Algorithm of Nested Clustering for Unsupervised Learning

J. Albus, A. Lacaze, A. Meystel[1]

Intelligent Systems Division NIST

United States Department of Commerce

albus@cme.nist.gov

Abstract

Autonomous learning in the architectures of intelligent control requires special procedures performed upon acquired knowledge. This affects the structure of World Representation and it is intimately linked with mechanisms of behavior generation. This paper illuminates algorithms of autonomous learning performed via nested clustering which is goal driven and exercises simulation of decision making process.

I. Introduction: Learning for Control

NIST has developed a reference model architecture RCS/NASREM for control of machines and systems in which the major features of intelligence are reproduced [1]. This system allows for substantial reduction of complexity due to its multiresolutional organization, and it provides flexible and adaptive decisions using a tool of generalization (including GFACS: grouping, focusing attention, and combinatorial search) [2]. The efficiency and convergence of control has been proven for robotic applications [3]. First efforts in equipping of this system with learning capabilities demonstrates compatibility of the RCS/NASREM concept with algorithms of hierarchical generalization [4]. In this paper, a novel algorithm of learning is explored based upon a concept of goal-driven nested clustering which is more efficient than existing algorithms of multidimensional clustering, and uses simulation of decision making processes as a part of consecutive operation. The algorithm organizes all incoming information into "eventgrams", builds statistical clusters and interprets them. In our effort we were driven by a desire to use the same algorithm for early learning processes. Thus, the results of the nested clustering actually build the World Model.

[1]On Sabbatical, from Drexel University, Philadelphia, PA 19104

II. Rule as a Representation of Goal-oriented Experiences

Learning can be defined as the acquisition of new concepts and the organization of these concepts into a control structure suitable for achieving the goal. Concept is defined recursively as a cluster or group of higher resolution concepts. At the highest resolution, the role of concepts is played by the primary cause-effect relationships (CER). Concepts are tools of organizing knowledge as to reduce the complexity of dealing with high resolution primary CER. All incoming information characterizes the readings of the available sensors and the commands submitted to the actuators. Thus, the experiences can be considered the CER since they are organized in the form of strings "previous situation + command generated $\Longrightarrow$ subsequent situation". One of the major principles that we use in this effort is our concept of selection of CER for the subsequent clustering. We evaluate "goodness" of CER depending on the "goal" at hand, and then we cluster only those CER which goodness exceeds some particular threshold. The reason for using this principle is our desire to learn how to achieve the goal. We contemplate a separate effort directed towards learning from "bad" experiences. We believe that this effort will not necessarily be required because by selecting the "good" experiences we learn implicitly the space of negative results.

Rules are CER formulated for the given goal in a given situation. A Rule is a statement which is obtained statistically from multiplicity of experiences formulated as CER. It is a statement about action to be done invariantly inside of the situation class. This statement turns out to be correct in every point of this particular precondition.

The existing classification algorithms are based upon two main strategies: based upon closeness or based upon sparseness (or separation) among the data. Some of the approaches in the literature of creation of classes from examples are: Feigenbaum's

EPAM [5], Lebowitz's UNIMEM [6, 7] , Smyth's ITRULE [8] Fisher's COBWEB [9, 10], Gennari's CLASSIT [11], Sammuts's MARVIN [12], Michalski's INDUCE [13], Langley's BACON [14], Quinlan's ID3 [15].

There exists a terminology confusion in the term "classification", some authors use it as the arrangement or sorting of elements in pre-specified classes; while others use it as the creation of classes or groups given a set of elements. We shall use the second definition when referring to the term. Our approach is different because we are looking for a hierarchical structure of rules of execution, instead of developing a decision tree for classification task [16]. When Quinlan [16] refers to search in the decision tree; he means that he is searching to see in which class the specimens in hand can be classified. In this paper we are looking for a recommendation about a behavior that should be applied in the situation at hand.

Sometimes the problem of creating rules of execution is considered a subproblem of the classifier group of algorithms. We see the creation of rules of execution as a set of problems that the classifying algorithms do not address, like the creation of goals and the recursive use of the generalization algorithm. The experience of the human controller are usually expressed as some linguistic "IF-THEN" rules that state in what situation(s) which action(s) should be taken [17].

