PERGAMON

AVAILABLE AT www.ComputerScienceWeb.com POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 701–710

Neural Networks

www.elsevier.com/locate/neunet

2003 Special issue

Autonomous mental development in high dimensional context and action spaces

Ameet Joshi[a,*], Juyang Weng[b]

[a]*Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824, USA*
[b]*Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA*

Abstract

Autonomous Mental Development (AMD) of robots opened a new paradigm for developing machine intelligence, using neural network type of techniques and it fundamentally changed the way an intelligent machine is developed from manual to autonomous. The work presented here is a part of SAIL (Self-Organizing Autonomous Incremental Learner) project which deals with autonomous development of humanoid robot with vision, audition, manipulation and locomotion. The major issue addressed here is the challenge of high dimensional action space (5–10) in addition to the high dimensional context space (hundreds to thousands and beyond), typically required by an AMD machine. This is the first work that studies a high dimensional (numeric) action space in conjunction with a high dimensional perception (context state) space, under the AMD mode. Two new learning algorithms, Direct Update on Direction Cosines (DUDC) and High-Dimensional Conjugate Gradient Search (HCGS), are developed, implemented and tested. The convergence properties of both the algorithms and their targeted applications are discussed. Autonomous learning of speech production under reinforcement learning is studied as an example.

Keywords: Autonomous mental development; High dimensional; Robotic system

1. Introduction and problem identification

Autonomous Mental Development (AMD) (Weng et al., 2001) is a new paradigm for developing autonomous machine. The machine is controlled by a new kind of program called developmental program ever since its *birth*. Although develop-mental program is different from a traditional program in many ways, the most fundamental difference is that the programmer does not know the tasks that the robot ends up learning after birth. Therefore a developmental program must be able to generate internal representation *on the fly* for virtually any task. The capability of the machine is developed through real time interactions with the physical world. It depends on the five constraints: (1) sensor, (2) effector, (3) computational resource, (4) developmental program and (5) the way the robot is taught.

The other challenges of an AMD robot include

1. Environmental openness.
2. High-dimensional sensors.
3. Completeness in using sensory information.
4. Online processing.
5. Real-time speed.
6. Incremental processing.
7. Perform while learning.
8. Muddy tasks.

There have been studies in humanoid control that involve high dimensional action space e.g. (Vijayakumar & Schaal, 2000; Billard and Mataric, 2000). These studies include strictly action space, without using the perception space. In other words, the robot is able to learn only one action trajectory, but is not able to produce different action trajectories under different contexts. Further, the studies in Vijayakumar and Schaal (2000) and Billard and Mataric (2000) are based on supervised learning. The explosion of both perception and action spaces creates a very practical but unaddressed challenging research issue. Compounding the challenge is the issue of learning modes. For such a high dimensional action space, supervised learning is often not practical, especially for internal actions (actions that are produced by internal effectors that are not reachable externally by human teacher). For example, an external action performed by an arm can be taught in a supervised

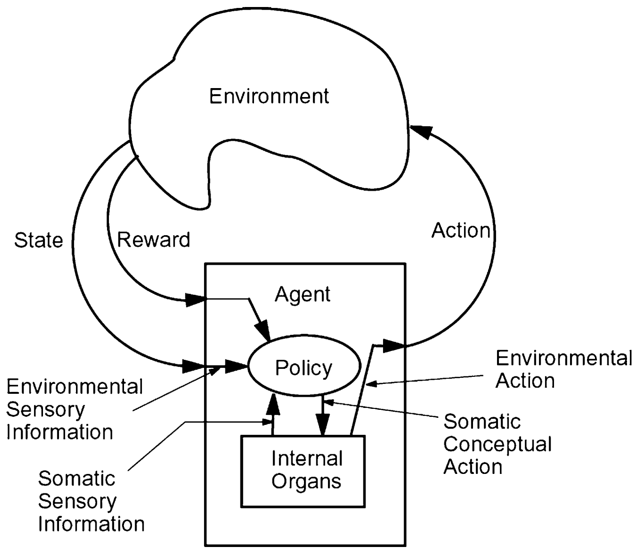
* Corresponding author.
E-mail addresses: joshiame@egr.msu.edu (A. Joshi), weng@cse.msu.edu (J. Weng).

0893-6080/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0893-6080(03)00134-5

mode by holding the learner's arm. However, an internal action such as speech production and internal attention cannot be taught in a supervised learning mode. We have to use reinforcement learning. These compounding challenges raised a series of research issues. This paper mainly deals with AMD learning in high dimensional context and action spaces using generalized reinforcement learning mechanisms. Section 2 deals with the modified reinforcement learning model. Sections 3 and 4 explain the methods DUDC and HCGS in detail and Section 5 discusses the results obtained by applying the two methods on real-time robotic system.

2. Modified reinforcement learning

Traditional reinforcement learning deals with modifying a value system attached to the actions based on the rewards (Sutton, 1988; Sutton & Barto, 1998) and possesses a distinct boundary between the environment to be served and the agent (Kaelbling, Littman, & Moore, 1996). However, in the context of humanoid robot, this model is inadequate. The humanoid robot has a body consisting of locomotory organs like hands and feet, sensory organs like ears and eyes and a brain. In order for a robot to function properly, its brain should have precise knowledge of the relative positions of the organs and their limitations. This is achieved by getting internal sensory information from the organs about their states. Fig. 1 shows a picture of the modified reinforcement learning system used in this project. Complete state information is obtained by a combined set of external (environmental) and internal (somatic) sensory state information. The learning policy uses the context retrieved from memory to choose a new action. Traditionally, Q learning is used to produce the desired actions from the current state information. However, due to inadequacy of this method in high dimensional state and action spaces, two new methods, DUDC and HCGS are developed.


