Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2008) 1449–1461
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Error-driven active learning in growing radial basis function networks for early robot learning

Qinggang Meng[a,*], Mark Lee[b]

[a]Department of Computer Science/Research School of Informatics, Loughborough University, LE11 3TU, UK
[b]Department of Computer Science, University of Wales, Aberystwyth SY23 3DB, Wales, UK

Received 22 December 2005; received in revised form 11 May 2007; accepted 18 May 2007
Communicated by G.-B. Huang
Available online 20 June 2007

Abstract

In this paper, we describe a new error-driven active learning approach to self-growing radial basis function networks for early robot learning. There are several mappings that need to be set up for an autonomous robot system for sensorimotor coordination and transformation of sensory information from one modality to another, and these mappings are usually highly nonlinear. Traditional passive learning approaches usually cause both large mapping errors and nonuniform mapping error distribution compared to active learning. A hierarchical clustering technique is introduced to group large mapping errors and these error clusters drive the system to actively explore details of these clusters. Higher level local growing radial basis function subnetworks are used to approximate the residual errors from previous mapping levels. Plastic radial basis function networks construct the substrate of the learning system and a simplified node-decoupled extended Kalman filter algorithm is presented to train these radial basis function networks. Experimental results are given to compare the performance among active learning with hierarchical adaptive RBF networks, passive learning with adaptive RBF networks and hierarchical mixtures of experts, as well as their robustness under noise conditions.

Keywords: Biologically inspired robotics; Active learning; Hierarchical adaptive radial basis function networks

1. Introduction

In many situations, artificial systems need to work in unstructured and unpredicted environments with uncertainties, such as dangerous, hostile or inaccessible environments. Autonomy, self-learning and self-organizing are crucial for these systems as we cannot program them to cope with all situations. In spite of recent advances in this area, the systems implemented are far from achieving high level performance and task flexibility, and the studies are still in their infancies. On the other hand, humans, even infants, can gradually adapt to different tasks and environments without too much effort by learning and re-organizing the cognitive mappings, etc. in their brains. This mechanism is just what artificial autonomous systems need. Therefore, great research effort has been made in recent years to develop biologically inspired artificial systems [1,8,35,59], in order to build the links among neuroscience, psychology and intelligent systems, and allow such systems to have human-like intelligence in some degrees. These studies can also provide a testbed for biology and psychology researchers to verify their ideas. Developmental robotics [32] is one of such research areas, aims to investigate how an autonomous robot system incrementally builds its sensorimotor coordination abilities from the very beginning, which is greatly inspired by developmental psychology and neuroscience.

In early robot learning, several mappings need to be set up for sensorimotor coordinations which build the links between sensory information and motor values such as eye/head/hand coordination, and transformation of sensory information from one modality to another and from one frame of reference to another, such as arm forward/inverse kinematics and information transformation among

*Corresponding author.
E-mail addresses: q.meng@lboro.ac.uk (Q. Meng), mhl@aber.ac.uk (M. Lee).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.05.012

eye-centered, head-centered, body-centered and hand-centered representations [4,18] for a humanoid robot system. Generally, these mappings build the links between a m-dimension space to another n-dimension space. These two spaces may or may not have the same number of dimensions, but the mappings, which links the two spaces, are usually nonlinear. Automatically building these mappings involves growing, shrinking and adjustments of neurons in the mapping network, similar to infant brain growth during cognitive development [53].

Traditional sensorimotor mapping learning algorithms treat the learning system as a passive recipient of training data and use these training data to build the mapping; while human beings adapt to different ways of learning by actively interacting with their environment, and by observing the consequences of such actions. This is called active learning, and it has been proven to be more efficient compared to passive learning [9,36]. Active learning optimizes the learning process by selecting properly the next actions or next queries [9,56]. We use active learning very often in our life, from basic skill learning of infants to more specific skill training of athletes, and from curiosity-driven active exploration and learning [47] to student active learning in groups by raising questions and then discussing rather than just listening to their teachers in class.

