in U_j$, $j=1,2,...,m'$. Notice that since C', U', and D', are supersets of C, U, and D, respectively, f'_c entails f_c.

We now describe another implementation of concept activation, hierarchical activation. In hierarchical activation the m' concepts in C' are organized in a concept hierarchy such that the m leaf concepts in the hierarchy correspond to the m concepts in C. The value of m' depends on the branching factor, b, of the hierarchy; m' is directly proportional to m. The number of levels, l, is given by $l \approx log_b(m')$. Knowledge is organized in the concept hierarchy, and distributed among the m' conceptual structures. The control of problem solving is top-down. Starting with the concept at the root of the hierarchy, each concept when activated is evaluated with a call to the oracle; if the match succeeds then the concept is established as present in the situation, and its subconcepts are activated, else the concept is rejected as present in the situation, and its subconcepts are rejected as well. This may result in the pruning of the tree.

The space complexity $S'(c)$ is given by

$$S'(c) = O(m') = O(m)$$

In the worst case, each concept in the hierarchy may be activated. The time complexity in the worst case $T'(c1)$ is given by

$$T'(c1) = O(m') = O(m)$$

In the best case only l concepts may be activated for evaluation. The time complexity for the best case $T'(c2)$ is given by

$$T'(c2) = O(log_b(m')) = O(log_b(m))$$

4. Cognitive Issues in Classificatory Reasoning

In our framework for classificatory reasoning, the data that describe a situation in a given domain are allowed to take on only qualitative values, and the likelihood that a concept in the domain is appropriate for the situation is expressed as a discrete value. However, the use of numerical values and continuous functions *might* yield more precise results in some domains. Nevertheless, we use only discrete values and functions because intuitively they appear cognitively more plausible. For the construction of knowledge-using systems, in domains in which data appears in numerical form, the data values may be converted to qualitative form by preprocessing.

Further, in our framework, uncertainty in the data values and the likelihood values for concepts is handled locally rather than through a global uncertainty calculus. In performing the task of classification with use of hierarchical activation for instance, a high likelihood value for the appropriateness of a concept for a given situation is interpreted as presence of the concept in the situation, and the uncertainty is not propagated to the subconcepts. Again, the use of a global uncertainty calculus *might* yield more precise results in some domains; nevertheless, we handle uncertainty locally because intuitively that appears cognitively more plausible.

It appears obvious that the computational efficiency of performing a task is a precondition for the cognitive plausibility of the implementation of the strategy used to perform the task. We comment below on the cognitive plausibility of structured matching and hierarchical activation.

4.1. Structured Matching

Concept evaluation in general is computationally most efficiently performed with use of structured matching. However, in structured matching we had assumed that it was possible to form disjoint subsets of a set of features at some level in the feature hierarchy, such that each such subset may be used to abstract the value of a feature at the next higher level. This assumption is valid only in problem domains that Simon has called *nearly decomposable* [Simon, 81]. If such a decomposition was not possible then the feature hierarchy would be tangled. A tangled feature hierarchy may be untangled by including the same feature(s) in different feature groups.

For nearly decomposable domains the efficiency of performing concept evaluation using structured matching is due to two interrelated reasons. Firstly, a small number of interrelated features are grouped, evaluated and abstracted together at each level of abstraction in the feature hierarchy. This grouping together of a small number of interrelated features is analogous to the phenomenon of "chunking" in cognitive psychology. Secondly, an upper bound is imposed on the number of features allowed in a group. The imposition of a such an upper bound on the number of features is reminiscent of the notion of "short-term memory" in cognitive psychology.

4.2. Hierarchical Activation

For performing the classification task in general the use of hierarchical activation in the worst case is as efficient as, and in the best case is more efficient than, the use of direct activation. However, in describing hierarchical activation we had implicitly assumed that building of a non-trivial untangled concept hierarchy with m leaf concepts was indeed possible. Again, this assumption may be valid only in *nearly decomposable* domains. Branches in the concept

hierarchy that are tangled at some concept c may be untangled by including a copy of c in each tangled branch.

The computational efficiency of classification with use of hierarchical activation is due to the organization of knowledge in a hierarchy, which allows for pruning of the tree. A concept is activated for evaluation only if its parent concept has been established as being present in the given situation. Thus, a concept in general is evaluated in the *context* of its ancestor concepts. This use of context is cognitively appealing. Furthermore, if only an incomplete description of a situation were available, then direct concept activation might not lead to the establishment of any concept in the taxonomy. However, for the same incomplete description of the situation hierarchical activation might lead to the establishment of concepts at a level higher than the leaf level. This too is cognitively appealing.

5. Conclusions

We have described different implementations of the strategies of concept matching and concept activation in terms of the organization of knowledge and control of information processing that they use. We have shown that structured matching and hierarchical activation are computationally the most efficient implementations for performing the tasks of concept evaluation and classification, respectively. Further, we have suggested that structured matching and hierarchical activation may be cognitively the most plausible implementations as well.

Hierarchical activation and structured matching are computationally the most efficient implementations because they use organizations of knowledge that are most appropriate for the tasks of classification and concept evaluation, respectively. Organization of knowledge specific to its use is a central issue in knowledge-using reasoning in general. It is the organization of knowledge specific to its use from which the computational power to perform a task emerges. For each primitive type of knowledge-using reasoning there exists an organization of knowledge that is appropriate for it. Chandrasekaran has identified classification and concept evaluation as primitive types of knowledge-using reasoning [Chandrasekaran, 1986] High-level knowledge representation languages for classification using hierarchical activation [Bylander and Mittal, 1986], and concept evaluation using structured matching [Johnson and Josephson, 1986] have been developed.

Acknowledgments

We are deeply grateful to Tom Bylander and Dean Allemang for their many contributions to this paper. This research was supported by research grants from the Air Force Office of Scientific Research (AFOSR-86-719026), and the Defense Advanced Research Projects Agency, RADC (F30602-85-C-0010).

References

[Berliner and Ackley, 1982] Hans Berliner and David Ackley. "The QBKG System: Generating Explanations from a Non-Discrete Knowledge-Representation". In *Proceedings AAAI-82*, pages 213-216, 1982.

[Bylander and Johnson, 1987] Tom Bylander and Todd Johnson. "Structured Matching". Technical Report, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, February 1987.

[Bylander and Mittal, 1986] Tom Bylander and Sanjay Mittal. "CSRL: A Language for Classsificatory Problem-Solving and Uncertainty Handling". *AI Magazine*, 7(3):66-77, August 1986.

[Chandrasekaran, 1986] B. Chandrasekaran. "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design". *IEEE Expert*, 1(3):23-30, 1986.

[Clancey, 1985] William Clancey. "Heuristic Classification". *Artificial Intelligence*, 27(3): 289-350, 1985.

[Gomez and Chandrasekaran, 1984] Fernando Gomez and B. Chandrasekaran. "Knowledge Organization and Distribution for Medical Diagnosis". *Readings in Medical Artificial Intelligence: The First Decade*, Chapter 13, pages 320-339, William Clancey and Edward Shortliffe, editors, Addison-Wesley, Reading, Massachusetts, 1984.

[Johnson and Josephson, 1986] Todd Johnson and John Josephson. "HYPER: The Hypothesis Matcher Tool". Technical Report, Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, April 1986.

[Samuel, 67] Arthur Samuel. "Some Studies in Machine Learning Using the Game of Checkers II. Recent Progress". *IBM Journal of Research and Development*, 11(6):601-617, November 1967.

[Simon, 81] Herbert Simon. *Sciences of the Artificial*, Second edition, MIT Press, Cambridge, Massachusetts, 1981.