Abstraction of Sensory-Motor Features

Kazuo Hiraki
Electrotechinal Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki, 305 Japan
khiraki@etl.go.jp

Abstract

This paper presents a way that enables robots to learn abstract concepts from sensory/perceptual data. In order to overcome the gap between the low-level sensory data and higher-level concept description, a method called *feature abstraction* is used. Feature abstraction dynamically defines abstract sensors from primitive sensory devices and makes it possible to learn appropriate sensory-motor constraints. This method has been implemented on a real mobile robot as a learning system called ACORN-II. ACORN-II was evaluated with some empirical results and shown that the system can learn some abstract concepts more accurately than other existing systems.

Introduction

All of existing intelligent agents can use sensory information for interacting with the external world, but on the other hand they are suffering the limitation from their hardware constraints. The representation of the perceived world depends on the type and capability of sensory devices of each agent. The perceived world of dogs that have a keen nose might be quite different from the one perceived by human beings. We could ask here interesting but difficult questions:

- *What representation can be formed by using limited sensory devices?*
- *Can agents, each of them having different sensory devices, share knowledge?*

The goal of this paper is to present the system that enables robots to learn abstract knowledge, such as natural language like commands, by interaction with human beings. To this end, we focus on *abstraction* of information from primitive sensors to enable to share the knowledge between humans and robots. A robot that is equipped with sensors must have a base representation language that is grounded in the available sensors. However, many tasks that the robot required to learn will need *higher-level* features that cannot be perceived directly by the sensors and may not appear in the initial representation language. Thus the abstraction of primitive features plays an important role for learning from sensory/perceptual information.

The recent progress in robot learning has given important results particularly in reinforcement learning(e.g. Lin, 1991; Mahadevan & Connell, 1991; Whitehead & Ballard, 1990). Through reinforcement learning under suitable conditions, a robot can optimally choose an action based on its current and past sensor values such that it maximizes over time a reward function measuring its performance. This approach seems to be attractive because robots can learn their behavior by using data only from primitive sensors. However, this approach still has the problem of abstraction of features. In reinforcement learning, features used for perception are carefully selected and defined *a priori*. The system designer is sometimes made uneasy with preparing appropriate features with respect to the target knowledge.

In order to resolve the problem of abstracting features, we have implemented ACORN-II, a system that learns robot-commands from positive and negative instances[1]. ACORN-II uses a combination of two different learning algorithms – *generalization to interval*(GTI) and *feature abstraction*(FA) – for learning from sensory/perceptual information. GTI is an incremental algorithm that generalizes over numeric attributes, and makes algebraic expressions that represent the constraints among sensors and actuators. FA is responsible for the discovery of adequate features in GTI's generalization.

The idea of feature abstraction is inspired from the recent progress of constructive induction or feature construction. Feature construction is a method to discover appropriate features for representing instances in inductive learning. Several researchers have shown the importance of constructing new features in the various fields (e.g. Aha, 1991b; Bala, Michalski & Wnek, 1992; Fawcett & Utgoff, 1992; Matheus, 1991; Muggleton, 1987). However, not much has been said yet about its application to appropriate domains. We claim that abstracting features in robot learning is indeed one of the most suitable domains where feature construction can be applied. The reason of this claim is that defining appropriate features in robot learning is often more difficult than other domains. We believe that the idea proposed in this paper could contribute not only to the field of machine learning, but also to problems in robotics, particularly in multi-sensor fusion and sensor integration (e.g. Durrant-Whyte 1987; Shafer, Stentz & Thorpe, 1986).

[1] Because we are much concerned with the interaction between human beings and robots(Anzai, 1993) more than *self-governing* of robots, the current system uses learning-from-example framework(i.e. assuming *teacher*). However, it could be possible to extend the idea of ACORN-II to unsupervised framework, such as reinforcement learning.

ACORN-II is implemented on real autonomous mobile robots shown in Figure 1. This robot is a small autonomous mobile robot with the size of approximately 30 cubic cm, controlled by a Toshiba TMP68301-16F processor. The robot is equipped with four sonar sensors to measure the distance, and two pulse motors for wheels. ACORN-II can use data those sensors and actuators, but does not assume any unrealistic sensors/actuators that can detect high-level descriptions.

