A multiagent data warehousing (MADWH) and multiagent data mining (MADM) approach to brain modeling and neurofuzzy control

Wen-Ran Zhang a,*, Lulu Zhang b

a Department of Computer Science, College of Information Technology, Georgia Southern University, Statesboro, GA 30460-7997, USA
b Department of Computer Science, The University of Texas, Austin, TX 78705, USA

Received 15 April 2002; received in revised form 15 April 2003; accepted 8 May 2003

Abstract

Based on the hypothesis that the brain is a society of semiautonomous neural agents and full autonomy is the result of coordination of semiautonomous functionalities, a multiagent data warehousing (MADWH) and multiagent data mining (MADM) approach is presented for brain modeling and illustrated with robot motion control. An algorithm named Neighbor-Miner is proposed for MADWH and MADM. The algorithm is defined in an evolving dynamic environment with semiautonomous neurofuzzy agents. Instead of mining frequent itemsets from customer transactions, the new algorithm discovers new neurofuzzy agents and mines agent associations in first-order logic for coordination that was once considered impossible in traditional data mining. While the Apriori algorithm uses frequency as a priori threshold, Neighbor-Miner uses agent similarity as a priori knowledge. The concept of agent similarity leads to the notions of agent cuboid, orthogonal MADWH, and MADM. Based on agent similarities and action similarities, Neighbor-Miner is presented and illustrated with brain modeling for robot control. The novelty of a multiagent data warehouse lies in its ability to systematically combine neurofuzzy systems, multiagent systems, database systems, machine learning, data mining, information theory, neuroscience, decision, cognition, and control all together into a modern multidimensional information system architecture.
that is ideal for brain modeling of different animal species with manageable complexity. Although examples in robot control are used to illustrate the basic ideas, the new approach is generally suitable for data mining tasks where knowledge can be discovered collectively by a set of similar semiautonomous or autonomous agents from a geographically, geometrically, or timely distributed environment, especially in high-dimensional scientific and engineering data environments.

**Keywords:** Multiagent data warehousing and mining; Agent similarity and orthogonality; Neurofuzzy agents; Mining agent association in first-order logic; Brain modeling; Robot control

1. Introduction

Does a cat understand dynamic motion laws? Apparently not. Does it apply dynamic laws for motion control? Of course, any motion control on the earth has to follow dynamic laws. Can animals learn dynamic laws? If you answer “No”, then how can they apply the laws without learning them first? Logically, animals have to learn motion laws in body control. The question is not whether but how animals learn, represent and use them.

Based on the well-known theory in neuroscience that *the mind commands the body and the body constructs the mind*, it is hypothesized that repeated animal body movements build commonsense brain representations of dynamic motion laws that are in turn used in subconscious body control. This theory naturally leads to an orthogonal multiagent data warehouse (MADWH) brain architecture for commonsense brain modeling. In this paper, the agents are artificial neurofuzzy agents that exhibit semiautonomous functionalities. With this brain model, autonomy can be explained as a result of coordination of semiautonomous functionalities of the neurofuzzy agents, and learning can be defined as a multiagent data mining (MADM) process.

Neurofuzzy control has enjoyed great popularity in the last decade. A comprehensive coverage on fuzzy control and its applications can be found in [7]. Multiagent neurofuzzy control is introduced in [8–15] with a prototype system that accommodates a limited number of cerebellum agents for control. It is not clear, however, how large number of neurofuzzy semiautonomous agents form sophisticated brain systems for different autonomous functionalities, and how modern computer information systems can be developed for brain modeling of different agent species. This work aims at searching for answers to these tough questions through a novel MADWH and MADM approach. It brings findings and developments in neuroscience, cognitive science, information science, neurofuzzy systems, multiagent systems, data base
systems, decision, control, and data mining into a general framework of computer information system that resembles an agent society with many different kinds of brain structures as a basis for brain modeling for different applications.

In the follows, the concepts of agent similarities, agent action similarities, agent cuboid, orthogonality of agent cuboid, and multiagent data warehousing are introduced. An algorithm named Neighbor-Miner is proposed for orthogonal multiagent data warehousing. The algorithm can be considered a generalization of traditional data mining approaches from a static data environment with customer transactions [1–6] to a dynamic environment with agents and agent capabilities [8–15]. While the Apriori algorithm [1] cannot discover association rules in first-order predicate calculus, the new algorithm discovers agent association rules (or motion laws) in first-order logic. While a usual neurofuzzy system cannot accommodate OLAP and OLAM techniques, a MADWH enables neurofuzzy learning and OLAP/OLAM for brain modeling and knowledge discovery.