In [17], although (fuzzy) rules are created, the discretization of the input and output space is prespecified by the designer, so input and output concepts are already defined leaving the learning algorithm the task of matching the input-output pairs to create a look-up table system where no new concepts can be created. Our approach is different in the sense that we create the situation and action concepts before doing the input output matching.

In [18], we find a genetic rule learning algorithm. In this algorithm each rule is a gene, these genes are crossed using a variety of genetic operators swapping parts of the genes or adding new parts to some other. A fitness function executes and selects the rules that reflect the desired performance for the task, and these new generation of rules goes again through the same procedure.

In [19] the learning of control takes a different approach. It is a reward-penalty reinforcement stochastic learning automaton which learns by changing a set for weights by which the inputs are linearly multiplied in order to achieve different outputs. The convergence of these weights into a set which will maximize a certain criterion is the learning component in this algorithm.

Salganicoff in [20] uses Quinlan's ID3 and IE [21] to classify the experiences into rules of action.

The pitfalls of these current algorithms can be summarized in the following points:

1. Only a small subset of them address the issue of learning from "valued experiences" where there are good and bad experiences.
2. Most of the authors learn declarative knowledge, while we are interested in procedural-causal knowledge.
3. The existing algorithms tend to build concepts of a resolution implicitly assigned. We are interested in a technique that will make a multiresolutional (MR) structure. We will see how a recursive use of our clustering algorithm will create an MR structure.
4. Most of the existing techniques jump into a particular level of development and complexity. We are interested in early learning so instead of a mature clustering algorithm we rather have a simpler (and computationally advantageous) algorithm that can evolve.
5. Most of the existing algorithm pre-assign the number of classes that will be found. We do not know and cannot predict the amount of clusters that will be present in the experiences, because we do not want to give this knowledge to the learning algorithm.

III. Nested Clustering or Recursive Algorithm of Generalization

In this section we will address the algorithm of generalization and we will explain how this algorithm can be applied in a nested manner in order to create a goal decomposition tree. The algorithm of generalization takes as input a goal and the database of experiences; and it processes these experiences until it comes out with a set of hypotheses. These hypotheses are then stored in the database of hypotheses. Older hypotheses influence the generalization algorithm in the sense that they pass a goal to the algorithm of generalization. It is composed of the following steps:

1. The whole database of experiences is separated in two classes: admissible and non-admissible. Admissible experiences are good experiences with respect to the given goal. There are different ways of choosing these "good" experiences, these procedures will be explained in detail.
2. The representation of each experience is Enhanced. Enhancing means that new labels are included in the experiences, these new labels are combinations among sensors values, among actuators values , and between sensors and actuator values.
3. Using the enhanced experience, it creates classes of enhanced experiences

4. Each of these classes of enhanced experiences becomes a hypothesis
5. The hypotheses are stored in the database of hypotheses.

The algorithm of generalization creates the new concept (rule) by making clusters by closeness: first roughly and then finer and finer. It creates clusters by goodness (which can also be considered a measure of closeness) and then it creates classes of similarities among situations and actions.

III.A. The Creation of the Subset of Admissible Experiences

This procedure is given the complete database of experiences and it outputs the set of admissible experiences. This procedure is a procedure of "focusing attention". Since it is not possible to process all the experiences in the database of experiences to find rules that will help us with this procedure, it chooses only good examples. The fact that we are able to distinguish "good" from "bad" experiences presumes that we are giving a measure of goodness. Goodness is defined as the difference between a measure of distance between the goal and the situation after the action, and a measure of distance between the goal and the situation before the action. There are different ways to select the experiences suitable for generalization: percentage threshold (all experiences in the top x percentile), goodness threshold (all expereinces better than a certain goodness), number threshold (x best experiences).

