Adaptive Methodologies for Intelligent Agents

Ibrahim F. Imam

SRA International
4300 Fair Lakes Court, FS-5
Fairfax, VA 22033

Also Affiliate with the
Machine Learning and Inference Laboratory,
George Mason University
email: iimam@verdi.iisd.sra.com

ABSTRACT

The performance of an agent depends highly on its ability to adapt its methodologies to solve different problems, or to solve the same problem in different situations. Such adaptations may (or may not) change the external behavior of the agent. This paper introduces two adaptive methodologies that utilized by intelligent agents for improving their performance. In the first methodology, the agent adapts existing knowledge-base to fit different decision making situations. The agent restructure the knowledge-base when the user requests information or a decision that can not be directly determined from the knowledge-base. The agent uses a set of cost functions and other parameters to describe different decision making situations. A simulated air-travel database is used to illustrate the adaptive methodology. The second agent generates optimized plans for object recognition. The goal of the agent is to recognize visual objects in order to conduct the appropriate actions. The agent should have many tools or classifiers, each utilizes different parameters or recognizes different characteristics of the objects. Initially, the agent generates a recognition plan. Then, it iteratively adjusts the plan until it satisfies a set of criteria that maximize the recognition rate. A set of real-world hand gesture images is used to illustrate the adaptive methodology.

key words: intelligent agent, machine learning, decision tree, neural networks.

1. Introduction

An agent is a machine or a system that accomplishes something the user needed without knowing how the agent did it (Minsky, 1994). Most researchers define agents as systems that interact with the outside world. An agent usually has a set of tasks to perform for the user. These tasks define the scope and limitations of the agent. Intelligent agents utilize inferential or complex computational methodologies to accomplish their tasks. Usually, the methodologies adopted by the agent has no relationship to the user satisfaction. Only the external actions of the agent are recognized and may be evaluated by the user.

External actions of an intelligent agent are usually projections of interior commands determined by a system or an algorithm. Adaptation in intelligent agents can be categorized into three categories based on the relationship between the exterior and interior behaviors of the agent: 1) Internal Adaptation—where the interior systems of the agent are adaptive, but its external actions do not reflect any adaptive behavior (e.g., an agent may optimize its search algorithm every time it performs a task, but it produces the same output to the user); 2) External Adaptation—where the interior systems of the agent are not adaptive, but its external actions reflect adaptive behavior (e.g., a robot uses a user's feedback or runs a model to determine the next action); 3) Complete Adaptation—where the interior systems of the agent are adaptive and its external actions reflect adaptive behavior (e.g., a robot that follows a plan and update the plan based on its observations). Intelligent adaptive agents are defined, here, as systems that employ some inferential or complex computational methodologies for adapting a set of parameters, a knowledge-base, a representation

space, a problem solving technique, a course of actions, or other objects to successfully perform a given task. Intelligent adaptive agents can effectively improve many application systems for different domains including military or strategic planning, mechanical control, interactive multimedia, image and voice recognition, etc.

This paper introduces two adaptive methodologies for improving the performance of an agent. In both methods, the agent is a program that controls the input data, parameters, and sets of cost functions to the learning system AQDT-2. The agent has a set of criteria for solving a given task (e.g., solving the problem with the minimum number of classifiers, avoiding some combination of attributes, etc.). The agent starts with the default setting of the system, then it adjusts iteratively the data and/or the cost functions to reach a better solution. After each iteration, the agent analyzes the output of the system and stores the configuration of the best iteration.

The first methodology can be used by an agent to adapt a knowledge-base to fit different decision making situations. The agent generates a task that fits the decision making situation (the user's problem). Then, it adapts the representation and the content of its knowledge-base to provide the most suitable knowledge for solving the user's problem. Decision making situations or tasks are described by sets of parameters and cost functions. The second adaptive methodology can be utilized by an agent to learn optimized plans for object recognition. The methodology assumes that there are multiple systems (e.g., classifiers) for recognizing objects. A *plan* is a directed graph where nodes may contain actions and/or the systems or tools to perform the actions; arcs may represent different results of actions. A plan describes whether a classifier can be used for recognition, or for determining the best classifier for recognizing a given object. Each classifier was trained to recognize different aspects or characteristics of a set of objects.