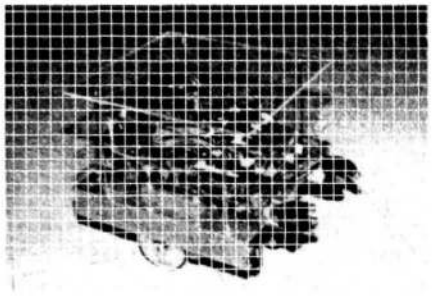

Figure 1: One of autonomous mobile robots developed in our laboratory.

The following sections describe ACORN-II's idea and empirical evaluation of the system. The next section gives the concept of feature abstraction with a simple example for illustrating the problems in learning higher-level knowledge from low-level data. Section 3 elaborates ACORN-II's learning in detail. Section 4 evaluates ACORN-II by comparing the system with other existing systems. Finally, Section 5 concludes with some discussions and future work.

Situated Concept and Feature Abstraction

An Example

Meanings of abstract concepts sometimes depend on situations, and higher-level features are needed for realizing those situations. Suppose that a robot has two sonar sensors for detecting distances and also has two actuators for wheels. Also assume that this robot is to learn the natural language-like command *"Big Turn"*. Figure 2 shows positive instances((A) $\sim$ (C)) and a negative instance(D) for learning the command. A circle represents a locus of the robot and a square represents a room in which the robot is located. Note that Figure 2(D) is a negative instance even though its radius is larger than (A)'s because the actual curvature for a *"Big Turn"* depends on the size of the room.

Table 1 shows the primitive features and the time series values of those features[2]. We suppose that primitive features, each of which corresponds to a sensor or an actuator, are given. In Table 1, $sensor_r$ and $sensor_l$ denote

[2]It is not so simple to define sampling intervals and the correspondences between sets of sampled data. For simplicity, we assume that all instances can be represented with the same number of samples.

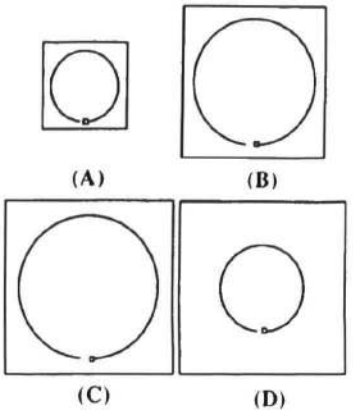

Figure 2: Positive instances (A)–(C) and negative instance (D) for command *"Big Turn"*

the primitive features for the data from the robot's right and left sonar sensor, and $motor_r$ and $motor_l$ denote those for the speed of right and left wheels, respectively.

Table 1: **(a)** Primitive features of positive instances (A)–(C) and negative instance (D) for *"Big Turn"*.

		$sensor_r$	$sensor_l$	$motor_r$	$motor_l$
(A)	0	5.000	165.000	0.000	0.000
	1	17.293	176.762	5.965	4.080
	2	5.086	165.005	5.965	4.080
	3	21.668	181.068	5.965	4.080
(B)	0	15.000	305.000	0.000	0.000
	1	36.864	326.259	10.046	8.162
	2	15.150	304.999	10.046	8.162
	3	43.643	332.989	10.046	8.162
(C)	0	20.000	350.000	0.000	0.000
	1	44.589	373.973	11.302	9.418
	2	20.173	350.008	11.302	9.418
	3	52.809	382.133	11.302	9.418
(D)	0	85.000	285.000	0.000	0.000
	1	109.557	309.006	7.221	5.336
	2	85.139	285.040	7.221	5.336
	3	117.775	317.166	7.221	5.336

An important problem in this example is that the direct use of primitive features often causes generation of a huge number of disjunctions, even if a good generalization algorithm is available. For example, if we use the "hyper-rectangle method"(Salzberg, 1991) (one of exemplar-based learning approaches), in which concepts are represented with hyper-rectangles in Euclidean n-space, the method generates a huge number of disjunctions such as:

$$\{(17.293 \leq sensor_r^1 \leq 44.589) \wedge (sensor_l^1 = 176.762) \wedge (motor_r^1 = 5.965) \wedge (motor_l^1 = 4.080)\} \vee$$
$$\{(17.293 \leq sensor_r^1 \leq 44.589) \wedge (326.259 \leq sensor_l^1 \leq 373,973) \wedge (motor_r^1 = 5.965) \wedge (motor_l^1 = 4.080)\} \vee$$
$$\{(17.293 \leq sensor_r^1 \leq 44.589) \wedge (sensor_l^1 = 176.762) \wedge (10.04 \leq motor_r^1 \leq 11.302) \wedge (motor_l^1 = 4.080)\} \vee$$
$$\{(17.293 \leq sensor_r^1 \leq 44.589) \wedge (326.259 \leq sensor_l^1 \leq 373,973) \wedge (10.04 \leq motor_r^1 \leq 11.302) \wedge (motor_l^1 = 4.080)\} \vee \cdots \vee \{(17.293 \leq sensor_r^1 \leq 44.589) \wedge (326.259 \leq sensor_l^1 \leq 373,973) \wedge (10.04 \leq motor_r^1 \leq 11.302) \wedge (8.162 \leq motor_l^1 \leq 9.418)\}$$

This occurs because of the lack of appropriate features to

Table 2: Possible higher-level features

		$\frac{motor_l}{motor_r}$	$sensor_r + sensor_l$	$\frac{\frac{motor_l}{motor_r}}{sensor_r^0 + sensor_l^0}$
(A)	1	0.684079	194.056	0.004024
	2	0.684079	170.092	0.004024
	3	0.684079	202.737	0.004024
(B)	1	0.812416	363.124	0.0025388
	2	0.812416	320.15	0.0025388
	3	0.812416	376.634	0.0025388
(C)	1	0.833258	418.564	0.00225205
	2	0.833258	370.182	0.00225205
	3	0.833258	434.943	0.00225205
(D)	1	0.739018	418.564	0.00199735
	2	0.739018	370.18	0.00199735
	3	0.739018	434.941	0.00199735

represent the relative size of a locus with respect to the size of a room, nevertheless the meaning of the command *"Big Turn"* depends on the size of a room.

In order to avoid inappropriateness as in the example *"Big Turn"*, one would consider to use higher-level features for representing instances given to the generalization procedure. Table 2 shows some possible higher-level features that could be used for representing instances. Note that the feature $\frac{\frac{motor_l}{motor_r}}{sensor_r^0 + sensor_l^0}$ represents the ratio of the radius of a circle $(\frac{motor_l}{motor_r})$ to the size of a room $(sensor_r^0 + sensor_l^0)$. This is a missing feature hidden in the given primitives for the situated concept *"Big Turn"*.

Using this new feature, a generalization method will be able to generate appropriate constraints between the sensors and actuators from the data as follows:

$$0.00225205 \leq \frac{\frac{motor_l}{motor_r}}{sensor_r^0 + sensor_l^0} \leq 0.004024$$

The problem is that the robot cannot directly perceive higher-level features such as $\frac{\frac{motor_l}{motor_r}}{sensor_r^0 + sensor_l^0}$ for given instances.

We can consider here two alternative ways for resolving this problem.

1. Taking account of all of possible situations to be recognized, prepare a huge number of primitive features with respect to the situations.
2. Dynamically construct new features from the existing features when a situation needs to be recognized.

Obviously, the first way is unrealistic in real-world domains because we have to consider virtually infinite situations. As for the example *"Big Turn"* in the previous section, it is impossible to know all of the sizes of rooms in the world. In this paper we choose the second way, constructing new features from existing ones.

Sensor Integration and Abstract Sensors

As shown in the example *"Big Turn"*, one could consider a variety of features such as *the size of the room* or *the radius of the circle*. There are two important aspects in these features. First, we do not have to prepare any extra sensors for defining the features. These features can be defined only by using primitive features that correspond to each sensory devices on the robot. Second, these features may be more abstract than primitive features, since these features are constructed by applying some operators to primitives. In the example *"Big Turn"*, abstract features are constructed by using arithmetic operators such as $\{-, +, *, /\}$ to the primitive features.

In general, possible abstract features derived from primitives can be formed a tree structure such as shown in Figure 3. In this tree, there are four primitive features, *A*, *B*, *C*, *D*, and four operators $\{-, +, *, /\}$. The top node represents the set of primitive features, and each node at other levels represents the set of features that includes new features constructed from primitives. An interesting point is that the depth of the tree corresponds to the level of sensory abstraction. The level of a node in the tree matches the number of operations that are applied to the primitive features. For example, the node $\{\frac{A}{B} * C, D\}$ exists in the second level, since the feature $\frac{A}{B} * C$ in this node is constructed by applying an operator twice.