Section 2 introduces the concepts of agent similarities, action similarities, agent cuboid, orthogonality, and multiagent data warehousing. Section 3 presents Neighbor-Miner—a general-purpose algorithm for agent discovery and orthogonal MADWH development. Section 4 illustrates the basic concepts with a multiagent cerebrum/cerebellum system modeled as an orthogonal MADWH for neurofuzzy control. Section 5 is a brief conclusion.

2. Orthogonal multiagent data warehousing and multiagent data mining

2.1. Agent identification

In this work, agents are classified as autonomous or semiautonomous. An autonomous agent is meant to perform different functions with full autonomy. A semiautonomous agent performs one type of function with limited autonomy. Full autonomy is a result of coordination of semiautonomy [10]. While a human being is an autonomous agent, the left and right cerebellum systems, the left and right eyes, and the left and right ears of a human being can be considered semiautonomous to each other that perform, respectively, the same type of motor control, viewing, or hearing function collectively. Further, it can be hypothesized that a cerebellum system consists of semiautonomous agents for controlling long jumps, high jumps, forward jumps, backward jumps, gymnastic jumps, and other control tasks. This hypothesis leads to the multiagent cerebrum/cerebellum model for coordinated computational intelligence (CCI) as described in [10]. CCI provides a theoretical basis for a multiagent data warehousing and multiagent data mining approach to cerebrum/cerebellum modeling.
While autonomous agents can be identified by their multiple functionalities, semiautonomous agents can be identified by corner parameters for similar functions. A heuristic algorithm is proposed in [10] for semiautonomous agent identification. Is there an objective way to identify the corner parameters of semiautonomous agents? The answer is analytical mining or characterization [4].

Given a set of attributes of an agent relevant to the function performed by the agent, the attributes can be ranked based on attribute relevance to the function. In general, an agent can be described by agent properties and action properties. The most relevant agent properties should be identified as corner parameters.

2.2. Agent similarities and action similarities

Agents and their actions can be compared with similarity measures. Agent similarity and action similarity can be defined differently in different applications. Eq. (1) is a simple but useful definition.

\[ S = \frac{C}{N}, \]  

where \( N \) is the total count of parameters that define an agent or an action; and \( C \) is the total count of parameters on which two agents or two actions are considered the same or similar.

Apparently, dissimilarity can be defined as

\[ D = 1 - S = \frac{(N - C)}{N} = \frac{D_c}{N}, \]  

where \( D_c \) is the ratio of total number of parameters on which two agents are dissimilar.

Note that when a parameter has a continuous domain, the similarity and dissimilarity definitions have to use approximation to consider whether two agents or two actions can be considered the same or not on that dimension.

**Definition 1.** Two agents are similar, denoted \( A_1 \equiv A_2 \), if (1) they differ on at most one corner parameter; and (2) their actions or capabilities of actions are measured similar. The difference of the two agents on the differing corner dimension is said the (corner) distance of the two agents.

It can be observed from the real world that the following commonsense law holds

**Law 1.** Similar agents have similar functionalities.
We refer Law 1 as the *similarity law*. Can this law be discovered with data mining? Can this law be used in multiagent data warehousing and multiagent data mining? These questions are answered in the follows.

### 2.3. Agent cuboids vs. data cuboids

Based on agent identification, agent similarity, and function similarity, the author proposes the concepts of agent cuboids, multiagent data warehousing, and multiagent data mining as in the following:

While data cuboids organize relevant data sets into hypercube structures for business decision support, *agent cuboids* organize relevant or similar (cooperative or competitive) agents into hypercube structures for multiagent decision analysis and coordinated control. Agents in an agent cuboid are dynamic, which can perform collective learning/decision/control tasks once being called. An agent can be an AI agent (using symbolic representation), CI agent (using numerical representation), or coordinated CI (CCI) agent (using both approaches) [10]. They can be autonomous or semiautonomous. A cuboid of agents is *an agent community*. A *base cuboid* forms the lowest level community, an *apex cuboid* defines *an agent society*. *Agent oriented drill-down, roll-up, slice, dice, and pivot* support brain analysis and cognition at different levels of abstractions. *A base agent cuboid is said orthogonal* if every pair of neighbors are similar agents. *A non-apex agent cuboid is said orthogonal* if every pair of neighbors are similar cuboids.

### 2.4. Multiagent data warehousing

A *multiagent data warehouse (MADWH)* is a dynamically evolving data and knowledge structure that classifies and organizes autonomous or semiautonomous agents into communities and societies for coordinated learning and decision analysis. While data cube schema is one of the most popular schemas used in data warehousing [2], agent cube is meant to be a major schema for multiagent data warehousing.