In this paper, we are interested in biologically inspired sensorimotor mapping algorithms for robot learning at a very early stage, specifically, we investigate an error-driven active learning method in growing radial basis function networks for robot sensorimotor coordination learning. We use arm inverse kinematics, i.e. learn to predict the joint values from a given arm endpoint, as a testbed for our algorithm as it is a typical nonlinear mapping problem, and the idea discussed here can be applied to other robot sensorimotor mappings and coordination transformation problems such as eye/head/hand coordination control. Also arm control and learning is an important research topic in neuroscience and other areas [4]. In this paper, a plastic radial basis function (RBF) network is used as the computational substrate for automatically constructing a sensorimotor mapping network. We refer to plasticity in the mapping network as the result of two forms of change: an increase or decrease in the number of neurons or nodes in the network; or a change in an existing node's parameters, either as a shift of location of the covering field or a change in size of that coverage. These two kinds of change in the network have different mechanisms but both represent plasticity for growth and development, similar to that reported in neuroscience [53]. This plastic RBF network differs from the traditional self-organizing map (SOM) of Kohonen [26] and similar artificial neural networks [58] which need to predefine the network structure and the number of nodes in the network. The learning error distribution over the input space is an important resource indicating the difficulty of learning at each point in the working space and can be used to guide active learning. In this paper, an error-driven active learning approach is introduced to explore these large error clusters further, and local growing RBF subnetworks are used to cover the related subdivisions of input space in a hierarchical way.

In the following, Section 2 overviews the whole learning system architecture and its main components. Section 3 describes the plastic RBF and its simplified node-decoupled extended Kalman filter (EKF) learning algorithm. Section 4 introduces an error-driven active learning approach, and applies a hierarchical clustering technique for dividing the input space and a local learning approach for approximating the large error clusters. Section 5 details the experiments and results. Section 6 extends the algorithm to 3D space. Section 7 discusses the related work. Finally, we conclude this paper in Section 8.

2. The architecture of the learning system

Fig. 1 shows the flowchart of the learning system. The learning system has two parts: one is learning process and another is reconstruction process. The learning process builds the base layer and subnetworks of the higher level layers; while the reconstruction process combines these learned networks to predict the output for a given input. It should be noted that these two processes work simultaneously rather than separately. The learned subnetworks of previous layers are immediately used to do the construction during the learning process of the subnetworks of the current layer, as shown in Fig. 1a.

The whole learning system is based on a hierarchical structure of growing radial basis function networks (GRBF). The system firstly uses a GRBF to cover the whole work space. When the mapping error from this network does not change significantly, the residual error from the network drives active learning to explore detailed information in some areas with large errors, and local subnetworks are used to approximate these residual errors in each large error cluster separately. Similarly for the next higher level of subnetworks, i.e. in some areas with large mapping errors, the higher subnetworks approximate the residual error of the previous layers of network/subnetworks. Automatic growing radial basis function networks are utilized as the main components for different scales of mapping network or subnetworks. The GRBFs are trained by a simplified decoupled extended Kalman filter (SDEKF) algorithm [33], which will be discussed in Section 3.

We applied a hierarchical clustering technique to group and locate the large mapping error clusters for active learning. The hierarchical clustering technique groups the two most similar error clusters each time and forms a new error cluster, until one final single cluster remains. It keeps the records of clustering in a binary tree structure so we can cut off the tree at different levels to obtain different scales of clusters with different cluster sizes. The higher cutoff value we use, the bigger size each cluster is, and therefore the less similarity among the elements in each cluster.