Both methodologies can support agents of the first and the third classes of adaptive agents (i.e., agents utilize internal adaptation only or both internal and external adaptations). External adaptation can be achieved by mapping the results of each method into actions (e.g., open the door if the object recognized, etc.). The agents in both methods used the AQDT-2 system (Michalski & Imam, 1994) as a back-end system. An example of the first methodology was presented using simulated data of trips between Washington and Tokyo. Different scenarios were presented to illustrate how the agent adapt its knowledge-base to suit different decision making situations. The second method were illustrated by an example describing the problem of object recognition. Many classifiers were trained to recognize different characteristics of visual objects. The agent goal was to optimize the process of recognizing new objects.

2. Related Work

This section presents an overview of research on adaptive systems and intelligent agents. The research introduced in this section can be categorized into three categories: 1) restructuring knowledge-bases; 2) optimizing representation space; and 3) controlling multistrategy systems.

Brian Gaines (1994) introduced a method for transforming decision rules or decision trees into exceptional decision structures. The method builds an Exception Directed Acyclic Graph (EDAG) from a set of rules or decision trees. The algorithm starts by assigning either a rule, say R0, or a conclusion, say C0, or both to the root node. If the algorithm assigns a conclusion to the root node, it places that conclusion on a temporary conclusion list. The algorithm, then, generates a new child node and add to it a new rule, say R1. The method evaluates the new rule R1, if the rule satisfies the conclusion C0 in its parent node, the algorithm builds a new child and repeats the processes. Otherwise, if the rule R1 does not satisfy the conclusion C0, it changes the temporary memory with the new conclusion C1 which is satisfied by the rule R0. The same process is repeated until all rules that have common conditions with the rule at the root are evaluated. The method then creates a new child node from the root and repeat the process until all rules are evaluated. In the decision structure, nodes containing rules only represent common conditions to all of its children.

Another example of an agent that has capabilities to adapt its knowledge is CAP (Calendar APprentice) (Mitchell, et al, 1994). CAP assists the user in scheduling his/her calendar based on some background knowledge representing his/her own scheduling preferences. CAP is an intelligent agent that learns rules from a set of examples that describe previous meetings. The learning algorithm is applied to different sets of decision classes or features (e.g. duration of a meeting, location of a meeting, day of the week, and appropriate time for the meeting). CAP acquires factual knowledge about each new attendee that appears on the calendar and uses this knowledge later to control the learned rules.

Examples of other adaptive systems that optimize the representation space of a given problem include the AQ17-DCI (Bloedorn, et al, 1993), and others (e.g., Rendell & Seshu, 1990). These approaches are very useful when the original representation space is inadequate for learning. AQ17-DCI, for example, searches the space of possible logical and mathematical combinations of the original attributes, then it reformulates the given database using only those attributes which improves the representation space.

In Multistrategy learning, one of the main research issues is how and when a single strategy can be used to successfully solve a given problem. An example of an intelligent agent that searches for the best inferential process to apply is MOBAL (Morik, 1994). MOBAL is a model for balancing the cooperation between several learning algorithms for supporting the knowledge engineers in developing knowledge-based systems. Also, examples of adaptive systems that can modify their knowledge-base to accomplish different tasks include: 1) systems that change the representation and the content of the knowledge (e.g., the AQDT-2 system for learning task-oriented decision structure from decision rules—Michalski & Imam, 1994); 2) systems that improve the performance of the learned knowledge without changing the representation (e.g., Baffes & Mooney, 1993).

Brodley (1993) addressed the problem of automatically selecting one or more algorithms or models from among a set of learning algorithms that will produce the best accuracy for a given task. A model class selection (MCS) system is developed to learn a hybrid tree that classifies to determine the best model class for each node. MCS uses a set of heuristic rules that describe the biases of learning algorithms and learning representations to solve the given problem. These rules represent an information guide in generating the hybrid tree.

Another example is the Copycat agent (Mitchell, 1993). Copycat was used to discover new analogies in alphabetic letter strings. Copycat consists of different agents that compete with one another to find the strongest analogy. An agent may search for analogy to the string "ss" as the first and last letters are equal; another may search for similarity of the same string as a repetition as it appears in the string "aaxxsskk"; a third agent may search for an analogy where characters of that string are successors to surrounding characters such as in "pqrsst". Copycat can discover entirely unpredictable analogies.