A MADWH is essentially different from a traditional data warehouse. While a traditional data warehouse is a static data storage structure, a MADWH is an evolving agent world that emulates one or more social societies. A traditional data warehouse uses data cubes as a major data structure, a MADWH uses agent cuboids as a social organization. A traditional data warehouse is for supporting business decision-making, a MADWH is for learning, decision, control, and self-governance. A traditional data warehouse supports the mining of association rules as item sets, a MADWH supports multiagent
discovery of new agents and agent associations in first-order logic or agent laws for self-coordination and further discovery.

A MADWH is orthogonal if all its agent cuboids are orthogonal. A MADWH is partially orthogonal if some of its agent cuboids are orthogonal. Note that, agent-level similarity defines base-cuboid orthogonality, community-level similarity defines non-base cuboid orthogonality.

Although complete orthogonality might be difficult to achieve, it is undoubtedly a very desirable property for developing a MADWH. Orthogonality provides a basis for coordinated data outcropping and data mining in a geographically, geometrically, or timely distributed space. An orthogonal MADWH is a virtual agentization of a multidimensional space for organizing, allocating, and dispatching agents as coordinated actors or data miners. By data outcropping we mean to find leads for relevant data and knowledge sources from a distributed environment. By orthogonal multiagent data warehousing we mean to achieve partial or complete orthogonality for a MADWH.

2.5. Multiagent data mining with a MADWH

Based on agent communities and societies, traditional data mining [1,3] can be extended to multiagent data mining with an MADWH. Here multiagent data mining (MADM) is referred to as an iterative and collective learning effort by the agent communities in a MADWH. An agent can be a miner, decision maker, a controller, actor, or all in one that has local or partial learning and decision capabilities, can manage and use its local data and knowledge, and can cooperate or be coordinated with other agents for collective learning and decision-making. While traditional data mining is global mining, MADM is distributed mining where coordination is a key. The following MADM activities are identified:

1. identifying agents and agent communities;
2. adding and training new agents to an evolving MADWH;
3. dispatching agents to their post;
4. deploying knowledge and coordination protocols;
5. mining new knowledge including new coordination protocols;
6. Go to (1).

It should be remarked that, not only does multiagent data mining with a MADWH use the warehouse, it also develops the warehouse by identifying, adding, and training new agents based on new knowledge and new coordination protocols. Thus, an algorithm for multiagent data mining with a MADWH should naturally be an algorithm of multiagent coordination, which takes multiagent data warehousing into consideration.
3. An algorithm for multiagent data mining and orthogonal multiagent data warehousing

It can be observed that the popular Apriori algorithm [1] mines association rules from transactions by identifying frequent item sets. The Apriori algorithm cannot discover association rules in the form of first-order predicate calculus. In multiagent data mining, an algorithm deals with agents, agents’ functionalities, and agents’ capabilities. Therefore, transactions in the Apriori algorithm are replaced by agent actions in the new algorithm and itemsets are replaced by action sets. Moreover, in traditional data mining, the actors are often customers who make transactions. In the multiagent case, the actors are agents with certain functionalities determined by their actions or capabilities. While customers are not active components of a data warehouse, an artificial agent or miner can be an integral part of a MADWH. Therefore, the Apriori algorithm can be extended for discovering agent associations in first-order logic.

Note that, if the similarity law applies to a MADWH, some or all agent cuboids in the MADWH can be made orthogonal. Then we must have

**Law 2.** Given any pair of similar agents \( A_1 \cong A_2 \) with (corner) distance \( d \), a similar agent \( A_3 \) must exist between \( A_1 \) and \( A_2 \) that satisfies \( A_3 \cong \frac{A_1 + A_2}{2} \), \( A_3 \cong A_1 \) and \( A_3 \cong A_2 \). Formally, this law can be represented as the association rule in first-order predicate logic as

\[
\forall A_1, A_2, \{ \text{similar}(A_1, A_2) \Rightarrow \exists A_3 \{ A_3 \cong \frac{A_1 + A_2}{2} \wedge \text{Similar}(A_3, A_1) \\
\wedge \text{similar}(A_3, A_2) \} \}.
\] (2)

**Law 3.** Given any pair of similar agents \( A_1 \cong A_2 \) with (corner) distance \( d \), if a similar agent \( A_3 \) exists, \( A_3 \cong A_2 \), and \( A_3 \) is on the \( A_2 \) side with distance \( d \) to \( A_2 \) and distance \( 2d \) to \( A_1 \), it must satisfy \( A_3 \cong 2 \bullet A_2 - A_1 \). Formally, this law can be represented as the association rule in first-order predicate logic as

\[
\forall A_1, A_2, \{ \text{similar}(A_1, A_2) \wedge \text{similar}(A_2, A_3) \wedge \text{Distance}(A_2, A_3) = d \\
\wedge \text{Distance}(A_1, A_3) = 2d \} \Rightarrow A_3 \cong 2 \bullet A_2 - A_1.
\] (3)