Environment
State
Reward
Action
Agent
Policy
Internal Organs
Environmental Sensory Information
Somatic Sensory Information
Environmental Action
Somatic Conceptual Action
Fig. 1. The modified reinforcement learning model.

3. Learning with direct update on direction cosines (DUDC)

The most important aspect of this method in regards with the standard Q learning is the removal of discrete actions. The action space is high dimensional and numeric. Each action is represented by a single set of direction cosines, $\Psi(i)$ and a step size parameter δ. The direction cosines form a unit vector that specify the direction and the magnitude of the distance traveled is given by the step size δ. The step size is reduced hierarchically based on rewards. The learning of the direction cosines starts afresh with every change in the step size δ. A set of direction cosines $\Psi(i)$ represents an attempt towards the optimal direction in the given search space and in the given hierarchy of step size δ. The objective of the reinforcement learning is to find the optimal set of direction cosines $\Psi(i)$. This explains the quantization imposed on the search space. The value update rules and the policy of action generation are now discussed.

3.1. The value update rule

The value update rule in DUDC is given below.

$$\Psi(i) = \Psi(i) + \alpha r(\Psi'(i) - \Psi(i)) \tag{1}$$

The value of 0.1 is normally used for α, which is known as learning rate. r is the reward obtained from the environment.

3.2. EMMP and action generation

EMMP stands for 'Exploration with Maturation using Multidimensional Perturbation'. Exploration means random change in the given direction. For exploration to be ideal, all the possible directions should be equally likely. The concept of maturation comes with a tradeoff between exploration and exploitation. The direction cosines $\Psi(i)$ are shaped with reinforcement learning. Direct use of direction cosines implies the exploitation. Exploration leads to finding the global optima and exploitation leads to a local optima. Exploration is slowly reduced as the system matures. The mathematical model that is developed with this concept is described below.

$$\kappa = \log\left(1 + \frac{\tau}{40}\right) \tag{2}$$

$$\Omega(i) = \epsilon + \Psi(i)\kappa \tag{3}$$

$$\xi = \left(\sum_{i=1}^{i=10} \Omega^2(i) \right)^{1/2} \tag{4}$$

$$\Psi'(i) = \frac{\Omega(i)}{\xi} \tag{5}$$

With respect to above equations, ϵ is a real valued random number between -1 and $+1$ such that it can take all the values in the range with equal probability. τ is the iteration number, which is incremented with each arrival of context. The number 40 is chosen from the empirical evaluation of various other values. The first step calculates κ, the quantitative measure for the maturity of the system from the number of iterations. The next three steps describe how the new set of direction cosines $\Psi(i)$ is obtained from the old one using the EMMP method. The weights, denoted as $\Omega(i)$, are the intermediate variables used before the normalization. ξ is the normalizing variable. This method uses a variation of the exploration and exploitation tradeoff, which is based on the principles of Boltzmann Exploration. The $\Psi(i)$ represent the original direction cosines and the $\Psi'(i)$ represent the new direction cosines obtained after doing the exploration.

The complete action is obtained by multiplying the direction cosines with the step size δ. The choice of the δ is crucial in the working of the method. For a fixed step size, there is limitation on the maximum proximity towards the target. Hence it is essential to have a hierarchical step size variation which will enable the coarse-to-fine search. With the change of step size the previously learned direction becomes useless, as can be seen from Fig. 2.

3.3. Robust learning

The EMMP method finds the near optimal direction towards the target under the constraint of the given step size. With each reduction in the step size δ, the search becomes finer and scope of the search also reduces. The reduction in δ is solely controlled by the rewards, the details of which are discussed in Section 3.4. In order to take into account the possible errors in the rewards, and make the algorithm converge in spite of erroneous rewards, the following rule is applied to change the step size.

Reversible step size change algorithm:

(1) Initialize δ.
(2) Initialize counter for bad rewards.
(3) Use EMMP for updating the $\Psi(i)$ also keep updating the bad reward counter.
(4) If the number of bad rewards exceed the threshold then increase δ and restart the EMMP.
(5) Stop.

The above algorithm makes sure that the EMMP is not stagnated at certain point in the search space due to the acquisition of a erroneous reward leading to unwanted reduction in δ. This makes the algorithm robust.

3.4. Considerations about rewards

The rewards can be relative or absolute. In former case, a Good reward means that the performance of the system is improved compared to the previous attempt, while in the latter case a *Good* reward means that the performance of the system is within certain predetermined bounds irrespective of the previous performance. In current framework, both types of rewards are expected by the system. The rewards are of three types, *Good reward*, *No reward* and *Bad reward*. The *Good* reward is considered as absolute, and the latter two rewards are considered as relative. A *Good* reward means that the search has reached sufficiently close to the target and the step size can be reduced. A *Bad* reward means that the performance is getting worse compared to previous performance, and the *No reward* means the performance is unchanged or that it has improved compared to the previous attempt, but still not within the predetermined bounds. This structure of rewards is constructed based on real time evaluation, where the rewards are obtained from the human teacher. The formal algorithm of the entire DUDC method is presented below.