3. The AQDT-2 System

The AQDT-2 system (Michalski & Imam, 1994) learns task-oriented decision trees or structures from decision rules or from examples. The AQDT's approach was motivated by the need to build a learning and discovery system which can not only generate and store knowledge, but also can effectively adapt this knowledge for use in different decision making situations. A decision tree/decision structure can be an effective tool for describing a decision making process, as long as all the required tests can be performed easily, and the decision-making situations it was designed for remain constant. Problems arise when these assumptions do not hold. For example, in some situations, measuring certain attributes may be difficult or costly, a set of attributes are not of interest to the decision making process, etc. In such situations it is desirable to reformulate the decision tree/structure so that the "inexpensive" attributes are evaluated first, and the "expensive" attributes are evaluated only if necessary. A restructuring may also be desirable if there is a significant change in the frequency of different decisions or attribute's values.

In the AQDT method a test is selected from an available set of tests based on a combination of three *utility functions*. These criteria are: 1) *disjointness*, which captures the effectiveness of the test in discriminating among decision rules for different decision classes; 2) *importance*, which determines the importance of a test in the rules; and 3) *value distribution*, which characterizes the distribution of the test importance over its of values. The AQDT-2 uses a set of cost functions and different parameters to represent different tasks.

The agents presented in this paper are systems that automatically adjust the cost functions of AQDT, set its parameters, or modify the knowledge-base to form a task. For example, the agent may adjust the cost of one or more attributes or values of an attribute, combines multiple values into one value, set different frequencies to the given decision classes, etc. The following simple example illustrates the AQDT-2 system. Suppose there are three decision classes, C1, C2 & C3, described by the ruleset shown in Figure 1. Suppose that these rules constitute the initial ruleset context. Table 1 presents information on these rules and the values of two of the four elementary criteria computed by AQDT-2 for all attributes. For each class, the row marked "Values" lists values occurring in the ruleset for this class. For evaluating the disjointness of an attribute, say **A**, each rule in the ruleset above that does not contain attribute **A** is characterized as having an additional condition [**A**= a v b ...], where a, b, ... are all legal values of **A.**

{**C1** <= [x1=2] & [x2=2] , **C1** <= [x1=3] & [x3=1 v 3] & [x4=1] }
{**C2** <= [x1=1 v 2] & [x2=3 v 4] , **C2** <= [x1=3] & [x3=1 v 2] & [x4=2] }
{**C3** <= [x1=1] & [x2=1] , **C3** <= [x1=4] & [x3=2 v 3] & [x4=3] }

Figure 1: Rules used for illustrating the algorithm.

Table 1: Determining values of the selection criteria for each attribute.

Class		Attributes			
		x1	x2	x3	x4
C1	Values	2, 3	1..4	1..3	1..3
	Class disjointness	3	0	0	0
C2	Values	1, 2, 3	1..4	1..3	1..3
	Class disjointness	3	0	0	0
C3	Values	1, 4	1..4	1..3	1..3
	Class disjointness	5	0	0	0
Attribute Disjointness		11	0	0	0
Attribute Dominance		12	6	6	6

The row "Class disjointness" specifies the class disjointness for each attribute. The attribute x1 has the highest disjointness (11), and is assigned to the root of the tree. From the rules shown in Figure 1, one can also determine disjoint groupings of attribute values. This is done as follows: 1) determine for each attribute the sets of values that the attribute takes in individual decision rules, and remove those value sets that subsume other value sets. The remaining value sets are assigned to branches stemming from the node marked by the given attribute. For example, x_1 has the following value sets in the individual decision rules: {2}, {3}, {1, 2}, {1}, and {4}. Value set {1, 2} is removed as it subsumes {2} and {1}. In this case, branches are assigned individual values of the domain of x_1. For attribute x2, the value sets are {1}, {2}, {3,4}, and {1, 2, 3, 4}. In this case, branches are assigned value sets: {1}, {2} and {3,4}.

Attribute x_1 ranks highest (as it has the highest disjointness), and is assigned to the root of the tree. Four branches are created each one is corresponding to one of x1 possible values. Since all rules containing [x1=4] belong to class C3, the branch marked by 4 is assigned a leaf node with a decision C3. .Rules containing other values of x1 belong to more than one class. This process is repeated for each subset of rules until the decision tree is completed. Figure 2a shows a

conventional decision tree, and Figure 2b a compact decision tree learned from these rules. For combinations of attribute values that do not lead to any leaf, the decision tree assigns no decision ("unknown" decision).

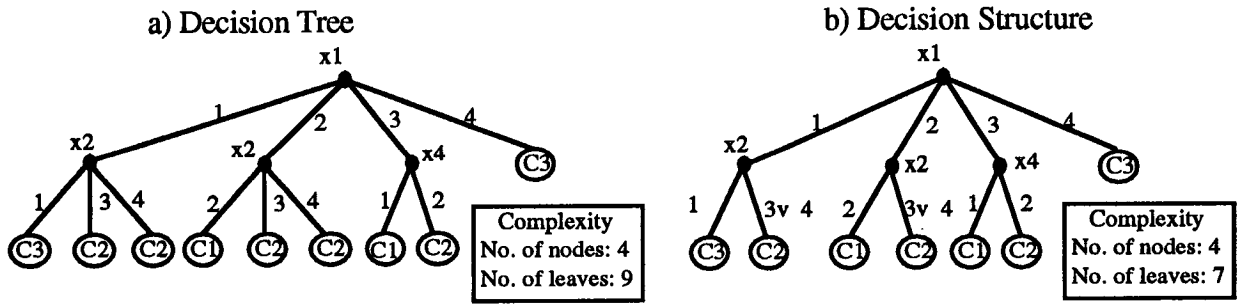


Figure 2: Decision tree generated for the rules in Figure 1.

4. Examples of Adaptive Agents

4.1. Agents That Can Adapt Their Knowledge-Base

This subsection presents an adaptive methodology for optimizing knowledge for different tasks. The agent uses the learning system AQDT-2 (Michalski & Imam, 1994) to learn task-oriented decision structures from rules. AQDT-2 utilizes a set of cost functions and other parameters to represent different tasks. The agent is a program that sets the cost functions and parameters of the AQDT-2 system, runs the system, and stores information about each run. The agent iteratively changes the cost functions until it obtains results that satisfy a set of criteria. The possible evaluation criteria may include one or combination of the following: 1) the maximum predictive accuracy, 2) the minimum number of nodes, 3) the minimum number of levels, etc. Figure 3 shows an illustration of the adaptation process of the proposed agent.

Input: A set of decision rules or examples and description of one or more tasks.
Output: A situation-oriented decision structure/tree for each decision making situation.

Step 1: For each decision making situation, the agent repeats steps 2 to 4.
Step 2: The agent evaluates all the attributes (using one of its criteria) and uses the cost functions to update the ranking of the attributes.
Step 3: The agent continues the process of learning decision structure/tree. It selects the top ranked attribute, assigns it to a node, creates branches equal to the number of its values, and divides the decision rules associated with this node into subsets each corresponds to one branch.
Step 4: If the agent reaches a situation where non of the inexpensive attributes can be used for further classifications of the decision rules at a certain node, the agent generates a leaf node with all possible decisions at this node. The agent can also determine a confidence probability for each decision or solution.

Figure 3: Description of the adaptive algorithm.

Example: This example presents an application of the adaptive methodology to a travel agent. The experiment uses a simulated travel database. The database is concerned with trips involving multiple flights between two destinations (e.g., Washington D.C. and Tokyo). The agent deals with six airline companies. Non of the six companies offers a direct-flight trip between the two destinations. Each successful trip consists of two or more flights. The agent can also sell tickets for any flight starts from the first destination to any intermediate destinations. The agent discovered that the customer's preferences can be summarized by 8 attributes, Table 2. All possible trips from Washington to Tokyo were split into two categories. The first category of trips labeled *preferred*

(many customers requested these routes), while the rest of the data was labeled *not-preferred*. Normally, the agent asks the customer a set of questions to determine the most appropriate flight. In this paper, the agent will attempt to discover interesting combinations that of interest to the customer in order to optimize the process of finding a preferred trip for the customer.

Table 2: The set of attributes used in the experiments ("i" : stands for the flight number).

Name	Attribute	Values					
		0	1	2	3	4	5
x_{11}	over-night-stay	yes	no				
x_{12}	waiting-time	x< 1	1<x<3	3<x<6	6<x>10	x>10	
x_{13}	over-night-flight	yes	no				
x_{14}	frequent-flight	yes	no				
x_{15}	entertainment	all	music	movie	games	none	
x_{16}	smoking	no	allowed				
x_{17}	airline-company	TWA	CAL	Pan-AM	United	KLM	JAL
x_{18}	No.-of-meals	one	two	more	none		

The data was generated using Mugglton's program (Michie, et al, 1994). User's preferences of a flight are described in terms of attribute describing: 1) if it requires over night stay, 2) the average waiting time during the whole trip, 3) if it is an over night flight, 4) if its frequent flight program is compatible with the customer's program, 5) what kind of entertainment the airline company provides, 6) if smoking is allowed, 7) the name of the airline company, and 8) the number of meals served during the trip. Table 2 describes these attributes and their possible values.
The agent first learns a classification of preferred and non-preferred trips. The agent uses the descriptions of the preferred trips to assist the user in selecting the best trip. Figure 4 shows a decision tree, obtained by AQDT-2, that classifying preferred and non-preferred trips. From this decision tree, one can observe that preferred trips depend highly on the airline company, and the frequent-flight-mileage. The agent offers any new customer the preferred trips. If the customer requires other options, then the agent presents him/her the non-preferred trips.

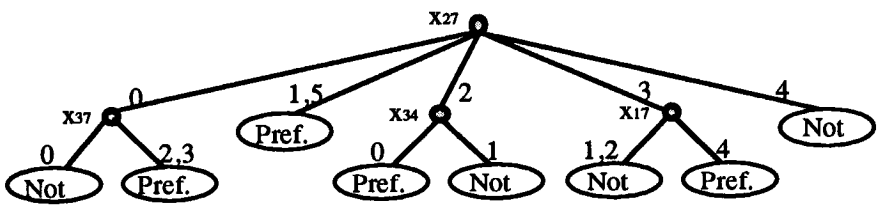
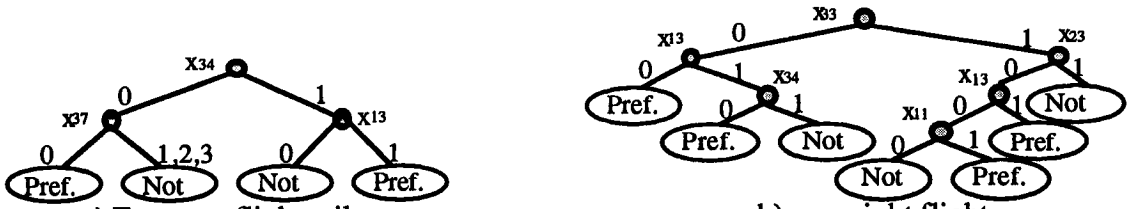


Figure 4: A decision structure obtained by AQDT-2 in its default setting.

Adaptation is required whenever the user has a specific request that can not be answered directly by the agent. For example, suppose that the customer was interested in the frequent flight mileage program. For this task, the agent assigns lower costs to all attributes representing the frequent flight mileage in all flights, and runs the AQDT-2 system. Figure 5-a shows a decision tree obtained by AQDT-2 for the given task. The agent's best choice is to propose the customer a trip consisting of three flights, where either the third flight should be on TWA, or the first flight is not an overnight flight (e.g., short flight or one starts early in the morning).
Another example of an adaptive task is when the user requires an overnight trip. In this case, the agent assigns lower costs for all overnight flights. The agent runs AQDT-2 with the new task. AQDT-2 learns the decision tree shown in Figure 5-b. Parts of this decision tree show classifications of all trips which include an overnight flight. All such trips consist of three flights. The same Figure shows also that there are four preferred ways for a trip with an overnight flight, two of which have two overnight flights.



a) Frequent-flight mileage b) overnight flight

Figure 5: Examples of adaptive situations.

A more complicated scenario would be, For example, if the customer asks for United airlines in as many flights as possible. The agent defines lower costs for all attributes representing the airline company, and assigns lower cost for the value "United" of these attributes (i.e., reduces the cost of the value 3 of attribute x_{17}). The agent groups all values representing other airline companies into one value "~". The agent runs the AQDT-2 system with this new task to obtain the decision tree in Figure 6. Two preferences were determined by the agent. The first has three trips and the third trip is a United flight. The second has four flights and requires staying over night. Note that the third preferred node may mean that there are other trips where the first or the second flight are served by United airlines. To obtain more details about these trips, the agent is supposed to set the cost of both attributes x_{37}, x_{47} slightly higher and runs the AQDT-2 system to obtain a decision tree for the new task.

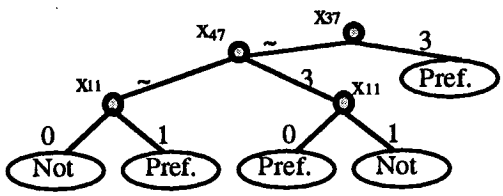


Figure 6: Decision tree describing user's preference in United flights.

4.2. Identification Agents

Another interesting problem that can benefit from the use of an adaptive agent is that of object identification or recognition problem. Object recognition systems are widely used by federal investigators to identify finger prints or facial images, companies to recognize identification cards and employees codes, by medical labs to diagnose diseases, by robots to identify known objects or surrounding environment, etc. Usually, the results of the recognition process are used to form a set of decisions or actions. Imam and Gutta (1996) proposed an approach to improve the recognition of visual objects using adaptive methodology. This rest of this section presents how an agent may use such approach for performing different tasks (e.g., security tasks—to allow the user to enter when match is occurred, etc.).

In order to justify the goals of the agent, this section will introduce a brief description of the object recognition approach proposed by Imam and Gutta (1996). To illustrate the approach, assume that an agent has a library of classifiers to recognize a set of objects. Each classifier can recognize the objects based on different set of characteristics. The approach is concerned with determining which classifier the agent should choose to obtain accurate recognition of a given object. In the approach, the agent is not allowed to acquire any information about the testing object, but it can use any number of classifiers to recognize that object. The agent learns an optimized plan to recognize any given object. The adaptation of the agent is done during the process of generating the optimized plan. The plan should recognize all objects using the minimum number of classifiers. According to

the result of the recognition process, the agent is supposed to take an action (e.g., allow access to some facilities, allow passage, etc.).
The agent uses the AQDT-2 program to create a complete plan for recognizing new objects. Two actions can be taken at each step of the plan, either *recognize* or *select* another classifier. The plan describes a sequence of actions based on the maximum value given by the classifier to a decision class. Whenever the action is "recognize", the agent assigns the decision class with the maximum value to the tested object. The agent adapts the cost functions and the different parameters to obtain the optimal plan. Figure 7 shows a brief description of the adaptive methodology of the agent. The algorithm stops after testing all intermediate nodes. This algorithm is not concerned with building the decision tree. However, it controls the process of learning decision trees.

Input: A set of training examples (records of recognition) described by a set of attributes A.
Output: A tree shows a set of optimized plans of classifiers needed for object recognition.

Step 1: Quantize all attribute values in the records of recognition.
Step 2: Specify the learning task for the agent (in this case, the decision structure should have minimum number of classifiers and minimum number of levels).
(*) For each attribute in the set A (i.e., classifier), repeat steps 3 and 4.
Step 3: Set the cost for that attribute lower than all other attributes in the set A, and use AQDT-2 to learn decision tree.
Step 4: Save the attribute name, the number of nodes, and number of levels. Assign similar cost to all attributes in set A.
Step 5: The attribute produced the minimum number of nodes and minimum number of levels, say C, is permanently assigned the lowest possible cost among the attributes in set A.
Remove C from A.
For each branch stemming from the node C, repeat steps 6 to 8.
Step 6: Assign a copy of the attribute set A to each branch.
Step 7: If the node connected to the branch is a leaf node, skip step 8.
Step 8: If the branch is connected to a non-leaf node, go to (*).

Figure 7: Description of the adaptive algorithm used by the agent.

Example: This example was copied from the work by Imam and Gutta (1996). The results were altered to fit the scope of this paper. The methodology was applied on a domain of hand gestures. A total of 150 gestures representing 15 different gestures were obtained. Three different testing combinations, each consisting of 9 cycles of two cross-fold method were used to test the approach. In each combination, five different gestures were considered as one class of gestures (number 11) to increase the complexity of the experiment.
The AQDT-2 program was modified to allow all nodes to contain actions in addition to the tested attributes. The agent generates a table of actions with their possible places. For example, in this case there are two actions "Select" and "Recognize". The "Select" action is assigned to any internal node. The "Recognize" action is assigned to leaf nodes only. The goal of the adaptive process is to maximize the recognition rate of objects using the minimum number of classifiers. Figure 8 shows a portion of the complete set of plans obtained by the agent.

5. Summary and Conclusion

The paper introduced a preliminary framework for the development of intelligent adaptive agents. In this framework, intelligent adaptive agents were defined as systems or machines that employ some inferential or complex computational methodologies for modifying or changing a set of control parameters, a knowledge-base, a plan, a representation space, a problem solving technique, a course of actions (for the same agent or for other agents), or other objects to successfully

accomplish a given task. Usually, the results of any task are represented by external actions (i.e., can be seen or captured by the user). The framework clearly distinguishes between the external actions of an agent and the problem solving methodologies used "internally" by the agent to determine these external actions.

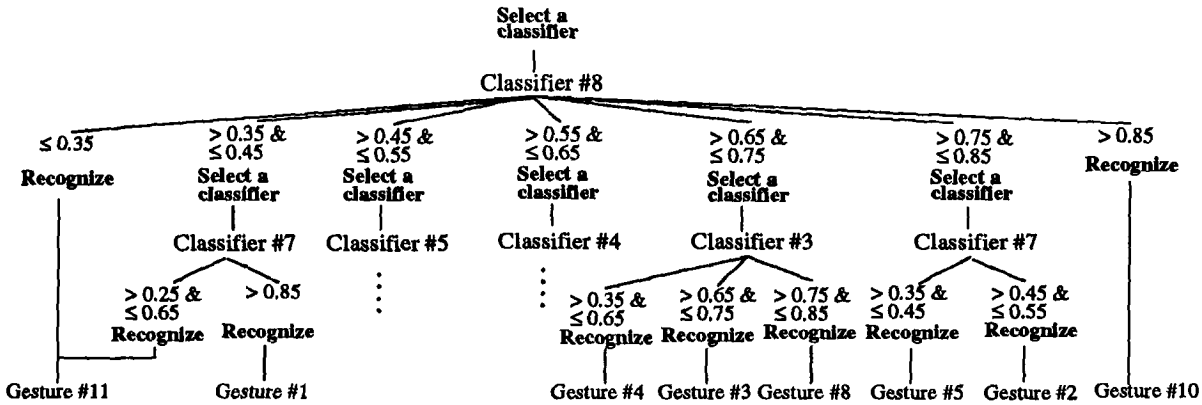


Figure 8: Portion of a complete plan for recognizing visual objects.

Intelligent adaptive agents were categorized, in this framework, into three groups based on the relationship between the external and internal behaviors of the agent. These categories are: 1) Internal Adaptation—where the interior systems of the agent are adaptive, but its external actions do not reflect any adaptive behavior (e.g., an agent optimizes its search algorithm every time it performs a task, but it produces the same output to the user); 2) External Adaptation—where the interior systems of the agent are not adaptive, but its external actions reflect adaptive behavior (e.g., a robot uses a user's feedback or runs a model to determine the next action); 3) Complete Adaptation—where the interior systems of the agent are adaptive and its external actions reflect adaptive behavior (e.g., a robot that follows a plan and updates the plan based on its observations). Adaptive behavior of an agent refers to the agent's ability to accomplish: different tasks within the scope of the agent functionalities, similar tasks within different environments, similar tasks within similar environment but using different problem-solving methodology, etc.