Since Law 2 is a common sense law and Law 3 follows Law 2 directly, proofs for the two laws are omitted. We refer Law 2 as the **interpolation law** and law 3 as the **extrapolation law**. In Eqs. (2) and (3), The similar() predicate depends on a priori threshold measure of similarity as the Apriori algorithm [1] uses a priori frequency threshold for mining itemset association. Different thresholds can be used in different applications, however, the following commonsense law holds on agent distance \( d \) for any valid agent-oriented decomposition:
Law 4. Given any pair of similar agents $A_1 \equiv A_2$ with (corner) distance $d$, the smaller the distance $d$, the more similar the two agents. When $d$ approaches zero, the two agents are equal with the same functionality.

If an agent cuboid is orthogonal, agent interpolation and extrapolation can be used for data mining and orthogonal development of the MADWH. Then, multiagent data mining can be primarily a function of determining the applicability of the similarity laws, coordinating similar agents, and discovering new agents and laws for orthogonal MADWH development.

While agent extrapolation can be used for exploring an unknown data space, it must be based on at least two similar agents with a small enough corner distance $d$. If $d$ is not small enough, Eq. (3) will result in an extrapolated agent $A_3$ that is not similar to its neighbor. To resolve this problem, agent interpolation and fine-tuning can be used to an original agent cuboid before agent extrapolation.

It should be remarked that agent extrapolation, as an exploration process, does not guarantee success even when corner distance $d$ is very small. When an exploration failed, it discovers a dead end in the direction that is also very important knowledge in multiagent data mining. Based on agent interpolation and extrapolation, multiagent data mining with an orthogonal MADWH can be accomplished with the following algorithm:

**Neighbor-Miner: a MADM algorithm for orthogonal MADWH.** Let $a_0$ be a single action of an agent characterized by a pair of control and measure vectors $(V, M)$.

**Phase A. Identify corner parameters, primary measures, and build a base agent cuboid**

1. Generate different sets of actions on each dimension based on Eq. (1) by changing the control parameters one at a time starting with $a_0$.
2. Use a target and a contrast set [4, p. 199] of actions from Step (1) to rank the control parameters in vector $V$ (agent attributes) and action properties in vector $M$ (action measures) based on their information gain. The ranks lead to the identification of corner parameters in $V$ and primary action measures in $M$.
3. Use corner parameters to identify a number of corner agents and organize the agents into a base cuboid of the MADWH; use primary action measures to organize the data of each agent into local data cubes.

**Phase B. Local and global data mining**

While (not done) {
4. Let each new agent learn and refine its local knowledge with its action data (data in its local data cubes) using local learning techniques. (Typi-
cally, unsupervised learning techniques can be used for neural/fuzzy/genetic learning.) If all learning techniques failed for an agent, mark it as a DeadEnd, continue (to B). If all learning techniques failed for all corner agents in the base cuboid, exit with error message “inappropriate corner parameter and agent identification”. Restart from A.

(5) Let every new pair of neighbor agents try a set of $N$ similar actions selected randomly from a local data cube with a total of $T$ actions; use a similarity threshold that states “indifferent on at least $i$ primary measures among $p$ primary measures” or “different on at most $j$ primary measures among $p$ primary measures”, $p = i + j$, to determine whether each pair of actions are similar or not; count all similar pairs of actions as $Y$ (successful test) and dissimilar ones as $X$ (failures).

(6) Compute $S = N/T$ as the support for the similarity test. Compute $C = Y/(X + Y)$ as the confidence for the similarity of the two agents and the similarity law in the neighborhood.

(7) If $C$ is too low because the distance $d$ is too large, divide the cuboid into two by reducing $d$ using agent interpolation; continue (to B). Otherwise, mark that dimension with DeadEnd. Continue (to B).

(8) If $S$ and $C$ exceed the given thresholds for a neighborhood, Laws 1–3 are discovered and applicable in the neighborhood; if $S$ and $C$ exceed the given thresholds for all pairs in an agent cuboid, Laws 1–3 are discovered and applicable in every neighborhood of the cuboid.

(9) For any orthogonal neighborhood, Laws 2 and 3 can be interpreted as application-specific associations in Rule1 and Rule2, respectively, in first-order predicate logic as

\begin{align*}
\text{Rule 1:} & \quad \forall A_1, A_2, \{\text{action}(A_1, M, V_1) \land \text{action}(A_2, M, V_2) \land \text{neighbor}(A_1, A_2) \Rightarrow \exists A_3, \text{action}(A_3, M, (V_1 + V_2)/2) \land \text{neighbor}(A_1, A_3, A_2)\}; \\
\text{Rule 2:} & \quad \forall A_1, A_2, A_3, \{\text{action}(A_1, M, V_1) \land \text{action}(A_2, M, V_2) \land \text{neighbor}(A_1, A_2, A_3) \Rightarrow \text{action}(A_3, M, (V_1 + 2 \times V_2)/2)\}; \quad \text{OR} \\
& \quad \forall A_1, A_2, A_3, \{\text{action}(A_1, M, V_1) \land \text{action}(A_2, M, V_2) \land \text{neighbor}(A_3, A_1, A_2) \Rightarrow \text{action}(A_3, M, (2 \times V_1 + V_2)/2)\};
\end{align*}

(10) If any pair of neighbor agents is similar, use agent extrapolation in the desired directions or dimensions (based on the goal) for coordinated data outcropping and data mining. If goal is reached, done = true.

Note that the initial action $a_0$ is like the first step of a child learning to walk. With a first step, a child is on its way to explore the world of locomotion control. In robot locomotion, however, the first step has to be generated by a trial-and-error method. Also note that Neighbor-Miner can discover new agents based on agent-level similarity. Cuboid-level or community-level similarity is not discussed in this paper.
4. Computer-based brainstorming for neurofuzzy control

This section presents the Neighbor-Miner approach to multiagent neurofuzzy control and discuss the advantages of the new approach. Although, self-organization and reorganization are illustrated in [10], it is not clear, however, (1) how to combine neurofuzzy systems with modern information systems, e.g. a data warehouse, to organize, store, and process a large number of brain agents of different species; (2) how to combine the advantages of machine learning and OLAP/OLAM; and (3) how to model the brain systems of different species. This is illustrated in the follows with MADWH/MADM or a brainstorming approach.

4.1. Multiagent neurofuzzy control

A MAC (multiagent cerebellum) model is presented in [8–15] based on a CCI theory. The CCI theory assumes that a brain system consists of a number of cognitively identifiable semiautonomous cerebral/cerebellar agents that can be coordinated in learning, decision, and control. Therefore, full autonomy is a result of the coordination of semiautonomous agents.

The goal is to enable a simulated $N$-link uniped (whose motion is governed by a set of differential equations that has infinite number of inverse solutions) to learn gymnastic jumps. Each jump by an $N$-link uniped can be characterized with a $\langle V, M \rangle$ pair, where $V$ is a control vector and $M$ is a measure vector as defined in Table 1 for a 3-link and a 4-link uniped.

Note that in the 4-link (foot, lower leg, upper leg, and body) case $V$ has 10 dimensions and $M$ has seven dimensions. The angles $\theta_{11}$–$\theta_{42}$ in $V$ determine the take-off configuration of the robot; $T_{1}$–$T_{3}$ are torque applied to the three joints for taking off; $T_{4}$–$T_{6}$ are torque applied to the joints in flight to configure the robot for proper landing. The torques can be replaced with desired joint angles $\theta_{d1}$, $\theta_{d2}$, $\theta_{d3}$ for landings. $H$, $D$, and $A$ define the jump height, distance, and landing angle and the four angles $\theta_{L1}$, $\theta_{L2}$, $\theta_{L3}$, and $\theta_{L4}$ define the landing con-

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Control and measure parameters of an action</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-Link</td>
<td>$V = (\theta_1, \theta_2, \theta_3, T_1, T_2, T_3, T_4)$ or $V = (\theta_1, \theta_2, \theta_3, T_1, T_2, \theta_{d1}, \theta_{d2})$</td>
</tr>
<tr>
<td></td>
<td>$M = (H, D, A_w, \theta_{L1}, \theta_{L2}, \theta_{L3}) \equiv (H, D, A, LH, A_w, \theta_{L1}, \theta_{L3}), x, y = 1, 2, or 3, x \neq y$</td>
</tr>
<tr>
<td>4-Link</td>
<td>$V = (\theta_1, \theta_2, \theta_3, \theta_4, T_1, T_2, T_3, T_4, T_5, T_6)$ or $V = (\theta_1, \theta_2, \theta_3, \theta_4, T_1, T_2, T_3, \theta_{d1}, \theta_{d2}, \theta_{d3})$</td>
</tr>
<tr>
<td></td>
<td>$M = (H, D, A_w, \theta_{d1}, \theta_{d2}, \theta_{d3}, \theta_{d4}) \equiv (H, D, A, LH, A_w, \theta_{L1}, \theta_{L3}), x, y = 1, 2, 3, or 4, x \neq y$</td>
</tr>
</tbody>
</table>
figuration. The landing configuration can be equivalently determined by \((A; LH; \theta_{Lx}; \theta_{Ly})\) where \(A\) is landing angle, \(LH\) is landing mass center height, \(\theta_{Lx}, \theta_{Ly}\) are any two different link angles. A 3-link uniped has two joints and needs two take-off torques and two in-flight torques for a jump.

4.2. Neighbor-Miner steps (1)–(3): using information gain for corner agent identification and base cuboid construction

While the information gain method is well-known for classification, it can also be used for cognitively identifying corner agents for robot control. Steps (1)–(3) of the Neighbor-Miner algorithm is an analytical mining or characterization approach for agent identification based on attribute relevance analysis [4, p. 199]. Here, it is to find the most relevant attributes that can distinguish different jumps. For instance, backward and forward jumps are most distinguishable classes. There are many methods for assessing attribute relevance. Each has its bias. Although the information gain measure is biased towards attributes with many values, this bias can be removed when different domains are divided into the same number of ranges. For instance, angles and torques can both be fuzzified with a linguistic fuzzy set \{small, medium, large\}. With this preprocessing, the 10 parameters in \(V\) can be ranked based on their information gains when a group of forward jumps are used as a candidate relation of a target class and a group of backward jumps with similar landing measures are used as a candidate relation of a contrasting class. The ranking shows that the taking-off \(\theta\) angles have the highest information gains in vector \(V\). Similarly, \(H\), \(D\), and \(A\) are found having the highest information gains in vector \(M\). Thus the \(\theta\) angles are determined as corner parameters. \(H\), \(D\), and \(A\) are determined as primary measures for associative memory \((M-V\) vector pairs\) organization of a corner agent.

Although the result of the information gain method is the same as that of the heuristic method, the information gain technique provides an analytical mining approach for corner agent identification. It is objective rather than ad-hoc; it exhibits autonomous learning behavior rather than heuristic. Fig. 1 shows a set of identified corner agents and Fig. 2 shows a 4-D agent cuboid.

![Fig. 1. 16 corner agents (adapted from [10,12]).](image-url)
4.3. Neighbor-Miner step (4): local learning or data mining

Table 2 shows two link weights matrices of a trained 3-layer BP neural controller with error 0.000009. It is assumed that an autonomous agent coordinates many semiautonomous neurofuzzy (controller) agents in the MADWH. Whenever an agent is called from the warehouse, the link weights are assigned to the neural/fuzzy controller to generate a $V$ vector for a desired jump measure vector $M$.

4.4. Neighbor-Miner steps (5)–(9): mining dynamic motion laws as agent association rules in first-order logic based on agent similarity

Note that orthogonality is defined based on the similarity law (Law 1). Therefore, to determine the orthogonality of a base cuboid is actually to mine the similarity law from the agent community. We need to test the two conditions of similarity. The first condition has been met by selecting corner

![Fig. 2. A 4-D base cuboid with 16 corner agents (adapted from [10]).](image)

<table>
<thead>
<tr>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>Err</th>
<th>link weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>0.000009</td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Two link weights matrices of a 3-layer BP neural controller

$$
\begin{array}{cccccccc}
-3.380671 & -5.789867 & -0.803275 & -1.805584 & 3.367871 & -0.567861 & -2.064914 \\
4.225339 & 8.748863 & 0.977284 & 1.778801 & -4.920876 & 2.00594 & 1.970769 \\
0.740411 & 0.74155 & 0.709369 & 0.195054 & 0.843568 & 0.826496 & 0.622034 \\
-0.949556 & -1.602977 & -1.486067 & -1.475501 & -2.318124 & -0.778026 & -2.533893 \\
-0.968122 & -2.248199 & -0.549651 & -0.268073 & 1.282669 & 0.178415 & 0.500489 \\
-1.110398 & -3.399694 & 0.118515 & 0.746009 & 1.559476 & 0.44012 & 3.492346 \\
\end{array}
$$
parameters and construct agent cuboid. The second condition is to test the behavioral similarity or capability similarity of every pair of neighbor agents in the cuboid.

Five actions (jumps) by five corner agents of a 3-link uniped are listed in Table 3 to illustrate the basic idea. Evidently, agents \( A \) and \( B \) are similar based on their jumps because they differ on one corner parameter (\( \theta_1 \)) and they are able to make almost the same jump with different joint torques. Agent \( C \) and \( D \) are similar also. But \( C \) can make longer jumps with the same height. Agent \( D \) and \( E \) are also similar, Agent \( E \) can make an even longer jump. (Note: angles are in degrees and height and distance are in meter.) Thus, every pair of neighbor agents can be considered similar. Here only one pair of actions is selected. If it is selected from 100 actions, the support is 0.01. Since only one pair of actions is tested successfully, the confidence is 1.0.

