1993-01

A Memory Model for Concept Hierarchy Representation and Commonsense Reasoning

Soon, Hui-Shin Vivien

Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems

http://hdl.handle.net/2144/2009

Boston University
A MEMORY MODEL FOR CONCEPT HIERARCHY
REPRESENTATION AND COMMONSENSE REASONING

Hui-Shin Vivien Soon and Ah-Hwee Tan

January 1993

Technical Report CAS/CNS-93-032
A Memory Model for Concept Hierarchy Representation and Commonsense Reasoning

Hui-Shin Vivien Soon
College of Computer Science
Northeastern University, 161 Cullinane Hall
Boston, MA 02115, USA

Ah-Hwee Tan
Dept. of Cognitive & Neural Systems
Boston University, 111 Cummington St
Boston, MA 02215, USA

Abstract

Most associative memory models perform one level mapping between predefined sets of input and output patterns, and are unable to represent hierarchical knowledge. Complex AI systems allow hierarchical representation of concepts, but generally do not have learning capabilities. In this paper, a memory model is proposed which forms concept hierarchy by learning sample relations between concepts. All concepts are represented in a concept layer. Relations between a concept and its defining lower level concepts, are chunked as cognitive codes represented in a coding layer. By updating memory contents in the concept layer through code firing in the coding layer, the system is able to perform an important class of commonsense reasoning, namely recognition and inheritance.

1. Introduction: Concept Hierarchy Representation

Formulating concepts about objects is the basis of human intelligence. There are evidences indicating that human actually performs abstraction of knowledge and encodes concepts hierarchically; i.e. the meaning of a concept is usually built on a small number of simpler concepts, each in turns is defined using other concepts, and so on. As such, any memory model which maps a set of features directly to object concepts, are inherently limited in representing human knowledge. AI systems such as semantic networks (Fahlman, 1979; Touretzky, 1986) allow hierarchical representation of knowledge. However, they are generally hard-wired, and even in their connectionist implementation (Shastri, 1989), often do not include learning mechanism. One of the main difficulties in learning as mentioned by Feldman (1989), is to create new concepts and new memory structure dynamically. The problem is aggravated by the slow learning nature of most neural network algorithms.

In this paper, a memory model is proposed which represents its knowledge structure using supervised learning Adaptive Resonance Theory (ART) networks. For simplicity, a compressed version of fuzzy ARTMAP (Carpenter et al., 1992), called fuzzy Adaptive Resonance Associative Map (ARAM) (Tan, 1992) is adapted to implement concept hierarchy. The proposed system consists of a three-layer architecture: a microfeature layer, a concept layer and a coding layer (Figure 1). The microfeature layer contains sensory activities from visual, auditory and verbal inputs, which form the distributed representation of concepts. All concepts are represented in the concept layer in which a node is used to represent a concept. The activity value of a node indicates the degree of activation of the corresponding concept. Bidirectional conditionable links are used to connect the microfeature layer and the concept layer. By adaptive resonance theory (ART), distinct activity patterns across the microfeature layer can be organized into meaningful categories (concepts) in the concept layer. The top-most coding layer $P_2$ implements concept hierarchy by allocating a node to learn a relation between a concept and its definitive lower level concepts. Two copies of working memory fields $F^P_1$ and $F^F_1$ in the concept layer, connected to the coding layer by bidirectional conditionable links, are used for matching the conditions for code firing and for readout of code activation. Coding concept hierarchy is the theme of this paper and the details are described in the following section.
2. Coding of Concept Hierarchy

Note that a concept hierarchy is composed of a set of relations, each associates the meaning of a concept to its defining sub-concepts. The approach taken here is to learn each such relation using a cognitive code represented in the coding layer. An example is given in Figure 2. The learning procedure learns one relation at a time. A learning cycle involves code activation, code competition and template learning. It is important to note that the system does not merely remember each and every relation given (or else it will be of little interest). Competitive learning and template matching achieve code compression and abstraction of concept relations. The mathematics of the system dynamics is described as follows:

**Code Activation:** Let $A^a$ and $A^b$ be the activity vectors in the concept layer fields $F_2^a$ and $F_2^b$ respectively. Let $W^a_j$ and $W^b_j$ be the weight vectors associated with a node $j$ in the coding layer for coding concepts in $F_1^a$ and $F_1^b$ respectively. The activity of node $j$ is computed as follows:

$$T_j = \gamma \frac{|A^a \land W^a_j|}{|A^a| + |W^a_j|} + (1 - \gamma) \frac{|A^b \land W^b_j|}{|A^b| + |W^b_j|}$$

where $\alpha_a$ and $\alpha_b$ are small constants, $\gamma$ is a control parameter (typically set to 0.5 during learning), the fuzzy AND operation $\land$ is defined by $(x \land y) = \min(x, y)$ and the norm $|.|$ is defined by $|x| = \sum_{i=1}^{M} x_i$.

**Code Competition:** To ensure that only one code can be fired at a time, all $F_2$ nodes have to undergo a code competition process in which the eligibility for activation, $E_j$ of a node $j$ is evaluated as follows:

$$E_j = \begin{cases} 1 & \text{if } T_j = \max\{T_j: \text{for all node } J \text{ in } F_2\} \text{ and } T_j > \rho \\ 0 & \text{otherwise} \end{cases}$$

where $\rho$ is a vigilance parameter specifying the minimum match required for code firing.