III.B. Formatting the Representation of Admissible Experiences

The input to this procedure is the set of admissible experiences and it transforms each of these experiences into enhanced representation experiences (ECER). For simplicity we will use a notation E (experience) for all CER and EE for the enhanced CER (ECER). We do not only use the information directly but we also create combinations by enhancing the situations and the actions. Changing or enhancing the representation of the examples is also used in [22], [23] and [24] where they introduce a new set of operators in order to increase the accuracy and decrease the complexity of hypotheses. This procedure is done in two steps:

1. Use the selected experience to create an enhanced representation situation. This procedure is shown in detail in the definition of enhanced representation situations.
2. Use the selected experience to create an enhanced representation action.

Thus, given an experience:

$$E_i = \{A_{i-1}, S_i, A_i, S_{i+1}\} \tag{1}$$

an enhanced representation experience is

$$\begin{aligned} EE_i &= \{ES_i, EA_i\} \\ &= \left\{ \begin{matrix} \{A_{i-1}, AA_{i-1}, AS_{i-1}, SS_i, S_i\}, \\ \{A_i, AA_i, AS_{i+1}\} \end{matrix} \right\} \end{aligned} \tag{2}$$

The reason behind the creation of this enhanced representation is that by doing this we are giving the capability to the learning algorithm to have extra bases for the creation of classes of rules. The subtraction for example can create rules of comparison: greater than or smaller than. In order to create an enhanced representation experience, the following enhancing function is used upon the situation and the action of the non-enhanced experience:

$$Enh\left(v_1, v2, \ldots, v_n\right) = \left\{ \begin{array}{l} v_i + v_j, v_i - v_j; \\ i = 1 \ldots n; \quad j = 1 \ldots n \\ i > j \end{array} \right\} \tag{3}$$

These combinations among coordinates allow us to use one-dimensional clustering algorithms. If instead of creating combinations we could use multi-dimensional clustering algorithms with the disadvantage that these algorithms will give classes which are cumbersome to store. The one-dimensional clustering algorithms yield classes that can be stored as intervals or centers and standard deviation.

III.C. Creation of Classes of Enhanced Experiences

The procedure takes as input the set of enhanced experiences created in the previous step and a threshold. And, it outputs a set of sets of experiences.

Let us specify some properties that these sets of experiences have:

1. There are no repeated experiences in any of the output classes:

$$Cl_a \cap Cl_b = \{\emptyset\} \, \forall a, b \tag{4}$$

2. Every experience in the input class is included in one of the output classes. The class forming algorithm does not eliminate any experience, (M is the group of all the selected experiences),

$$Cl_1 \cup Cl_2 \cup \ldots \cup Cl_n = M \tag{5}$$

3. Every experience in the output classes was included in the input class. There are no new experiences created by the class forming algorithm.

$$\begin{array}{c} Cl_1 \subseteq M \\ Cl_2 \subseteq M \\ \vdots \\ Cl_n \subseteq M \end{array} \tag{6}$$

Given a set of enhanced experiences as shown in Equation 2 we will represent this set as

$$Cl_{EE_i} = \left\{ \begin{array}{c} \{Cl_{A_{i-1}}, Cl_{AA_{i-1}}, Cl_{AS_{i-1}}, Cl_{SS_i}, Cl_{S_i}\} \\ \{Cl_{A_i}, Cl_{AA_i}, Cl_{AS_{i+1}}\} \end{array} \right\} \quad (7)$$

where Cl_x is a set of triplets $top, bot, cent$ which represent the smallest intervals which include all the values of the coordinates in x and the center of the class. Remember that x is a set and each experience has an x set.

III.D. Constructing the Cause Effect Couples and Hypotheses

The cause effect couples that we are looking for in order to create the rules are already present in each individual enhanced experience (Equation /refeq:enhance) which have the following interpretation:

$$\{A_i, AA_i, AS_{i+1}\} \xrightarrow{applied\ at}$$
$$\{A_{i-1}, AA_{i-1}, AS_{i-1}, SS_i, S_i\} \xrightarrow{yields} g\ for\ G \quad (8)$$

where g is goodness and G is the assigned goal. This interpretation is taken directly from the way that the experience was recorded.