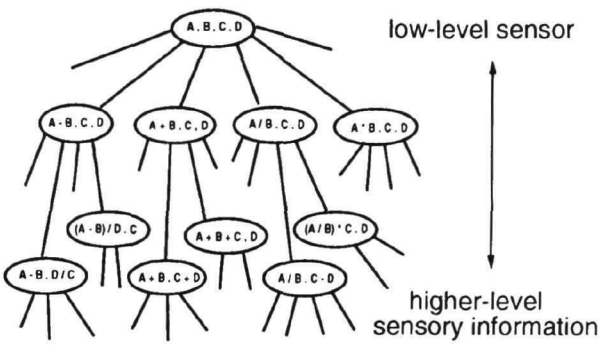

Figure 3: Search depth and level of sensor abstraction

As shown in Figure 3, abstract features play the role of abstract sensors that do not actually exist as sensory devices. This characteristic leads to the possibility of *automatic sensor fusion* or *automatic sensor integration*. In general, integrating sensory information needs the prior knowledge such as the relationship between sensory devices(e.g. Durrant-Whyte, 1987; Shafer, Stentz, & Thorpe, 1986). However, if particular procedures for exploring the tree are available, sensory information can be automatically integrated depending on the situation.

Feature Abstraction

Formally, the tree of a possible feature set can be defined by *primitive features* $(f_1, \ldots, f_n)$ which correspond to initial sensory devices, *operators* $(op_1, \ldots, op_m)$ for constructing new features, and *criteria* for selecting features. Now we call this tree as the *feature space*, and the procedure for exploring this tree *feature abstraction*.

In general, it is one of the most difficult task to define features for learning systems. In the robot domain, however, primitive features are easily determined because they correspond to sensory devices on the robot.

For operators, we could assume many possibilities. We used only four operators, $\{-, +, *, /\}$ in the example

"Big Turn", but in general we can use many other operators such as sin, cos, log and so on. Practically, however, we should select a small subset of operators from the infinite set of potential ones.

Criteria for selecting features that will be applied to operators also has many possibilities. As for the tree in Figure 3, we removed two selected features used to construct a new feature. However, one would consider to keep those features. Changing a criterion for selecting features often causes the change of computational costs. For example, if we use the former criterion, the size of the feature space is

$$\prod_{i=2}^{m} N^i *_i C_2$$

where N is the number of primitive features and m is the number of operators. But the size becomes infinite if we take the latter criterion.

The concept of feature abstraction is analogous to the one in scientific discovery systems such as Bacon (Langley, Bradshow, & Simon, 1983) and ABACUS (Falkenhainer, & Michalski, 1986). Those scientific discovery systems try to discover mathematical expressions that summarize a body of data. However, feature abstraction differs from those systems in the purpose of discovering expressions. Feature abstraction generates the expressions as just "features" for representing instances. So, one would need another procedure(e.g. generalization module) that makes use of the result of feature abstraction. But, on the other hand discovery systems generally seek expressions that summarize given data and these expressions are never used for any other purpose.

Learning Sensory-Motor Constraints

According with the concept of feature abstraction, we have implemented ACORN-II. Learning in ACORN-II is divided into two parts, GTI for generating hypotheses and FA for feature abstraction. This section elaborates the relationship between the two procedures with a concrete example.

Generalization in Acorn-II

There are two important characteristics that should be considered in a robot learning system. First, its generalization method must deal with numeric features. Since most perceptual/sensory information is expressed in continuous, numerical form, a learning system of an intelligent agent has to transform continuous values into discrete ones for symbolic reasoning. Second, the system should use an incremental algorithm for learning because it is difficult for an autonomous agent to have an entire set of instances at the same time.

To deal with these issues, ACORN-II uses the generalization method called GTI. GTI is one of algorithms that are used in families of *exemplar-based* systems (e.g. Aha, Kibler & Albert, 1991a; Salzberg, 1991). GTI is a simple version of "hyper-rectangle method" (Salzberg, 1991) that generates hyper-rectangles on n-dimensional space, where n denotes the number of features. The primary difference between GTI and Salzberg(1991)'s program is that our method has the interface between the procedure for feature abstraction described below[3].

A Procedure for Feature Abstraction

For discovering appropriate features from primitives, ACORN-II uses the feature abstraction method called FA. As mentioned in Matheus(1991), all of the systems that try to construct new features must take account of the appropriate time and the conditions for invoking the procedure for constructing new features. Figure 4 illustrates the relationship between GTI and FA. When the number of GTI's disjunctive regions[4] is over a given threshold, FA is invoked and constructs new features. After that, GTI re-represents regions by using these new features. FA uses arithmetic operators in order to construct higher-level features from numeric ones.

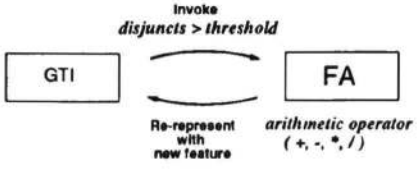

Figure 4: The relation between GTI and FA

Table 3 illustrates the algorithm of FA, where *make_new_features* selects two existing features and applies an operator from $\{-, +, *, /\}$ to these two features. This algorithm explores the search space that consists of operators and primitive features until the number of regions becomes under a given threshold γ.

This algorithm does not use ecological constraints such as the functional similarity of sensors and/or motors. For a practical search, however, we need to extend the current algorithm to more efficient one that can use ecological constraints.

Evaluation of Acorn-II

In this section, we evaluate ACORN-II with empirical results on robot-command learning. We created a data set for the situated command *"Big Turn"* described in previous sections. The data set contained 100 instances and was divided into two subsets. One subset was used for training, and the other for testing. Each subset had 50 positive or negative instances. Those instances were represented by four primitive features that correspond to two sonar sensors for detecting distances and two actuators for wheels, respectively.

Figure 5 shows the comparison of learning performance between ACORN-II, CN2 (Clark & Niblett 1989) and C4 (Quinlan, 1993). All of the three systems can

[3] We used GTI for learning spatial relations from images in our previous system(Hiraki et.al., 1991a,1991b). This system enable to perform a variety of performance tasks by using *constraint logic programming* (Leler, 1988).

[4] We use the words, *region, hyperrectangle, constraint expression*, in the same meaning.

Table 3: **FA**: Feature abstraction algorithm in ACORN-II

Let N be the number of regions constructed by GTI, F be the set of existing features $F = \{f_1, \ldots, f_n\}$, γ be a positive integer and OP be the set of operators $\{-, +, *, /\}$ and $L = \{\}$

1 If $N > \gamma$
  Then L:= L + **make_new_features(F,OP)**
2 If $L = null$ Then stop.
3 Select F_{new} from L, $L := L - F_{new}$
  M := number of new regions re-represented with F_{new}
4 If $M < \gamma$ Then Return M, F_{new}
5 Else L:= L + **make_new_features(F_{new},OP)** and goto 2.

make_new_features(F,OP)
Let FF be a set of pairs of two features (f_i, f_j) $(f_i, f_j \in F, f_i \neq f_j)$, $MF = \{\}$

1 If FF is null Then return MF
  Select a pair (f_k, f_l), $FF := FF - \{(f_k, f_l)\}$
2 If OP is null Then goto 1.
3 Select an operator op_m from OP, $OP - \{op_m\}$
  f_{new} := New feature obtained by applying op_m to f_k, f_l
  $NF := F - \{f_k, f_l\} + \{f_{new}\}$
  $MF := MF \cup NF$ and goto 2.

deal with numeric features, but CN2 and C4 do not support automated feature abstraction. These three systems were trained by instances taken from the training subset, then predicted an unseen instance taken from the testing subset. The result shows that ACORN-II's percentage accuracy is superior to CN2 and C4, even though CN2 and C4 use non-incremental methods and ACORN-II has to take incremental input of instances. This is because the meaning of the command *"Big Turn"* depends on situations and only ACORN-II is able to cope with it by using appropriate features generated by feature abstraction.