The paper presented two adaptive methodologies that can support either the first or the third categories of adaptive agents. The agent in both cases is a program controlling one or more other systems. The agent utilizes a set of parameters, cost functions, and other data structures to perform the adaptive process. The first agent, *a knowledge-optimizer agent*, adapts existing knowledge to fit different decision making situations. The agent changes repeatedly the cost functions to simulate different decision making situations. For each situation, the agent runs the AQDT-2 program and stores information about the results. The agent uses different criteria to evaluate these results to determine the best solution. The output of such methodology can be mapped to different external actions (e.g., reserve a ticket on flight number 534 on TWA, print a hotel accommodation form for one night) and/or other internal actions (e.g., ask the hotel agency to reserve a one night for the current customer).

The second agent adapts plans for recognizing visual object. It generates dynamic plans for utilizing a set of systems or classifiers for recognizing visual objects. The goal of the agent is to determine the best classifier for recognizing a given object without knowing any information about that object. The agent uses also a set of cost functions and data structures to optimize the plans. Plans generated by the agent describe different phases of the process of recognizing objects. This methodology could be applied to many applications and the recognition results can be mapped into many different external actions (e.g., security actions, access to banking accounts, sign language, etc.)

ACKNOWLEDGMENT

The author thanks Srinivas Gutta and Mark Maloof for careful review of the paper. The author would also like to thank Srinivas Gutta for running some of the experiments presented in this paper.

Most of this work conducted while the author was associated with the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's activities are supported in part by the Advanced Research Projects Agency under grant No. N00014-91-J-1854 administered by the Office of Naval Research, in part by the Advanced Research Projects Agency under grants F49620-92-J-0549 and F49620-95-1-0462 administered by the Air Force Office of Scientific Research, in part by the Office of Naval Research under grant N00014-91-J-1351, and in part by the National Science Foundation under grants DMI-9496192 and IRI-9020266.

REFERENCES

Baffes, P.T., and Mooney, R.J., "Symbolic Revision of Theories with M-of-N Rules", *Proceedings of the Second International Workshop on Multistrategy Learning*, pp. 69-75, Harpers Ferry, WV, May 26-29, 1993.

Bloedorn, E., Wnek, J., Michalski, R.S., and Kaufman, K., "AQ17: A Multistrategy Learning System: The Method and User's Guide", Report of Machine Learning and Inference Labratory, MLI-93-12, Center for AI, George Mason University, 1993.

Brodley, C.E., "Addressing the Selection Superiority Problem: Automatic Algorithm/Model Class Selection", *Proceedings of the Tenth International Conference on Machine Learning*, pp. 17-24, 1993.

Gaines, B., "Exception DAGS as Knowledge Structures", *Proceedings of the AAAI International Workshop on Knowledge Discovery in Databases*, pp. 13-24, Seattle, WA, 1994.

Imam, I.F., and Gutta, S.V., "A Hybrid Learning Approach for Better Recognition of Visual Objects", Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR, 1996. (To appear) ***http://www.mli.gmu.edu/~iimam/papers/AAAI96.ps***

Michalski, R.S., and Imam, I.F., "Learning Problem-Optimized Decision Trees from Decision Rules: The AQDT-2 System", *Lecture Notes in Artificial Intelligence*, No. 869, Ras, Z.W., and Zemankova, M, (Eds.), pp. 416-426, Springer Verlag, 1994. ***http://www.mli.gmu.edu/~iimam/papers/ISMIS94.ps***

Michie, D., Muggleton, S., Page, D. and Srinivasan, A., "International East-West Challenge", Oxford University, UK, 1994.

Minsky, M., "A Conversation with Marvin Minsky about Agents", Communications of the ACM, Vol. 37, No. 7, pp. 23-29, July, 1994.

Mitchell, M., *Analogy-Making as Perception*, MIT Press, Cambridge, Mass., 1993.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., and Zabowski, D., "Experience with a Learning Personal Assistant", *Communications of the ACM*, Vol. 37, No. 7, pp. 81-91, July, 1994.

Morik, K., "Balanced Cooperative Modeling" *in Machine Learning: A Multistrategy Approach* Vol. IV, Michalski, R.S. & Tecuci, G. (Eds.), pp. 259-318, Morgan Kaufmann Pubs., San Francisco, CA, 1994.

Rendell, L., and Seshu, R., "Learning Hard Concept through Constructive Induction: Framework and Rationale", *Journal of Computational Intelligence*, No. 6, 1990