As illustrated above, every pair of neighbor agents in the agent cuboids of Fig. 2 are tested as similar agents with high support and confidence measures. It is then can be concluded that both agent cuboids are orthogonal where the similarity laws are discovered. From Table 3, we can see that the similar action measures are different only on the distance dimension. The application-specific first-order association rules in this case can be determined as in Table 4. Interestingly, the two association rules are apparently dynamic motion laws. Such laws can be used as meta knowledge for further coordinated data mining.

It should be remarked that each neurofuzzy agent for the 3-link uniped in Table 1 has 14 dimensions, each agent for the 4-link uniped in Table 1 has 17 dimensions, and there could be many agent cuboids. The cognitive complexity

<table>
<thead>
<tr>
<th>( V )</th>
<th>( M )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta_1 )</td>
<td>( \theta_2 )</td>
</tr>
<tr>
<td>Agent A</td>
<td>170</td>
</tr>
<tr>
<td>Agent B</td>
<td>180</td>
</tr>
<tr>
<td>Agent C</td>
<td>170</td>
</tr>
<tr>
<td>Agent D</td>
<td>170</td>
</tr>
<tr>
<td>Agent E</td>
<td>170</td>
</tr>
</tbody>
</table>

Table 3

Similar actions by similar agents

Table 4

Mined dynamic motion laws as agent association rules

\( \forall A_1, A_2, \{ \text{jump}(A_1, H = x, D = \text{short}) \land \text{jump}(A_2, H = x, D = \text{long}) \land \text{neighbor}(A_1, A_2) \} \)

\( \Rightarrow \exists A_3, \text{jump}(A_3, H = x, D = \text{MediumLong}) \land \text{neighbor}(A_1, A_3, A_2) \} \);

\( \forall A_1, A_2, A_3, \{ \text{jump}(A_1, H = x, D = \text{short}) \land \text{action}(A_2, H = x, D = \text{MediumLong}) \land \text{neighbor}(A_1, A_2, A_3) \} \)

\( \Rightarrow \text{action}(A_3, H = x, D = \text{long}) \}; \) OR

\( \forall A_1, A_2, A_3, \{ \text{jump}(A_1, H = x, D = \text{MediumLong}) \land \text{jump}(A_2, H = x, D = \text{long}) \land \text{neighbor}(A_1, A_1, A_2) \) \)

\( \Rightarrow \text{action}(A_3, H = x, D = \text{short}) \} \)
is not manageable even with a usual data warehouse. For instance, it would be very difficult (if not impossible) to represent the data in Table 3 for visualization and decision using a table format without the multiagent approach and it is definitely impossible to mine the dynamic motion laws in first-order predicates as in Table 4 (Note: mining first-order association rules is usually considered impossible).

4.5. Neighbor-Miner steps (7) and (9): apply the similarity law for global mining

Similarity leads to agent interpolation and extrapolation in steps (7) and (9), respectively, in a global mining process. The MADWH approach provides an efficient and adequate platform for modeling a brain system in perform coordinated adventures. At the neural network level, interpolation and extrapolation result in weight matrices for a new neural net assuming the same neural architecture. The weight matrices can be used as initial weights for training the interpolated or extrapolated neural controller. This can reduce the training time dramatically compared with using random initial link weights. Given two similar BP neural agents \( A \) and \( B \) with neural weight matrices \( W_A \) and \( W_B \), respectively, based on the dynamic motion laws in Table 4 we have

\[
W_I \approx \frac{(W_A + W_B)}{2}, \quad \text{and} \\
W_E \approx 2W_A - W_B,
\]

where \( W_I \) is a weight matrix of an interpolated neural agent, and \( W_E \) is a weight matrix of an extrapolated neural agent assuming the same neural architecture.

Interpolation and extrapolation is learning by discovery. The learning speed of agent discovery is geometrical for locomotion control [10]. Therefore, the orthogonal neural agent representation of dynamic motion laws provides effective inverse dynamics for the second-order motion equations that govern the motion of an autonomous agent. It provides a brain structure that may well explain the phenomena that an animal can learn and apply dynamic motion laws without understanding them.