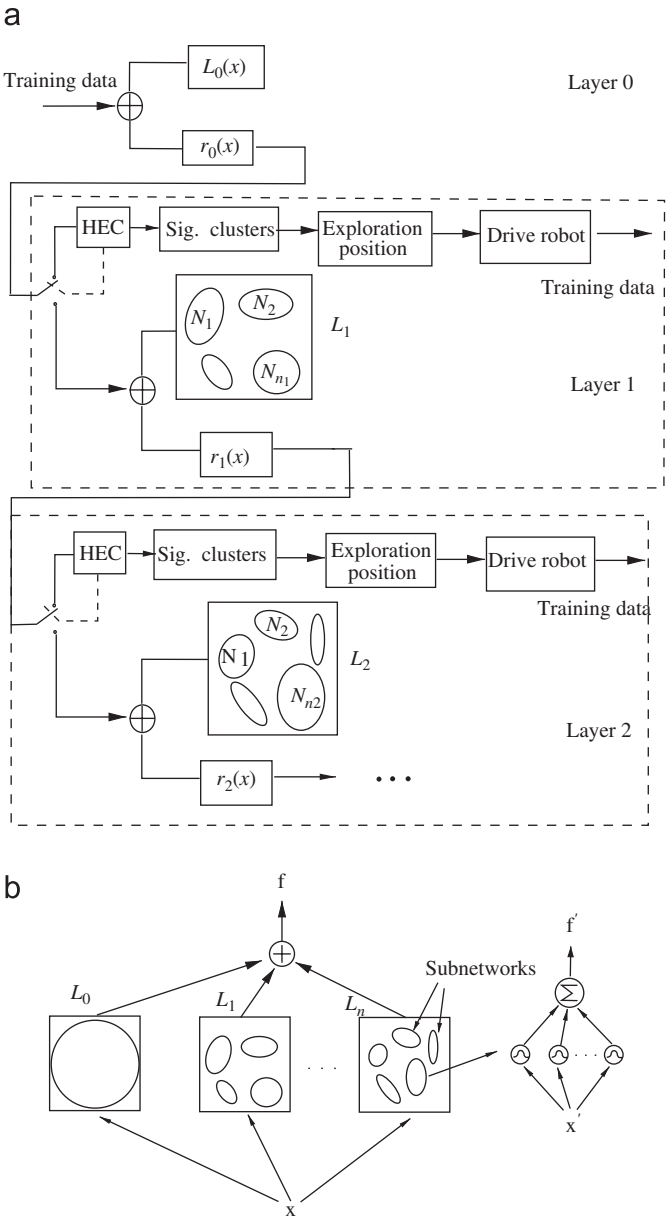

Fig. 1. Error-driven active learning system flowchart. Based on hierarchical error clustering, the system locates the areas with large learning errors in the current layer, and this information drives the robot system to explore these areas actively and separately, and build local mapping subnetworks to approximate the residual ($r(x)$) of the current layer at these significant areas. Each subnetwork is constructed by a growing radial basis function network. Different subnetworks have different number of nodes and different node locations and coverages. (a) Learning process. At the beginning of each layer, the residuals from previous layers are analyzed and grouped into clusters by using hierarchical error clustering technique (HEC). After the HEC analysis, the switch of the current layer in the figure is switched to the branch of constructing the subnetworks. Only the first three layers are shown in this figure. (b) Reconstruction process. The outputs from all the layers are summed up to generate the output. In each layer, for a given input, either one subnetwork or none is selected according to the coverages of the subnetworks of this layer.

During learning process, a module "learning phase" (not shown in the figure) monitors the change of the mapping errors, and determines when the error changes have effectively ceased and the map has become saturated. This module triggers the active learning process to build next level of subnetworks if the map becomes saturated. A long-term memory (LTM) stores training data of the most recent trials for later use in the hierarchical error cluster module for analyzing the mapping error distribution. The module "Sig. clusters" selects the large error clusters from the cluster tree generated by the module hierarchical error clustering (HEC). For each large error cluster, the system generates an active learning position at each learning step, sends this location to the robot and gets information back from the robot's action. The residual error from previous layers which cover this location is then used to train a local subnetwork. Such process continues until the mapping error changes from this cluster cease. The system then conducts the active learning in another area of large error clusters selected by the module HEC and the module "Sig. clusters". After finishing construction of all the subnetworks in the current layer, the system will continue to generate another level if the mapping error is still larger than a threshold.

3. A plastic RBF network and the simplified node-decoupled EKF learning algorithm

We need a nonlinear mapping algorithm to implement the base network (layer 0) and subnetworks of each layer in Fig. 1. Radial basis function networks not only have good global generalization abilities in function approximation [28,23], and have simple topological structure, but also have explicit structure that reveals how learning proceeds and allows us to interpret the learned network [52]. Pouget and Snyder [41] have shown that there is strong evidence that basis functions may exist in the human brain to support sensorimotor learning. In this section, we investigate a plastic radial basis function network to support error-driven active learning and present the SDEKF algorithm for learning and updating of RBF network parameters.

3.1. Radial basis function networks as a computational substrate for error-driven active learning

A RBF network is expressed as

$$\mathbf{f}(\mathbf{x}) = \mathbf{a}_0 + \sum_{k=1}^{N} \mathbf{a}_k \phi_k(\mathbf{x}), \tag{1}$$

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x}-\boldsymbol{\mu}_k\|^2\right), \tag{2}$$

where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{N_0}(\mathbf{x}))^{\mathrm{T}}$ is the vector of system outputs, N_0 is the number of outputs and $\mathbf{X}$ is the system input. $\mathbf{a}_k$ is the weight vector from the hidden unit $\phi_k(\mathbf{x})$ to the output, N is the number of radial basis function units, and $\boldsymbol{\mu}_k$ and σ_k are the kth hidden unit's center and width, respectively. The size of the receptive

field of each neuron varies and the overlaps between fields are different. Each neuron has its own center and coverage. The output is the linear combination of the hidden neurons.