DUDC algorithm:

(1) Initialize the start point.
(2) Initialize the direction cosines in all the dimensions. The values are normalized with the norm as unity.
(3) Initialize number of iterations, $n = 1$.
(4) Start iterations.
(5) Get the reward from the iteration.
(6) Use DUDC and EMMP to update $\Psi(i)$ and δ.
(7) Take the action.
(8) Increment the iteration number $n = n + 1$.
(9) Go to step 5. (No Stop.)

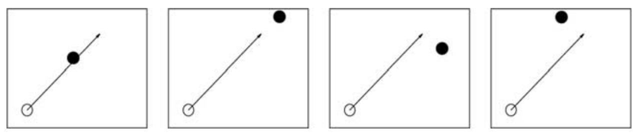


Fig. 2. The solid circle represents the target and the hollow circle is the starting point. The arrow denotes the direction of the previous action and its end denotes the new start position. The different variations of the positions of the new starting point and the target are shown. Although in each case the distance from the target is same, the direction towards the target is entirely different from the starting direction.

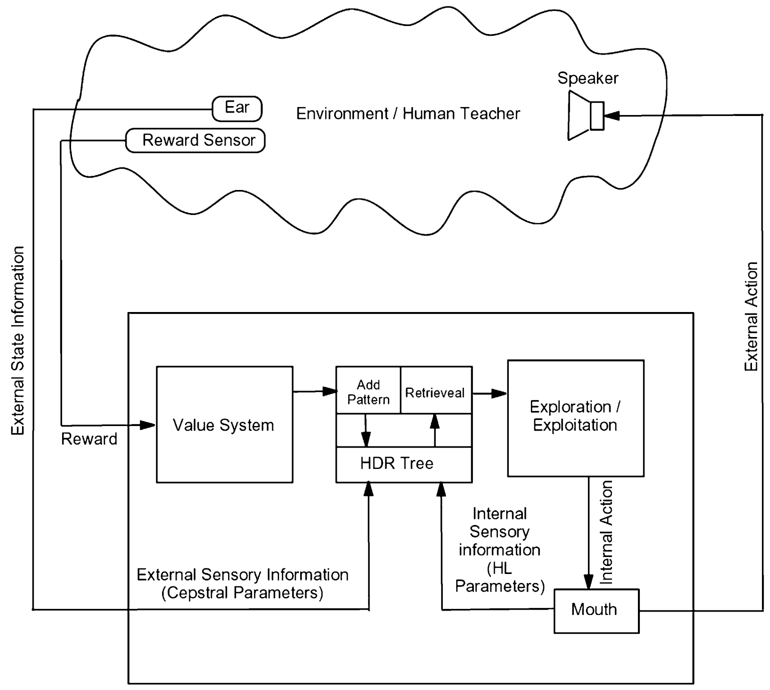

Ear
Reward Sensor
Environment / Human Teacher
Speaker
External State Information
External Action
Reward
Value System
Add Pattern
Retrieveal
HDR Tree
Exploration / Exploitation
Internal Action
Internal Sensory information (HL Parameters)
External Sensory Information (Cepstral Parameters)
Mouth

Fig. 3. The system architecture with DUDC.

The system architecture using DUDC is shown in Fig. 3.

4. Learning with high dimensional conjugate gradient search (HCGS)

In this approach, the continuous action space is quantized to form a discrete action space. A variation of vector quantization, called Amnesic Vector Quantization (AVQ) is used.

4.1. AVQ in action space

Using a suitable threshold, actions are quantized into micro-clusters. Each micro-cluster represents a single action and possesses a Q-value. Each context is limited to have a specific number of micro-clusters of actions. The mean value that represents a micro-cluster is updated using the method of Amnesic Average. The amnesic update rule is given in Eq. (6).

$$R^{(n+1)} = \frac{n-1-l(n)}{n}R^{(n)} + \frac{1+l(n)}{n}V \tag{6}$$

In traditional update, where $l(n) = 0$ for all n, with accumulation of large number of actions the contribution of new action is reduced to infinitesimal. This is not desirable, and all the system properties need to change with time. The parameter $l(n)$ is an amnesic parameter and is defined as,

$$l(n) = \begin{cases} 0 & \text{if, } n \le n_1 \\ 2(n-n_1)/(n_2-n_1) & \text{if, } n_1 < n \le n_2 \\ 2+(n-n_2)/m & \textit{if, } n > n_2 \end{cases} \tag{7}$$

When the number of updates is less than n_1, the update is same as traditional average. However after that the amnesic parameter $l(n)$ is designed in such a way that more priority is given to the incoming action vector. After n_2 updates the weight of the incoming sample is changed again and it increase with a rate $1/m$. $R^{(n)}$ and $R^{(n+1)}$ denote the nth and $(n+1)$th updated vector, respectively, and V denotes the new action taken in $(n+1)$th iteration. As the system continues the learning process, the micro-clusters are developed and their Q values are updated. These micro-clusters are used to generate the interpolation function for Q value in the composite context of state and action space.