**Template Learning:** Once a node $j$ is selected for firing, the weight vectors $W^a_j$ and $W^b_j$ are modified by the following learning rule:

$$W^F_j = (1 - \beta)W^F_j + \beta(A^F \land W^F_j) \quad \text{for } F = a, b$$

where $\beta \in [0, 1]$ is the learning rate.
3. Inferencing in Concept Hierarchy

Inferencing in the model is by evolution of memory contents in $F^a_1$ and $F^b_1$ through code firing in the coding layer $F_2$. A single inferencing cycle involves code activation, code competition (as in learning) and readout of activities. Readout into $F^a_1$ corresponds to propagation of activities down a concept hierarchy and readout into $F^b_1$ denotes upward flow of activities.

Activity Readout: After each code firing, the activities in $F^a_1$ and $F^b_1$ are updated as follows:

$$A^a = A^a \lor \mathcal{F}(\lambda \sum_j W^a_j T_j E_j) \quad \text{and} \quad A^b = A^b \lor \mathcal{F}(\sum_j W^b_j T_j E_j)$$

where the fuzzy OR operation $\lor$ is defined by $(x \lor y) \equiv \max(x, y)$, $\lambda \in (0, 1)$ is an attenuation parameter to prevent infinite propagation of activities down the concept hierarchy, and $\mathcal{F}$ is a threshold-linear function with an identity range in $[0, 1]$. The memory contents in $F^a_1$ and $F^b_1$ then update with each other as follows: $A^a = A^b = A^a \lor A^b$

To prevent perseverative firing of a code, a fired node is forbidden from getting fired again in a single inferencing task. In the following sections, the sample set of relations shown in Figure 2 is used to illustrate the model's recognition and inheritance processes. For simplicity, all computation assumes $\alpha_a = \alpha_b = 0$ and unit weights for all non-zero connections.

(A) Recognition by Bottom-up Activity Propagation

In a recognition task, a high level concept is identified given a set of lower level concepts. With $\gamma = 1$, the model is in recognition mode, in which the coding layer receives only inputs from $F^a_1$. Table 1 shows the transition of memory contents after three input cues (long-nose, big-ear and white) are given. After two inferencing cycles, royal-elephant is activated with activity 5/9.

<table>
<thead>
<tr>
<th>Time</th>
<th>long-nose</th>
<th>big-ear</th>
<th>gray</th>
<th>elephant</th>
<th>wear-clothes</th>
<th>white</th>
<th>royal-elephant</th>
<th>$T_1E_1$</th>
<th>$T_2E_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2\lambda/3</td>
<td>2/3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2/3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2/3</td>
<td>5\lambda/9</td>
<td>1</td>
<td>5/9</td>
<td>0</td>
<td>5/9</td>
</tr>
</tbody>
</table>

Table 1: Transition of system states in recognizing royal-elephant.
(B) Inheritance by Spreading Activation

Inheritance allows knowledge abstraction by inheriting properties of a concept from its lower level concepts. For example, the long-nose attribute of royal-elephant can be inferred from elephant. More complex situations require handling of exceptions, in which inheritance of a property can be overridden by other conflicting information. For example, royal-elephant cannot inherit gray from elephant as it is stated white. Two recall tasks are given below to illustrate property inheritance and exception handling. With $\gamma = 0$, the model is in recall mode. Table 2 shows the sequence of system states after royal-elephant is activated. The first code firing activates elephant, white and wear-clothes. The second code firing activates long-nose, big-ear, and gray with a smaller activation. By pre-organizing sets of conflicting concepts into competitive fields (Grossberg, 1973), cancellation of inheritance can be achieved where the activity of white quashes that of gray.

<table>
<thead>
<tr>
<th>Time</th>
<th>long-nose</th>
<th>big-ear</th>
<th>gray</th>
<th>elephant</th>
<th>wear-clothes</th>
<th>white</th>
<th>royal-elephant</th>
<th>$T_1E_1$</th>
<th>$T_2E_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>1</td>
<td>$\lambda$</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$\lambda^2$</td>
<td>$\lambda^2$</td>
<td>0</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Transition of system states after activating royal-elephant in a recall task.

With $\gamma = 0.5$, the model exhibits a more general form of activation spreading. Table 3 shows the system states after elephant is activated. The node coding elephant is first fired which activates long-nose, big-ear, and gray. With a low enough vigilance $p$, another code firing activates royal-elephant, white and wear-clothes with a smaller activation. In this case, the activity of gray quashes that of white.

<table>
<thead>
<tr>
<th>Time</th>
<th>long-nose</th>
<th>big-ear</th>
<th>gray</th>
<th>elephant</th>
<th>wear-clothes</th>
<th>white</th>
<th>royal-elephant</th>
<th>$T_1E_1$</th>
<th>$T_2E_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\lambda/2$</td>
<td>$\lambda/2$</td>
<td>$\lambda/2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$\lambda/2$</td>
<td>$\lambda/2$</td>
<td>$\lambda/2$</td>
<td>1</td>
<td>$\lambda/6$</td>
<td>$\lambda/6$</td>
<td>1/6</td>
<td>0</td>
<td>1/6</td>
</tr>
<tr>
<td>3</td>
<td>$\lambda/2$</td>
<td>$\lambda/2$</td>
<td>$\lambda/2$</td>
<td>1</td>
<td>$\lambda/6$</td>
<td>0</td>
<td>1/6</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Transition of system states after activating elephant.

Acknowledgement: The second author is on leave from the Institute of Systems Science, National University of Singapore.

References:


