THE DEVELOPMENTAL APPROACH TO MULTIMEDIA SPEECH LEARNING

Juyang Weng, Yong-Beom Lee and Colin H. Evans

Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

ABSTRACT

This paper introduces the developmental approach to speech learning, motivated by human cognitive development from infancy to adulthood. Central in the developmental approach is what is called the developmental algorithm. We introduce AA-learning as a basic learning mode for our developmental algorithm. The developmental algorithm enables the system to learn new tasks without a need of reprogramming. Some experimental results for AA-learning using our developmental algorithm are presented.

1. INTRODUCTION

Very impressive performance for speech recognition has been achieved for a large vocabulary under continuous speech, but the performance degrades significantly when environmental variation and speaker variation increase [2]. An important direction for improving speech recognition performance is to use the multimedia context. However, the challenging issue is how to define and use context. The traditional way to approach this issue is to restrict the environmental domain and the scope of the task to be performed. Then humans manually model the context in a domain-specific and task-specific manner. Such manual labor in system development is very tedious and this manual nature fundamentally limits the variety of environment to which a speech recognition technology can apply.

In this paper, we propose what is called the developmental approach. The developmental approach is motivated by human cognitive development from infancy to adulthood through interactions with the environment. From an engineering point of view, it is to automate the process of system development. Put more specifically, it is to automate task-specific programming. This is accomplished by what is call a developmental algorithm — an algorithm that automatically handles system development. The goal of the experiments presented here is to verify the working of our developmental algorithm, and it is by no means to compare with any of the existing speech recognition systems in terms of recognition performance.

This work was supported by NSF grant No. IIS-9815191. The authors would like to thank Wey S. Huang for his contribution to an earlier version of the program.

2. AA-LEARNING

An agent is something that senses and acts. For example, a speech recognition system may sense speech signals and may act by output information, such as text or control signal. An agent has a number of sensors and effectors. Its extroceptive, proprioceptive and interoceptive sensors sense the external world (e.g., auditory), its own actions (e.g., vocal tract shape), and internal events (e.g., internal clock), respectively. If the agent has a predefined preference to the sensed value of a sensor, (e.g. a preference of 1 to -1) this sensor is called a biased sensor. Otherwise, it is unbiased. The effectors include extro-effectors (those acting on the external world) and intero-effectors (those acting on internal components, e.g., attention selector between auditory and visual channels).

An algorithm is a developmental algorithm if it can conduct AA-learning (short for *automated animal-like learning* without claiming to be complete):

Definition 1 *A machine agent M conducts AA-learning at discrete time instances, $t = 0, 1, 2, ...$, if the following conditions are met: (I) M has a number of sensors, whose signal at time t is collectively denoted by $x(t)$. (II) M has a number of effectors, whose control signal at time t is collectively denoted by $a(t)$. (III) M has a "brain" denoted by $b(t)$ at time t. (IV) At each time t, the time-varying state-update function f_t updates the "brain" based on sensory input $x(t)$ and the current "brain" $b(t)$:*

$$b(t+1) = f_t(x(t), b(t)) \quad (1)$$

and the action-generation function g_t generates the effector control signal based on the updated "brain" $b(t+1)$:

$$a(t+1) = g_t(b(t+1)) \quad (2)$$

where $a(t+1)$ can be a part of the next sensory input $x(t+1)$. (V) The "brain" of M is closed in that after the birth (the first operation), $b(t)$ cannot be altered directly by human teachers for teaching purposes. It can only be updated according to Eq. (1).

As can be seen, AA-learning requires that a system cannot have two separate phases for learning and performance.

3. ARCHITECTURE

Our architecture SAIL (Self-organizing Autonomous Incremental Learner) is designed for a general agent. Fig. 1 gives a schematic

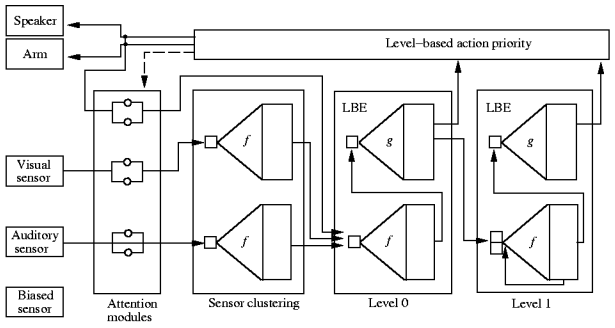

Figure 1: A schematic illustration of the coarse architecture of the learning mechanism. The attention modules perform sensor-specific preprocessing and attention selection. The sensor clustering layer performs online clustering of sensor inputs. Cluster centroids from different sensors are fused into a single vector and passed to the level 0 spatio-temporal associator (STA.) Cluster centroids from the level 0 STA are passed to the level 1 STA. Actions proposed by the different levels are arbitrated and then sent to the effectors.

illustration of the architecture of SAIL that we have tested. As shown in the figure, the input to the STA includes not just information from the sensors, but also the current control signal of the effectors.

3.1. Levels

We do not define architectural levels in terms of either domain knowledge hierarchy or system behavior hierarchy (both of which are used by task-specific approaches). Instead, our level corresponds to the extent of temporal context (a task-independent design).

The global state s of the "brain" $b(t)$ at any time t is represented distributedly by states at different levels: $s = (s_0, s_1, ..., s_L)$, where s_i, $i = 0, 1, ..., L$, represents the state at level i. The current number of levels is determined automatically based on the maturation schedule of the developmental algorithm which depends on the experience of the agent as well as its virtual age [1]. Level 0 is context free, to model S-R (stimulus-response) reflex. Starting from level 1, temporal context is incorporated. The higher the level i, the more temporal context each state at level i represents. The basic mechanisms of the level-building elements for each level are similar and thus level-building is automated.

3.2. States

An AA-learning algorithm must automatically generate states in a manner that is not task-specific, but can be sensor-specific. Let us first consider level 1 in Fig. 1. The part of the "brain" state at this level is denoted by a state vector $s(t)$ in a high dimensional space $\mathcal{S}$. Thus, our state has an explicit representation. $\mathcal{S}$ must contain all the possible sensory input $x \in \mathcal{X}$. State $s(t)$ is considered a

[1]The virtual age is the time of operation since the birth of the system.

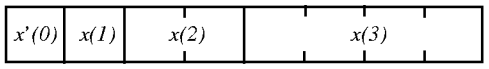

Figure 2: The state representation at $t = 3$ for 1-D signal with 2:1 uniform resampling. The symbols in each segment of the state indicate the main source of the signal.

random process. Eqs.(1) and (2) are closely related to the formulations for Markov decision processes (MDP) or HMMs (hidden Markov models). However, the states in MDPs typically are defined as a set of symbols. Thus, no distance metric is defined to measure the similarity between any two symbols.

In contrast to existing MDP methods, we require that the state records temporal context. Thus, we define the state space at level 1 recursively to be $\mathcal{S} = \mathcal{X} \times \mathrm{R}(\mathcal{S})$, where $\times$ denotes the Cartesian product and $\mathrm{R}(\cdot)$ denotes a re-sampling operator. The design of the re-sampling operator needs to take into account (a) the nature of the signal, (b) the desired temporal span in the state vector, and (c) the recursive relation $\mathcal{S} = \mathcal{X} \times \mathrm{R}(\mathcal{S})$. For example, for vector signal of 1-D nature, $R()$ can be a 2:1 uniform resampling operator. For image input, a 2-D 4:1 uniform resampling is used. For speech signal with Cepstrum vector $c(t)$, each state vector $S(t+1)$ may contain several frames: $c'(t), c''(t-1), c'''(t-2)$, to allow a state at low level to take into account more temporal context, where $c''(t-1)$ is subsampled from $s(t)$ and $c'''(t-2)$ from $s(t-1)$, etc. Since f is not an identity mapping, $c''(t-1)$ is not the same as $c(t-1)$.

The multi-level state transition function in Eq. (1) represents a simplified mapping $f : \mathcal{X} \times \mathrm{R}(\mathcal{S}) \mapsto \mathcal{S}$ at level 1. First, since the state space cannot be manually designed, we let f map $(x(t), \mathrm{R}(s(t)))$ directly to itself:

$$(x(t), \mathrm{R}(s(t)) = s(t+1) = f(x(t), s(t)). \quad (3)$$

In other words, the next state $s(t+1)$ keeps all the information of sensory input $x(t)$ and the re-sampled version of the current state $s(t)$. Given sensory inputs $x(0), x(1), ...$, this simplified f defines a trajectory of states $s(1) = (x(0), 0)$, $s(2) = (x(1), \mathrm{R}(s(1))$, and so on. Fig. 2 gives an illustration of a state at time $t = 3$.

Fig. 3 illustrates a part of the state trajectory that is automatically generated. Each state is associated with a number of actions through mapping $g(s)$ as in Eq.(2). Fig. 3 explains how the distance metric among states allows generalization of actions. This distance-based state representation also facilitates the following important functionalities: (1) States can be generated online as they are being recorded. (2) The distance metric in $\mathcal{S}$ makes it possible to access a huge number of states using a tree-based function approximator for real-time operation. (3) State clustering and forgetting can be naturally applied.

4. LEARNING TYPES

Eqs. (1) and (2) identify four components of the AA-learning agent for each time instance t:

$$(a(t+1), s(t+1), s(t), x(t)). \quad (4)$$

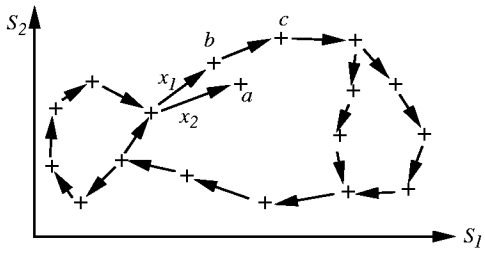

Figure 3: The finite state machine with dynamically generated and deleted states. Each "+" sign denotes a state. Each arrow indicates the time order. a is a newly generated state, which does not have any outgoing transition experience. The existing state b is the nearest neighbor of a in the state space $\mathcal{S}$. The transition path, from b to c, that is learned by b can be used for predicting the next state. The action choice for b, $g(b)$, can be used as the next action for a.

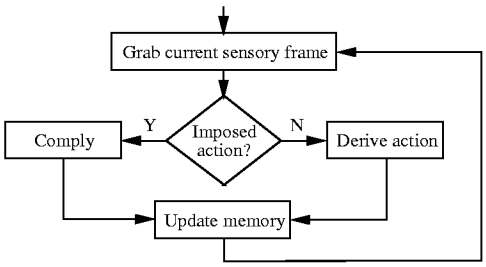

Figure 4: The flowchart for AA-learning. The system learns while performing.

They involve three entities: actions, states, and sensor inputs.

Depending on whether the action is imposed or not, the learning can be classified into *action-imposed* learning and *action-autonomous* learning. Depending on whether the biased sensor is used or not, the learning can be classified into reinforcement learning and communicative learning. *Reinforcement* learning is such that a *biased* sensor is used to reinforce or punish certain response from the machine agent. *Communicative* learning is such that only *unbiased* sensors are used in learning. This requires that the agent to correctly interpret the signal from unbiased sensors, either as an instruction for an action, an encouragement, an explanation, etc. Thus, the type of learning at any time t can be represented by a 3-tuple (A, S, X) where $A \in \{i, a\}$ denotes if an action is imposed or autonomous, $S \in \{i, a\}$ denotes if the state is imposed or autonomous, and $X \in \{r, c\}$ denotes if the biased sensor is used or not. There are 8 different 3-tuples, representing a total of 8 different learning types. AA-learning is state-autonomous learning. Thus, there are 4 types of AA-learnings: Type (1) action-imposed with reinforcement, Type (2) action-imposed and communicative, Type (3) action-autonomous with reinforcement, and Type (4) action-autonomous and communicative. It is worth noting that these four types are typically interleaved through time in a natural learning environment for animals and humans. Fig. 4 illustrates a flow chart of AA-learning. If the trainer imposes an action on an effector at any time through, e.g., a joystick, the system performs action-imposed learning for that effector. Otherwise, the system performs action-autonomous learning, during which reinforcement learning or communicative learning can be used.

4.1. Simple action-imposed learning

To facilitate understanding, we describe an oversimplified and thus very inefficient and weak version of action-imposed learning. Suppose that the machine agent M has recorded in memory $B = \{(x(i), s(i), a(i)) \mid i = 0, 1, ..., t-1\} \cup \{s(t), a(t)\}$. Note that $s(t), a(t)$ are the result from sensory input $x(t-1)$. According to the flow diagram in Fig. 4, M grabs the current sensory frame $x(t)$. Then, it computes the next state $s(t+1)$, e.g., using Eq. (3). If an action is imposed (e.g., in training session), $a(t+1)$ is supplied by a human being (or the environment) and thus M complies by sending $a(t+1)$ to the effector and then updates its memory by replacing B by $B \cup \{x(t), s(t+1), a(t+1)\}$. If an action is not imposed, M derives action $a(t+1)$ based on the past experience using a simplified g in Eq. (2) as follows. First, M finds the best matched state:

$$j = \text{argmin}_{0 \le i \le t} \|s(t+1) - s(i)\|. \quad (5)$$

Then, the output action is determined as the action associated with the best matched $s(j)$: $a(t+1) = a(j)$. The memory update is done as before. After $x(t+1)$ is grabbed in the next machine cycle, which may include the result of the action sensed by the sensor, the system memory becomes $B = \{(x(i), s(i), a(i)) \mid i = 0, 1, ..., t+1\}$.

As can be seen, this oversimplified version of AA-learning can do only a little generalization by extending the action learned by the nearest neighbor $s(j)$ (or multiple neighbors with action interpolation) from the current new state $s(t+1)$ whenever no action is imposed by the human.

4.2. Simple reinforcement learning

When no action is imposed, the learning is action-autonomous. The system generalizes using the nearest-neighbor rule. Such a generalization may or may not be good. Thus, a feedback signal in the range $[-1, 1]$ can be sensed by a biased sensor as a reward (positive or negative). An oversimplified reinforcement learning method incorporated into the above action-imposed learning algorithm is as follows: Modify the step of finding the nearest neighbor in Eq. (5) so that only the states whose corresponding action has received non-negative rewards are searched for. The other parts are the same.

5. A PRACTICAL ALGORITHM

To keep the memory size in dynamic balance, the states are automatically pulled together at each visit, merged when they are close, and deleted if they are not visited often. Our SHOSLIF [3], a regression tree algorithm [1] for high-dimensional space, can find top $k > 0$ matched vectors in logarithmic time in terms of the number clusters of input vectors. It is used to find the best matched

states as in Eq. 5 for both functions f_t and g_t in Eqs.(1) and (2), but the distance metric is not Euclidean.

6. SYSTEM AND EXPERIMENTS

A robot body for SAIL has been constructed at MSU as a testbed for our developmental algorithms. It has two micro-cameras (with pan-tilt control), four microphones, an arm with 5 degrees of freedom, a drivebase (for indoor and outdoor) and 13 pressure sensors on its body. However, for the need of examining the working of the developmental algorithm with full control, the experiments reported here are based on simulations using real sensory data.

We have tested two simulated architecture configurations, called "robot horse" (RH) and "robot receptionist" (RR), respectively, according to the tasks they have learned, although neither is limited to learning these tasks only. The architecture of RR is shown in Fig. 1 and that of AH is similar except that RH has no attention selection for simplicity. The major emphasis for RH is to study its capability to learn directly from sound waves using a microphone, contingent on another sensor (rein). The RR is to study its capability to learn directly from video images in conjunction with questions. RR has been trained to answer questions about name and gender when it sees individual human faces and to present the right presents to the right gender. Due to the space limit, we can only discuss AH in some detail.

RH has an auditory sensor and a numerical sensor, simulating its touch sense for rein when it is being pulled. In our prior work for autonomous navigation [4], we have used our framework SHOSLIF to implement a single function which maps visual input directly to a navigation update vector (heading, direction, and speed). It has been demonstrated that this tree-based mapping enables our Rome robot to navigate in real-time in our Engineering Building, using only a single sensor — a video camera. However, with this alone, the robot does not yet have a way to act interactively according to human verbal commands. Here, we are interested in adding an interactive mode for a "robot horse."

The rein sensor is used to teach the horse to listen and act only when the rein is pulled. When the rein is not pulled (simulated by 0 of the numerical sensor), it will roam along using the low-level reflex actions that have been demonstrated with the Rome robot. The verbal commands are four vowels: "a", "e", "i" and "u", representing "left," "right," "faster" and "slower," respectively. With multisensor fusion, in this case, the coupling of verbal signal with the rein signal, RH will less likely be confused by various background sounds in the environment.

We conducted speaker independent tests with five people: three adult men, one adult lady and one young girl. In other words, the persons who were tested were not among the persons whose sounds have been used in the training sessions. To get an average performance for RH, we conducted leave-one-out tests. In other words, five experiments were conducted, each with a new "birth." For each experiment, a different person's utterances were used in the test session. The other four persons' utterances were used for the training session. The response is considered correct if the correct action is produced immediately after the coupled rein and verbal command. Otherwise, the response is incorrect. During the training sessions, the desired action is imposed at the right context. For each action, a total of 20 training sounds (4 persons; 5 utterances from each person) are heard by RH. 5 utterances of each person are used for testing. The following table summarizes the test result recorded in the test sessions.

	Meaning	Man 1	Man 2	Man 3	Lady	Girl
"a"	left	100 %	100 %	100 %	100 %	100 %
"e"	right	100 %	100 %	100 %	100 %	100 %
"i"	faster	100 %	100 %	100 %	100 %	100 %
"u"	slower	100 %	100 %	100 %	100 %	100 %

To push the system at this young "age" to the limit, we added another vowel "o" to the vocabulary. It may represent a new action. It is known that the waveform and Cepstrum coefficient vectors of "o" and "u" are close. The result showed that about 4% of the "o" vowel sounds are recognized as "u" and the rest of cases are all correct. On the other hand, it is not always trivial for a human to perfectly distinguish individually pronounced "o" and "u." Complete words and sentence context can help greatly. Future studies along this line will test words and sentences using more levels of the proposed architecture.

7. CONCLUSIONS

Although the tasks SAIL has been trained to learn are relatively simple, the presented new concepts and the experimental results have demonstrated that the developmental algorithm can enable a machine to learn new tasks without a need of re-programming. The capability of such a machine will depend on its sensors, its effectors, its computational resources, its developmental algorithm, and how it is taught. It does not need humans to find a good task-specific representation for a task and to program for the task. Therefore, the developmental approach introduced here has a potential for dealing with tasks that are too complex for human to program effectively, including many multimedia sensing tasks that require different strategies in different contexts.

8. REFERENCES

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1993.

[2] L. R. Rabiner. Toward vision 2001: Voice and audio processing considerations. *AT&T Technical Journal*, 74(2):4–13, 1995.

[3] D. Swets and J. Weng. Discriminant analysis and eigenspace partition tree for face and object recognition from views. In *Proc. Int'l Conference on Automatic Face- and Gesture-Recognition*, pages 192–197, Killington, Vermont, Oct. 14-16 1996.

[4] J. Weng and S. Chen. Incremental learning for vision-based navigation. In *Proc. Int'l Conf. Pattern Recognition*, volume IV, pages 45–49, Vienna, Austria, Aug. 25-30 1996.