4.2. Interpolation using Q-values

In order to use the Conjugate Gradient (CG) Method (Shewchuk, 1994; Fletcher & Reeves, 1964) to decide the optimal action based on the Q values, it is essential to have an interpolation function representing the distribution of the Q values in the context-action space. CG method needs the interpolation function to be smooth and have a single well-defined local minimum. A new interpolation method with these properties is designed, denoted as 'Density Sensitive Kernel Interpolation' (DSKI). The equation for the interpolated function $G_n(x; x_1, x_2, \ldots, x_n)$ is given below

$$G_n(x; x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i(x)y(i) \tag{8}$$

Here $x_i \in R^d$, $y_i \in R^k$ and $x_1, x_2, \ldots, x_n$ are n nearest neighbors of $x \in R^d$ from S. S is a set of micro-clusters in R^d representing the approximation of reciprocal of density in R^d. S can also be called as a set of finite number of neurons for each state. The definition of the weights w_i is based on the squared local sparseness σ^2, which is defined as,

$$\sigma^2 = \frac{k}{n} \sum_{i=1}^{n} \|x_i - x\|^2 \tag{9}$$

where, k is called as *kernel variance factor*. The more the value of k, the more is the kernel variance and more flat the interpolated function is. The expression for weights is,

$$w_i = C \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2} \right), \quad i = 1, 2, \ldots n. \tag{10}$$

C is computed such that $\sum_{i=0}^{n} w_i = 1$. $y_i = f(x_i)$ is the function to be approximated. In this case, y_i denotes the Q-value of the action. The typical value of n is 3. The action list contains about 30 samples. The top-11 actions out of 30 existing action micro-clusters are chosen to interpolate the function. Due to inherent limitations of linear line minimization methods they cannot be used to give optimal performance in all the situations. To preserve the generality of the algorithm, line minimization part of the CG algorithm is removed. The initial guess for the step size is calculated based on the density of the samples at the starting point.

4.3. Value update using Q-learning

A queue of n recent states is maintained in memory, called a Prototype Updating Queue (PUQ). With each incoming reward, Q values of the states in PUQ are updated using Eq. (11).

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha(\gamma V) \tag{11}$$

where V is the maximum Q value in the currently retrieved state for the first element of PUQ, while for the succeeding states it represents the updated Q value of the previous element in PUQ.

4.4. Boltzmann exploration

The subtle difference between the list of action micro-clusters and the traditional set of actions is that the traditional set encompasses all the possible actions, while the current set does not. The actions outside the list of action micro-clusters can also be taken and they can be even better than the ones in the list. Hence a separate random exploration is used along with the choice of exploitation using CG search. The random exploration uses the upper and lower bounds on the action space along with the inter-dependencies among dimensions of action space to generate a random action. These two choices are feeded to the Boltzmann exploration procedure with probabilities which are based on the status of the action list and maturation.

4.5. Reorganization of the action micro-clusters

During the initial phase of creation of the action micro-clusters, a default distance threshold is used. If the incoming sample has a distance more than the threshold distance from all the existing micro-clusters, a new cluster is created for it. When the list of action micro-clusters is full, the existing micro-clusters are updated with new samples. With each update, the center of the micro-cluster moves towards the new sample. The amnesic average makes sure that even after large number of updates the new samples still make noticeable impact on the micro-cluster in which it is being added.

It is observed that, with the exploitation superseding the exploration, some of the micro-clusters being closer to the target are updated more often than others. Also, due to this, most of the micro-clusters are under utilized. As an implication of this, the required uniform convergence is not observed. After the search reaches certain proximity level towards the target, the algorithm stagnates. In order to address this issue, the re-organization of the micro-clusters is performed after the context is repeated certain number of times.

Let there be n be the number of micro-clusters and $\rho(i)$ denote their centers. The number of updates on ith micro-cluster be $\phi(i)$. The weighted mean of the micro-clusters is given by,

$$\Lambda = \frac{\sum_{i=1}^{i=n} \rho(i)\phi(i)}{\sum_{i=1}^{i=n} \phi(i)} \tag{12}$$

All the micro-clusters are pulled towards it, based on the number of times each one is visited. The update rule for the micro-cluster center is given by,

$$\rho'(i) = \frac{\phi(i)\rho(i) + k\Lambda}{\phi(i) + k} \tag{13}$$

where $\rho'(i)$ is the updated value of initial micro-cluster center $\rho(i)$. The k is a parameter specifying the weight of the Λ and is decided empirically. This is followed by reduction in the search space. Initial search space is denoted by S such that $\rho(i) \subset S$, $i = 1, \ldots n$. The new search space is given by S', such that, $\rho'(i) \subset S'$, $i = 1, \ldots n$. Also, $S' \subseteq S$.

The reduction in search space ensures that the random exploration is restricted to explore the region which is more likely to contain the global optima. After the reorganization of micro-clusters, number of updates on each micro-cluster is initialized along with their Q-values, $\phi(i) = 0$ and $Q(i) = 0$. This starts the new search in the reduced space. This mechanism basically represents the hierarchical coarse to fine search.

In another approach the top-k clusters with highest Q values are selected and remaining clusters are removed. They are then filled with the new incoming samples.