After training instances were given, ACORN-II constructed a new feature $(s_r + s_l + m_r)/m_l$ and generated constraint expression:

$$\{40.50 < (s_r + s_l + m_r)/m_l < 43.14\}.$$

Note that the feature used in the final constraint expression is different from the feature $\frac{\frac{m_r}{m_l}}{s_r + s_l}$ described in Section 2. We can interpret this feature as follows:

$$\text{When } (s_r + s_l) \gg m_r, \frac{s_r+s_l+m_r}{m_l} = \frac{\frac{s_r+s_l}{m_r}+1}{\frac{m_l}{m_r}} \approx \frac{s_r+s_r}{\frac{m_l}{m_r}}.$$

This is just the reverse version of the denominator and numerator in $\frac{\frac{m_r}{m_l}}{s_r + s_l}$.

One of the remarkable characters of ACORN-II is its incremental learning. To illustrate this advantage, we first gave the system 30 instances from the "large-size room", and then gave instances from the "small-size room"[5].

[5] We prepared five sizes of room for each.

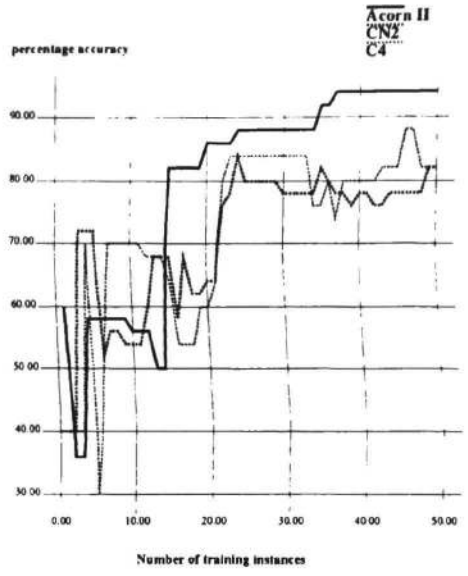

Figure 5: Learning curves of ACORN-II, CN2 and C4

Figure 6 shows the result of this experiment. The result suggests ACORN-II's robustness against the change of environments.

Conclusion

In this paper, we described ACORN-II that can learn situated robot-commands from primitive sensor/actuator data. In order to overcome difficulties in learning abstract commands from low-level data, ACORN-II constructs new features by using feature abstraction. Empirical results suggest that ACORN-II is superior to well-known existing systems.

The result of this research demonstrates the usefulness of feature abstraction in robot learning. The algorithms used in ACORN-II can be regarded as *automatic sensor fusion*. By using feature abstraction, the robot can integrates the sensory information without prior knowledge such as the relationship between each sensory device.

Problems of inventing an efficient search method for exploring the feature space is opened to our future works. For the implemented example described in this paper, the robot has only 4 sensors/actuators, and thus the size of the space is not so large. However, in case the number of sensors/actuators is large, we need more sophisticated criterion for selecting features. One of the possible approaches for avoiding combinatorial explosion would be using feature selection method (e.g. Kira & Rendell, 1992) before invoking FA.

The current version of FA does not have operators for symbolic features. The use of symbolic operators such as *logical and* and *or* is also opened to our future works.

The concept of feature abstraction could be thought as a kind of conceptual change (Ram, 1993). We will investigate the generality of our method in other domains as well as in learning more complex robot-commands.

This paper concentrated to develop the way to share knowledge between human and robot. So, we took the

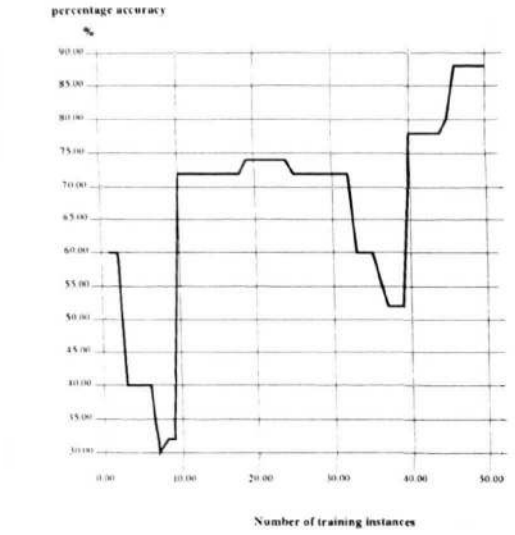

Figure 6: Learning curve of ACORN-II: First 30 instances are taken from "large-size room", remainders from "small-size room"

supervised approach, i.e. positive and negative instances are given by human. However there is no doubt that the concept of feature abstraction is useful also in unsupervised learning. Combining feature abstraction and some unsupervised learning is our next step.