The following assumption is used in our approach:

> There exists a class of enhanced representation actions A_{enh} which has already been applied to a class of enhanced representation situations S_{enh} and which produced values of goodness in the interval from G_{min} to G_{max}

then

> If we have a situation belonging to a class S_{enh} then an action can be determined within class A_{enh} which will provide goodness G_{max}.

So, under this assumption each class of enhanced experiences (Equation 7 becomes a hypothesis. becomes

$$H_i = \left\{ \begin{array}{c} G, g_{ave}, Num_{exp}, \\ \{Cl_{A_{i-1}}, Cl_{AA_{i-1}}, Cl_{AS_{i-1}}, Cl_{SS_i}, Cl_{S_i}\} \rightarrow \\ \{Cl_{A_i}, Cl_{AA_i}, Cl_{AS_{i+1}}\} \end{array} \right\} \quad (9)$$

where G is the goal, g_{ave} is the average goodness and Num_{exp} is the number of experiences used for the creation of the hypothesis. This last measure is a quantitative measure of the strength of the rule, and it will be used when we have conflicting rules for the same situation. In [25] a similar measure is called credibility.

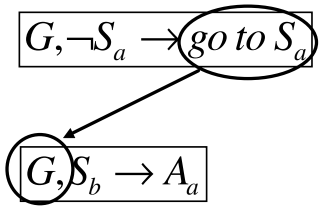

Figure 1: Father and son hypotheses

A second kind of hypotheses that can also be extracted from the previous class which is based in the following principle: if we only have recommendations about actions in certain situations that give the desired goodness, then we should assign as goal to achieve these situations where we have recommendations about actions. In other words:

$$\begin{array}{rcl} H_0 & = & \left\{ \begin{array}{c} G, g_{ave}, Num_{exp}, \\ \{\neg Cl_{A_{i-1}}\} \rightarrow go\ to\ Cl_{A_{i-1}} \end{array} \right\} \\ H_1 & = & \left\{ \begin{array}{c} G, g_{ave}, Num_{exp}, \\ \{\neg Cl_{AA_{i-1}}\} \rightarrow go\ to\ Cl_{AA_{i-1}} \end{array} \right\} \\ H_2 & = & \left\{ \begin{array}{c} G, g_{ave}, Num_{exp}, \\ \{\neg Cl_{AS_{i-1}}\} \rightarrow go\ to\ Cl_{AS_{i-1}} \end{array} \right\} \\ H_3 & = & \left\{ \begin{array}{c} G, g_{ave}, Num_{exp}, \\ \{\neg Cl_{SS_i}\} \rightarrow go\ to\ Cl_{SS_i} \end{array} \right\} \\ H_4 & = & \left\{ \begin{array}{c} G, g_{ave}, Num_{exp}, \\ \{\neg Cl_{S_i}\} \rightarrow go\ to\ Cl_{S_i} \end{array} \right\} \end{array} \quad (10)$$

for every subclass of the situation of the original situation. The problem with this kind of hypotheses is that the system does not know how to "go to" these situations. So, these situations that we have to go to become new sub-goals. And the generalization algorithm starts again.

Figure 1 shows a father hypothesis on the top, and a son hypothesis which takes its goal from the fathers situation on the bottom. Since the son hypothesis has a new goal (the father's situation) it has a new measure of goodness.

When the database of hypotheses grows, it becomes a tree of task decomposition shown in Figure 2.

III.E. Step by Step Recursive Generalization

The first step that the algorithm of generalization does is to check the database of hypotheses. The only schema present in the database says that it should "Make D=0". Since there is nothing in the database that shows what to do in order to "Make D=0" it assigns "D=0" as the goal, (D is the distance to the target) and collects a random sequence of experiences. Figure 3 shows "event-grams" of 3000, and