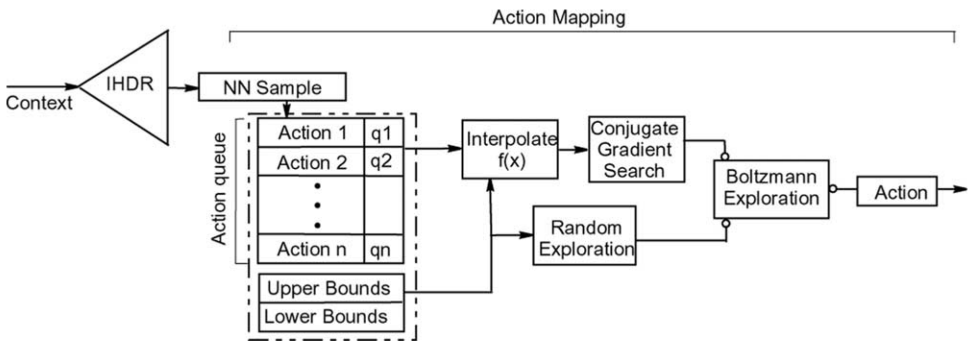


Fig. 4. The learning with HCGS.

The search boundaries are also reduced along with, in order to restrict the random exploration.

The partial block diagram to show the functioning of HCGS, is shown in Fig. 4. The flow chart of the system representing both the approaches is shown in Fig. 5 (Zhang & Weng, 2002). This work is part of the SAIL project, where an entire humanoid robotic system is being developed capable of having vision, and locomotion. The basic block diagram of the system as part of SAIL is shown in Fig. 6. In the entire block diagram of SAIL, the block of sensory mapping consists of all the sensory organs including the vision and locomotion. The details of the system can be found in Zhang and Weng (2001).

5. Results and discussion

We applied the above general mechanisms to a challenging high dimensional internal action learning example: *learning of speech production based on sensory*

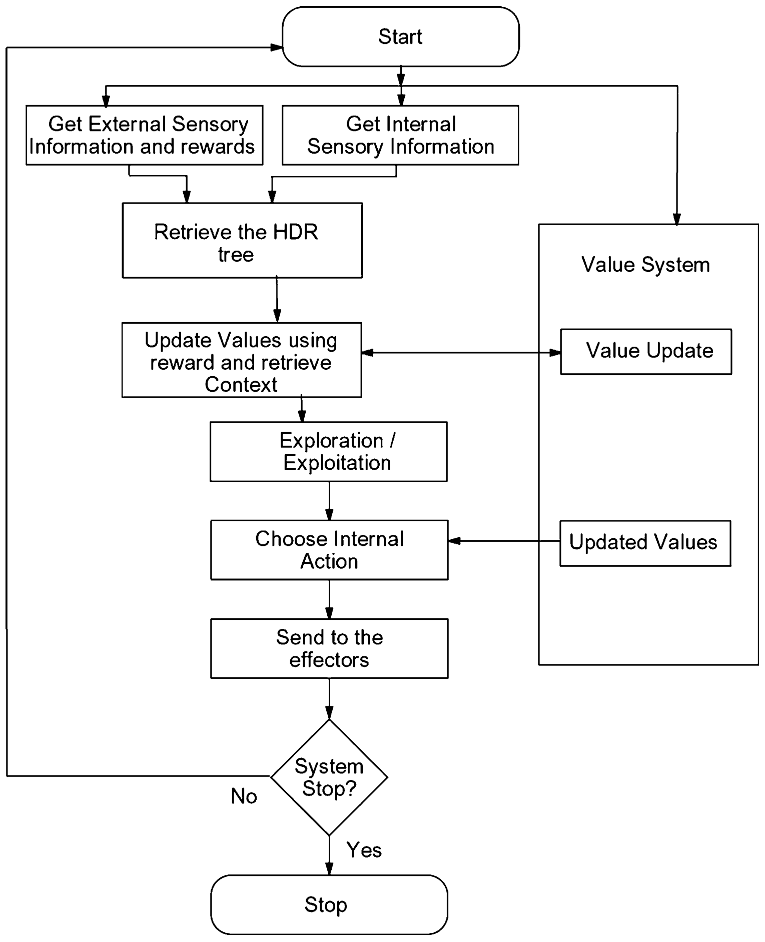

Start
Get External Sensory Information and rewards
Get Internal Sensory Information
Retrieve the HDR tree
Update Values using reward and retrieve Context
Exploration / Exploitation
Choose Internal Action
Send to the effectors
System Stop?
No
Yes
Stop
Value System
Value Update
Updated Values

Fig. 5. The flowchart of the program.

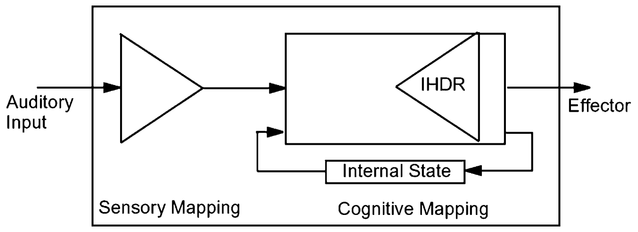


Fig. 6. The block diagram of the system as part of the entire SAIL architecture.

context. No task specific information is programmed into the AMD program. For example, if the teacher wants the robot to repeat what he has pronounced, all he can do is to teach him by giving rewards, as the *repeat* mechanism is not preprogrammed.