Acknowledgements

The author would like to thank Yuichiro Anzai for comments and suggestions, and Toyoshi Okada for his help in the implementation.

References

Aha,D.W., Kibler,D. & Albert,M.K. (1991a). Instanced-based learning algorithms, *Machine Learning*, Vol. 6, pp. 37–66.

Aha,D. W. (1991b). Incremental Constructive Induction: An Instance-Based Approach, In L. Birnbaum and G. Collins, editors, *Proc. of Eighth International Machine Learning Workshop ML'91*, pp. 117–121.

Anzai,Y. (1993). Human-Computer Interaction in Multiagent Environment, In *Proc. of HCI International '93*, pp. 2–7.

Bala,J.W., Michalski,R.S. & Wnek,J. (1992). The Principal Axes Method for Constructive Induction, In *Proc. of Ninth International Workshop ML'92*, pp. 20–29.

Clark,P. & Niblett,T. (1989). The CN2 Induction Algorithm, *Machine Learning*, Vol. 3, pp. 261–283.

Durrant-Whyte,H.F. (1987). Sensor Models and Multi-Sensor Integration, In *Proc. of Spatial Reasoning and Multi-Sensor Fusion*, pp. 303–312.

Falkenhainer,B. C. & Michalski,R. S. (1986). Intergrating Quantitative and Qualitative Discovery: The *Abacus* system, *Machine Learning*, Vol. 1, pp. 367–401.

Fawcett,T. E. & Utgoff,P. E. (1992). Automatic Feature Generation for Problem Solving Systems, In *Proc. of Ninth International Workshop ML'92*, pp. 144–153.

Hiraki,K. Gennari,J., Yamamoto,Y. & Anzai,Y. (1991a). Encoding Images into Constraint Expressions, In *Proc. of Thirteenth Annual Conference of Cognitive Science Society*, pp. 31–36.

Hiraki,K., Gennari,J., Yamamoto,Y. & Anzai,Y. (1991b). Learning Spatial Relations from Images, In L. Birnbaum and G. Collins, editors, *Proc. of Eighth International Machine Learning Workshop ML'91*, pp. 407–411.

Kira,K. & Rendell,L. A. (1992). A Practical Approach to Feature Selection, In *Proc. of Ninth International Workshop ML'92*, pp. 249–256.

Langley,P., Bradshow,G. L. & Simon,H. A. (1983). Rediscovering chemistry with Bacon system, In *Machine Learning: An artificial intelligence approach(Vol.1)*, Morgan Kaufmann.

Leler,W. (1988). *"Constraint Programming Languages: Their Specification and Generation "*, Addison-Wesley.

Lin, L. (1991). Programming Robot Using Reinforcement Learning and Teaching, In *Proc. of Tenth National Conference on Artificial Intelligence*, pp. 781–786.

Matheus,C. J. (1991). The Need for Constructive Induction, In L. Birnbaum and G. Collins, editors, *Proc. of Eighth International Machine Learning Workshop ML'91*, pp. 173–177.

Mahadevan,S. & Connell,J. (1991). Automatic Programming of behavior-based robots using reinforcement learning, In *Proc. of AAAI'91*, pp. 768–773.

Muggleton,S. (1987). Duce: An Oracle Based Approach to Constructive Induction, In *Proc. of Tenth International Joint Conference on Artificial Intelligence*, pp. 287–292.

Quinlan,J. R. (1993). *"C4.5: Programs for Machine Learning"*, Morgan Kaufmann.

Ram, A. (1993). Creative Conceptual Change, In *Proc. of Fifteenth Annual Conference of Cognitive Science Society*, pp. 17–26.

Salzberg,S. (1991). A Nearest Hyperrectangle Learning Method, *Machine Learning*, Vol. 6, pp. 251–276.

Whitehead,S. & Ballard,D. (1990). Active Perception and Reinforcement Learning, In *Proceedings Seventh International Conference on Machine Learning*, pp. 179–188.

Shafer,S.A., Stentz,A. & Thorpe,C. (1986). An Architecture for Sensor Fusion in a Mobile Robot, In *Proc. of IEEE International Conference on Robotics and Automation*, pp. 2002–2011.