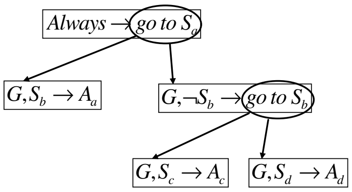

Figure 2: The Task decomposition tree

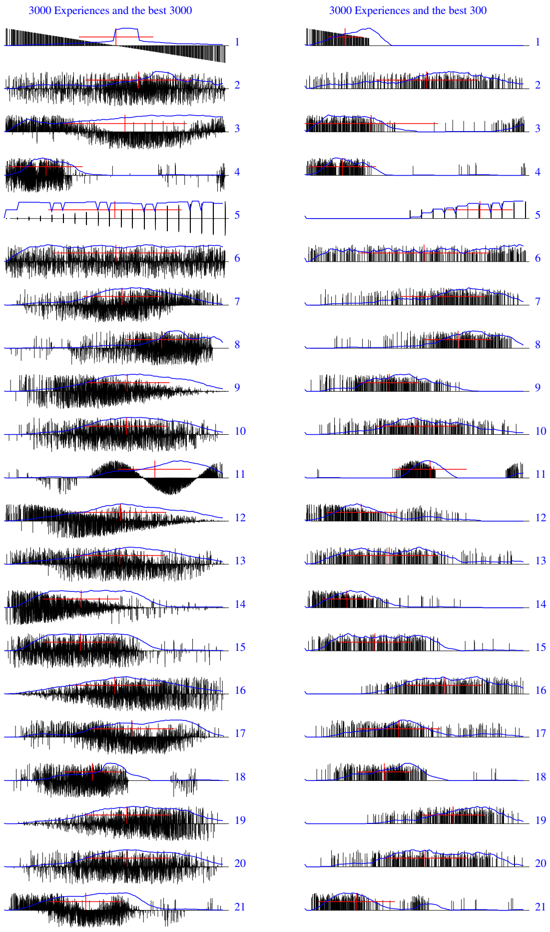

Figure 3: Eventgrams for 3000 experiences (left); 300 best experiences out of the 3000 (right)

the 300 best experiences respectively. These eventgrams show in each abscissa the value of each of the coordinate in each experiences and in the ordinates the goodness of that experience. In this case we have 21 coordinates in each experience, as previously descrived they correspond to convinations of among sensor values and actuator values.

The next step in the algorithm of generalization is the creation of the classes of experiences using one of the previously shown methods. The clustering algorithm discovers two classes. They correspond to the actuator "go forward" and the enhanced representa-

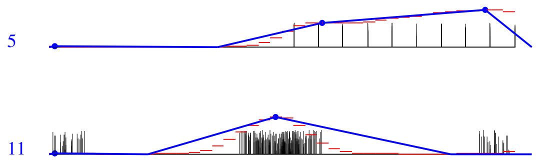

Figure 4: The remaining classes

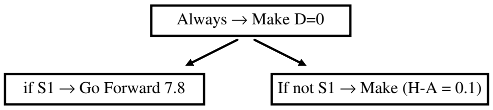

Figure 5: The database of hypotheses at this step

tion sensor "Heading - Angle to Target". Note that in the "go forward" the two classes that are present in Figure 4 (right) are unified using the unifying rules.

Figure 5 shows the database of hypotheses after the first iteration of the generalization algorithm. Two hypotheses are spawned. The values 0.1 and 7.8 are the centers of the clusters found by the averaging algorithm. "S1" is the union of all the situations of these "good" experiences. The two hypotheses have an important difference, the first one gives a recommendation about what actuator to use in this situation. The second one, says that if we are not in this situation we should go to this situation.

The problem with the second hypothesis is that there is nothing in the database of hypotheses that gives instructions about how to "Make (H-A=0.1)". Thus, the generalization algorithm starts again:

1. a new goodness measure is taken by checking whether the experience brought it closer or further to the new goal
2. the new goal is to "make H-A=0.1".
3. all the experiences are re-ranked using the new goodness measure for the new goal
4. the "best" experiences are selected using one of the previously selected methods.
5. these experiences are sent again to the classification algorithm, in other words, new eventgrams are created for the new goal and the classification algorithm creates new clusters which will be used for creating rules to follow the new goal.

These 300 experiences are sent to the averaging algorithm which finds two clusters in coordinate 6 which corresponds to the actuator "Rotate". These two cluster become two hypotheses that get incorporated in the database.