5.1. The system architecture

The system based on the architecture as given in Fig. 1, consists of three components, the ear, the mouth and the brain. The mouth is implemented with HL parameters obtained from the Klatt model (Perkell and Klatt, 1986; Stevens and Bickley, 1990; Allen, Hunnicut, & Klatt, 1987; Inc. 1997). For the implementation of the ear, cepstral parameters are used. The raw raw speech (O'Shaughnessy, 1987) is converted into cepstral parameters which are then used by the brain. The temporal behavior of these components is explained in the Fig. 7. The system works with time increment of one second. This restriction arose from the considerations of the parallel operation of the system on a single processor PC and use of multimedia sound card. It should be noted that this restriction does not hamper the basic principle of parallel architecture, as with dedicated components the same algorithm can be extended to arbitrarily small time interval, matching human behavior.

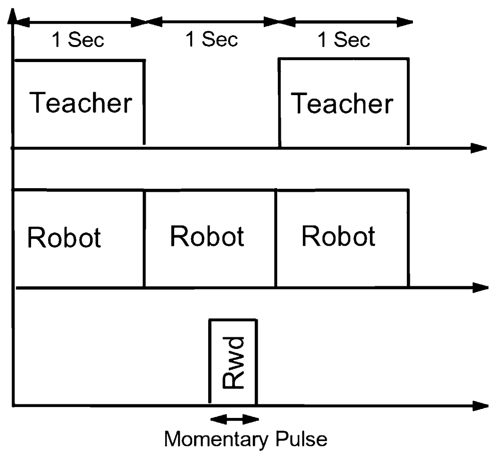


Fig. 7. The temporal behavior of the system.

5.2. Context based learning

The objective here is to respond the voice commands by the teacher. When teacher tells the system to speak certain vowel the system responds to it. The teacher trains the system by providing rewards. For example, the system can be trained to imitate /a/ after the teacher and also it can be trained to speak /e/ when teacher speaks /o/. This generalization makes the learning task independent and is entirely in the hands of the teacher. The rewards shape the system towards desired behavior. The learning is non-supervised (the internal parameters are never controlled by the teacher, no labels are provided), as ideal human learning should be. The details of the context parameters and their acquisition is explained in Section 5.3.

5.3. Obtaining context and normalization

The parameters taken from the ear are cepstral parameters. A set of cepstral parameters correspond to the sound acquired in a time window of 20 ms which is accurate within the bounds (O'Shaughnessy, 1987). The ear continuously keeps converting the incoming raw speech into the cepstral parameters and maintains a queue. The brain has access to this queue for further processing. Out of the ten HL parameters directly obtained from the mouth as internal (somatic) sensory information five parameters are formant frequencies. Formant frequencies possess a considerable overlap among different vowels as can be seen from Fig. 8. To ensure that the desired importance is given to each set of parameters in deciding the context, both the sensory parameters are normalized using their variance obtained from a sample data set.

5.4. Storing and retrieval of the context information

Another important aspect in the working of brain is the storage and retrieval of the sensory information. Brain memory is associative and takes a shape of a statistically generated tree structure, where all the information is stored according to the statistical similarity. The data structure used in this project to model memory is called IHDR (Incremental Hierarchical Discriminant Regression) (Hwang & Weng, 2000; Weng & Hwang, 2000).

5.5. Testing with DUDC

Testing with DUDC is performed in two stages. In the first stage the search algorithm is tested on a variable dimensional synthetic data. Dimensions varying as 2, 5 and 10. Different locations of the target point relative to the starting point are chosen and the results are tabulated in the Table 1. Each experiment is conducted for 100 iterations. The table shows the average values of iterations required for convergence, for each dimensional data. The convergence is defined as the reduction in the distance to the target by factor

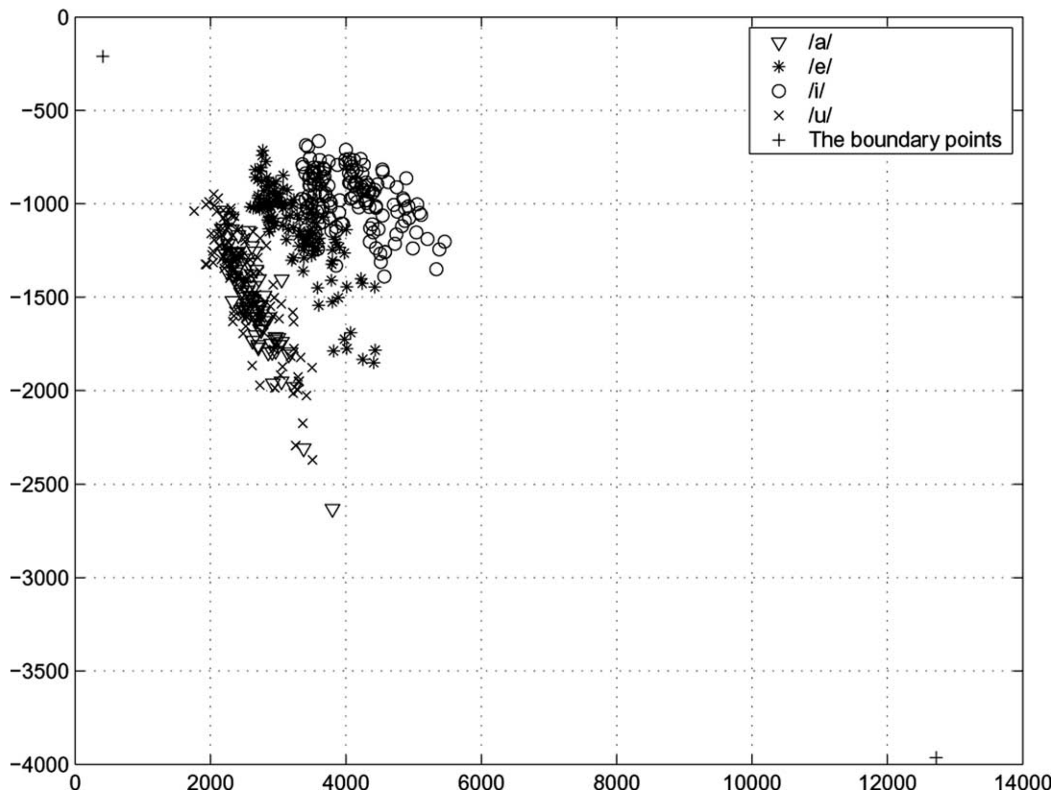


Fig. 8. This figure shows the distribution of the four vowels in the reduced dimensions (using PCA). The point in the left top corner represents one end of the search space and the point in the right bottom corner represents the other end.

of 20 from the starting distance, or 5% proximity compared with the starting point. As expected there is a monotonic rise in the number of steps needed for the convergence with the increase in the dimensionality.

After the first stage of testing with simulated data, the system is tested with a *Synthetic Teacher* (ST). ST is a computer program executing in conjunction with the main system and has access to the internal parameters to generate the rewards. ST is designed to replace the *Human teacher* (HT) for the purpose of automating the process of learning. The rewards given by ST are computed from the euclidian distance to the target and are always accurate. The convergence behavior of the system with ST is shown in Fig. 9. However, it should be noted that, although the ST has access to internal parameters of the system, the internal parameters are not directly altered by it and hence the this framework still satisfies the conditions of the non-supervised learning.

5.6. Testing with HCGS

The testing framework is similar to the one used in DUDC with the use of ST. The number of micro-clusters *nm* in the action list is a crucial parameter in the performance of HCGS. The number of micro-clusters is varied from 10 to 100 and the convergence plots for $nm = 30$ and $nm = 100$ are shown in Fig. 10. The convergence with $nm = 30$ is better compared to $nm = 100$. In either case, the system starts converging and after certain point the convergence gets stagnated. The reason behind this can be explained with following reasoning. Initially as the micro-clusters are formed by random exploration, some of the clusters are

Table 1
The table displaying the average learning rate with variable dimensions and variable error rates in rewards

Dim	Error rate	Target point location	Average no of steps
2	0%	Center	29.91
		Boundary	3.44
		Random	54.48
	25%	Center	35.24
		Boundary	3.84
		Random	44.32
5	0%	Center	257.37
		Boundary	55.66
		Random	319.18
	25%	Center	303.49
		Boundary	50.12
		Random	856.04
10	0%	Center	1248.22
		Boundary	445.27
		Random	1089.82
	25%	Center	1844.08
		Boundary	717.64
		Random	2279.45

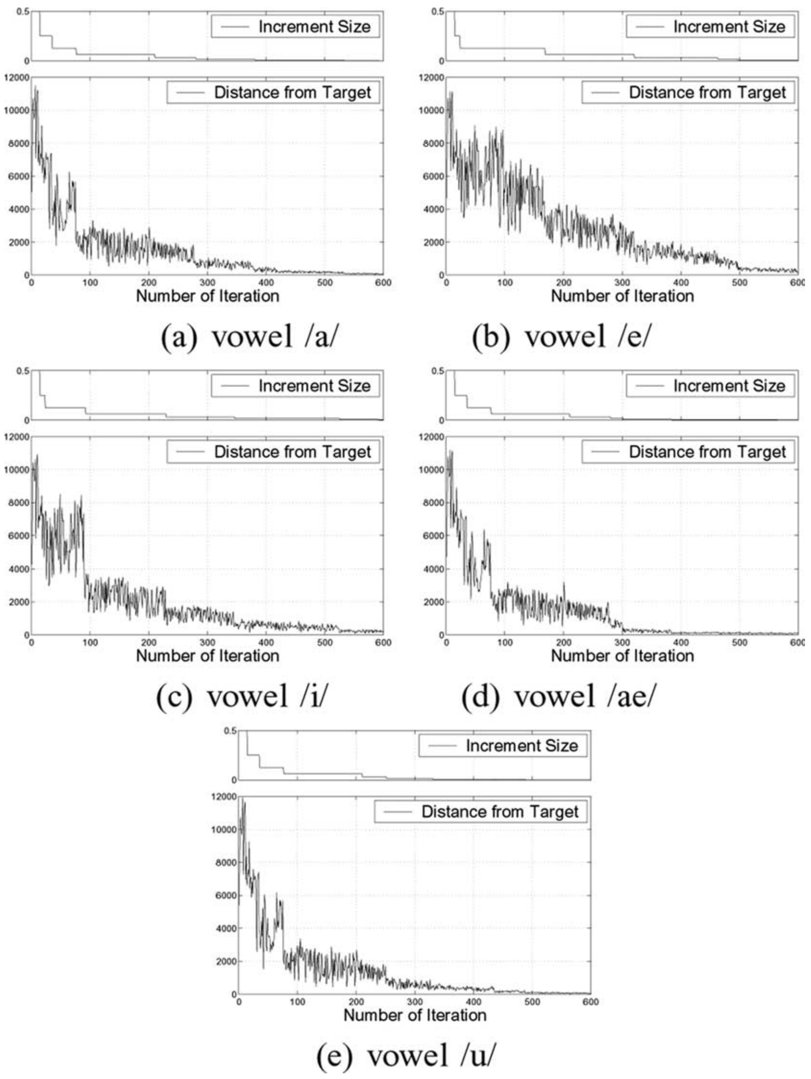


Fig. 9. The convergence plots generated during the learning of different vowels during 600 iterations using ST. The context used in this testing was the direct internal sensory information in the form of HL parameter.