With the similarity law, agent extrapolation allocates new agents in the plausible directions. The implausible directions are marked with DeadEnd. A DeadEnd is also an important discovery. It helps redirecting the exploration toward the plausible direction. It emulates the process of following the leads of outcropping in mineral deposit exploration by a team of miners. Fig. 3 shows the exploration in the long jump direction with agent A, B, C, and D as illustrated in Table 3. It should be remarked that without a MADWH brain structure, such coordinated exploration in a multidimensional data environment is impossible.
4.6. Dynamic schema development in multiagent brain modeling and brainstorming

A corner agent’s associative memory is a data cube and each agent cuboid is actually a hyper-cube of corner agents. If each 4-link uniped has six 4-D kernels for (1) long jumps, (2) high jumps; (3) forward flip jumps; (4) backward flip jumps; (5) dancing; and (6) diving; there will be 96 corner agents in six kernel spaces that can be coordinated in motion control. If the six functions were tested under three different ground conditions with 10 different body weights and 10 different link lengths on three levels of gravity (e.g. for the earth, the moon, and the Mars), it will lead to 5400 (6^3 × 10^2) 4-D kernel spaces and 86400 (16 × 5400) corner agents. In this case we may say the control space is agentized into agent cubes.

Note that when the corner agents start to learn motion control they are faced with a large amount of control data generated constantly. For instance, if each of the 5400 kernels has successfully controlled 1000 different jumps over a period, there will be 5,400,000 different jumps. Each jump is a different \( (V, M) \) pair with a total of 17 dimensions. Evidently, the 5,400,000 actual jumps in 5400 4-D kernel spaces form a multiagent data warehouse.

While a 4-link uniped cerebellum can be a small multiagent data warehouse with semiautonomous agents, the brain systems of 1000 different unipeds (short, tall, 3-link, 4-link, \( N \)-link, heavy, light, long leg, short leg, long foot, short foot, etc.) can form a pretty big data warehouse even without mentioning biped and arm control yet. Evidently, the CCI approach need a MADWH for brain modeling of different species. Fig. 4(a)–(d) shows four levels of a growing star schema of a MADWH. Fig. 5 shows the dimensions of the MADWH. Fig. 6 shows a lattice of cuboids making up the society of a robot species, which is a 4-D cube for the dimensions species, robots, kernels, and (corner) agents. Further, each kernel has four corner dimensions and 16 corner agents for a
simple 4-link uniped, and each corner agent (neural, fuzzy, or associative memory) has 17 dimensions (see Table 2).

With the lattice as in Fig. 6, an example drill-down is to find the best neural agent of a robot for a specific jump from the base cuboid. An example roll-up operation is to summarize the properties of all robot species from the apex.
cuboid. A typical application of slice and/or dice is to focus on a 2-D or 3-D subarea of the brain in cognition analysis on certain dimensions. The unimaginable complexity involved is well-contained with a MADWH.

5. Conclusions

A MADWH and MADM approach to brain modeling and neurofuzzy control has been introduced. Concepts of agent similarity, agent cuboid, orthogonality, multiagent data warehousing (MADWH), and multiagent data mining (MADM) have been proposed. Based on agent similarity and action similarity, a multiagent data mining algorithm, Neighbor-Miner, has been presented for orthogonal multiagent data warehouse development. The algorithm is defined in an evolving dynamic environment with autonomous or semiautonomous agents as “customers” and agent actions as “transactions”. Different from the Apriori algorithm where frequency is used as a priori threshold for mining item associations, the new algorithm uses agent similarity as a priori threshold for discovering agent associations in first-order logic, which is usually considered impossible in traditional data mining.

The major advantage of a MADWH lies in its ability to systematically combine neurofuzzy agents, machine learning, data mining, information theory, neuroscience, cognitive science, and decision support all together into one modern information system architecture with manageable complexity that is ideal for brain modeling of different animal species. The findings provide a
natural explanation to the question “how animals learn, represent, and apply dynamic motion laws without understanding them”. Although examples in legged robot gymnastics are used to illustrate the basic ideas, the new approach is generally suitable for similar data mining tasks where knowledge can be discovered collectively by a set of semiautonomous or autonomous neurofuzzy agents from a geographically or geometrically distributed but relevant environment, especially in a high-dimensional scientific and engineering data environments such as weather data mining and multisensor data mining. Moreover, an agent does not have to be neural-based. It could be a member of any collective data “miner” group. With this relaxation, the general idea of a MADWH/MADM approach is applicable in a broad range of data mining tasks including web mining.

A few topics deserve further research efforts. First, new agent discovery can be extended for new community discovery. Secondly, agent association can be extended for discovering the behaviors and activity patterns of different agent species with OLAP/OLAM. Thirdly, video and audio information can be incorporated into a MADWH for developing a brain-like computer system with enhanced autonomy.

Acknowledgements

The authors acknowledge the reviewers of this paper for their constructive reviews. One reviewer proofread the manuscript and made detailed suggestions and corrections. Thanks also go to Dr. Paul Wang, editor-in-chief, for his valuable suggestions and encouragement.

References