IV. Conclusions

1. The algorithm of nested clustering is introduced, which declares hypotheses with no prior interpre-

tation of what is the suggested assignment

2. If the suggested assignment is not an action then it becomes a new goal and the information about cause-effect relationships is rearranged to accommodate the new goal.
3. The algorithm converges if the problem has a solution. Otherwise, loops emerge that can be easily detected and interpreted as “no solution”.
4. The algorithm was checked statistically in simulation.

V. References

[1] J. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21, 1991.

[2] A. Meystel. Multiscale models and controllers. In *Proceedings of IEEE/IFACS Joint Symposium on Computer Aided Control System Design*, Tucson, AZ, 1994.

[3] J. Albus, A. Meystel, and S. Uzzaman. Nested motion planning fon an autonomous robot. In *IEEE International Conference on Aerospace Systems*, Westlake Village, CA., 1993.

[4] A. Lacaze and M. Meystel. Baby sub: Using schemata for conceptual learning. In *Proceedings of the Workshop on Neural Architectures and Distributed AI: From Schema Assemblages to Neural Networks*, Los Angeles, ,CA, 1993. University of Southern Califonia.

[5] E. A. Feigenbaum. The simulation of verbal learning behavior. *Computers and Thought*, 1963.

[6] M. Lebowitz. Categorizing numeric information for generalization. *Cognitive Science*, 9:284–309, 1985.

[7] M. Lebowitz. Concept learning in a rich input domain: Generalization based memory. In R. S. Michalski, J. G. Carbonell, and Mitchell T. M., editors, *Machine Learning : and Artificial Intelligence Approach*, California, 1983. Morgan Kaufmann.

[8] P. Smyth and R.M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Enineering*, 4(4), 1992.

[9] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[10] D. Fisher. *Knowledge Acquisition via Incremental Conceptual Clustering.* PhD thesis, University of California, Irvine, 1987.

[11] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40, 1989.

[12] C.A. Sammut. Concept development for expert system knowledge bases. *Australian Computer Journal*, 17, 1985.

[13] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2), 1986.

[14] P. Langley, G. L. Bradshaw, and H. A. Simon. Rediscovering chemistry with bacon system. In R. S. Michalski, J. G. Carbonell, and Mitchell T. M., editors, *Machine Learning: an Artificial Intelligent Approach*, California, 1983. Morgan Kaufmann.

[15] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach*, California, 1983. Morgan Kaufmann.

[16] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:88–106, 1986.

[17] L. Wang and J. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions of System, Man, and Cybernetics*, 22(6), 1992.

[18] J. Grefenstette and A. Schultz. An evolutionary approach to learning in robots. In *Proceedings MLC-COLT Workshop of Robotic Learnings*, New Brunswick, 1993. Rutgers University.

[19] J.A. Franklin. Learning control in a robotic system. *IEEE Transactions Systems, Man, and Cybernetics*, 1987.

[20] M. Salganicoff, L. G. Kunin, and L. H. Ungar. Active exploration based id-3 learning for robot grasping. In *Proceedings MLC-COLT Workshop of Robotic Learning*, New Brunswick, N.J., 1994. Rutgers University.

[21] L.P. Kaelbling. *Learning in Embedded Systems.* PhD thesis, Stanford University, 1990.

[22] S. Kramer. Cn2-mci: a two-step method for constructive induction. In *Constructive Induction and Change of Representation, Working Notes of the ML-COLT-94 Workshop.* Rutgers University, 1994.

[23] N. Japkwicz and H. Hirsh. Towards a bootstrapping approach to constructive induction. In *Constructive Induction and Change of Representation, Working Notes of the ML-COLT-94 Workshop.* Rutgers University, 1994.

[24] J. R. Pfahrringer. Cipf 2.0: A robust constructive induction system. In *Constructive Induction and Change of Representation, Working Notes of the ML-COLT-94 Workshop.* Rutgers University, 1994.

[25] I.J. Cox and J.J. Leonard. Modelling a dynamic environment using a bayesian multiple hypothesis approach. *Artificial Intelligence*, 66:311–334, 1994.