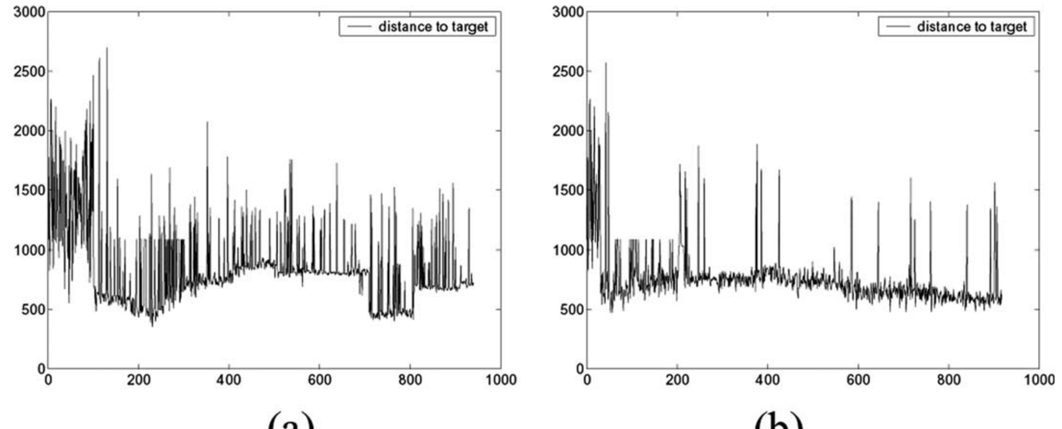


Fig. 10. (a) Shows the convergence with number of micro-clusters as 100 and (b) shows the convergence with number of micro-clusters as 30.

close to the target. With the progress of learning and the movement of micro-clusters, the micro-clusters that are close to the target, are chosen more often and the other clusters remain under-utilized.

Comparing the above results, the DUDC method appears to be superior to the HCGS in computational complexity and rate of convergence. However, it is left as future work to explore the reorganization of the micro-clusters for performance improvement in HCGS. Although current work is targeted towards learning a single action, the generation of smoother transitions from one action to another can be easily obtained using the combination of action and perception space.

6. Conclusions

The methods developed in this project appear promising towards tackling the problem of autonomously developing a robotic system capable of learning to produce high dimensional (e.g. 10D), action interactively and autonomously. The techniques designed and implemented in this work seem capable of realizing the initial development of basic, early behaviors in a high dimensional space through the AMD mode.

In the later stages of the speech learning development the system shows goal-directed behaviors, which facilitates faster learning. It is marked by a representation of the goal from the early learning experience, using the goals to activate actions, changing direction explicitly (e.g. understanding the goal by biting down on one's tongue and giving it a few tries).

The current work is, however, new and very important in bootstrapping higher level goal directed learning in the later development stage.

References

Allen, J., Hunnicut, S., & Klatt, D. (1987). *From text to speech: The MITalk system*. Cambridge: Cambridge University Press.

Billard, A., Mataric, M (2000). *Learning human arm movements by imitation: Evaluation of a biologically-inspired connectionist architecture*. Cambridge, MA, September 7–8.

Fletcher, R., & Reeves, C. (1964). Function minimization by conjugate gradients. *Computational Journal*, *7*, 149–154.

Hwang, W., & Weng, J. (2000). Hierachical discriminant regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1277–1293.

Kaelbling, L., Littman, M., & Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

O'Shaughnessy, D. (1987). *Speech communication*. New York: Addison-Wesley.

Perkell, J., & Klatt, D. (1986). *Invariance and variability in speech processes*. Hillsade, NJ: Lawrence Erlbaum Associates.

Sensimetrics Inc (1997). *High level parameter speech synthesis system*. Technical Report, Somerville, MA: Sensimetrics Corporation.

Shewchuk J (1994). *An introduction to the conjugate gradient method without agonizing pain*. Technical Report, Carnegie Mellon University.

Stevens, K., & Bickley, C. (1990). Constraints among parameter simplify control of klatt formant synthesizer. *Phonetics*, *1*.

Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R., & Barto, A. (1998). *Reinforcement learning, an introduction*. Cambridge, MA: MIT Press.

Vijayakumar, S., Schaal, S (2000). *Real time learning in humanoids: a challenge for scalability of online algorithms*. Cambridge, MA, September 7–8.

Weng, J., & Hwang, W. (2000). An incremental learning algorithm with automatically derived discriminating features. *Proceedings of the Fourth Asian Conference on Computer Vision*, *22*(11).

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., & Thelen, E. (2001). Autonomous mental development by robots and animals. *Science*, *January*.

Zhang, Y., & Weng, J. (2001). Grounded auditory development by a developmental robot. *Proceedings of International Joint Conference of Neural Networks*, 1059–1064.

Zhang, Y., & Weng, J. (2002). Action chaining by a developmental robot with a value system. *Proceedings of International Conference on Development and Learning*.