3.2. Growing and pruning strategies of the RBF network

The growing RBF network starts with no hidden units, and with each learning step, i.e., after the system observes the consequence after an action, the network grows or shrinks when necessary or adjusts the network parameters accordingly.

The network growth criteria are based on the novelty of the observations [40,31], which are: whether the current network prediction error for the current learning observation is bigger than a threshold, and whether the node to be added is far enough from the existing nodes in the network, as shown in Eqs. (3) and (4). The criterion in Eq. (5) is to check the prediction error within a sliding window to ensure that growth is smooth.

$$\|\mathbf{e}(t)\| = \|\mathbf{y}(t) - \mathbf{f}(\mathbf{x}(t))\| > e_1, \tag{3}$$

$$\|\mathbf{x}(t) - \boldsymbol{\mu}_r(t)\| > e_3, \tag{4}$$

$$\sqrt{\sum_{j=t-(m-1)}^{t} \frac{\|\mathbf{e}(j)\|^2}{m}} > e_2, \tag{5}$$

where $(\mathbf{x}(t), \mathbf{y}(t))$ is the learning data at tth step, and $\boldsymbol{\mu}_r(t)$ is the center vector of the nearest node to the current input $\mathbf{x}(t)$. m is the length of the observation window.

If the above three conditions are met, then a new node is inserted into the network with the following parameters: $\mathbf{a}_{N+1} = \mathbf{e}(t)$, $\boldsymbol{\mu}_{N+1} = \mathbf{x}(t)$, $\sigma_{N+1} = k\|\mathbf{x}(t) - \boldsymbol{\mu}_r(t)\|$, where, k is the overlap factor between hidden units.

The above network growth strategy does not include any network pruning, which means the network size will become large, some of the hidden nodes may not contribute much to the outputs and the network may become overfit. In order to overcome this problem, we use a pruning strategy as in [31,30], over a period of learning steps, to remove those hidden units with insignificant contribution to the network outputs.

Let o_{nj} be the jth output component of the nth hidden neuron,

$$o_{nj} = a_{nj} \exp\left(-\frac{\|\mathbf{x}(t) - \boldsymbol{\mu}_n\|^2}{\sigma_n^2}\right), \quad r_{nj} = \frac{o_{nj}}{\max(o_{1j}, o_{2j}, \ldots, o_{Nj})}.$$

If $r_{nj} < \delta$ for M consecutive learning steps, then the nth node is removed. δ is a threshold.

3.3. GRBF network learning based on simplified node-decoupled EKF

Section 3.2 gives the network growing and pruning algorithm. At each step, however, if no new hidden node is added, a learning algorithm is needed to adjust the parameters of the existing nodes. There are two groups of parameters that need to be updated, one is the weights between the hidden units to the outputs, another is the centers and the widths of hidden units in the network. By adjusting the centers, the widths, and the weights, the network optimizes the output according to the current learning data.

The extended Kalman filter is widely used for updating RBF network parameters and has been proven to be able to achieve better performance than using gradient descent [31,20,49]. However, the traditional global extended Kalman filter approach updates all the above parameters of the RBF network at each learning step, and its computational complexity is $O(N_0(N_iN + N)^2)$ per learning step, where N_i is number of input components, i.e. the dimension of input vector X in Eq. (1). Also, the matrix inverse operation is involved in calculating the Kalman gain at each step. As the size of the network grows large, the computation cost gets very high.

In this section, we derive the node-decoupled EKF (ND-EKF) algorithm for training the RBF network, and further present its simplified formulae. We use the terms of standard EKF which can be found in [20]. In ND-EKF, during learning, instead of updating all the network parameters once at each learning step, we update the parameters of each node independently. The parameters of the network are grouped into $N_0 + N$ components. The first N_0 groups are the weights, $\mathbf{w}_k = [a_{0k}, a_{1k}, \ldots, a_{Nk}]^{\mathrm{T}}$, $k = 1, 2, \ldots, N_0$ (a_{ij} is the weight from ith hidden node to jth output); and the rest N groups are the parameters of hidden units' parameters: $\mathbf{w}_k = [\boldsymbol{\mu}_k^{\mathrm{T}}, \sigma_k]^{\mathrm{T}}, k = 1, 2, \ldots, N$. The superscript T stands for transpose of a matrix.

Hence for kth parameter group at tth learning step, ND-EKF is given by

$$\mathbf{w}_k(t) = \mathbf{w}_k(t-1) + \mathbf{K}_k(t)\mathbf{e}_k(t), \tag{6}$$

where

$$\mathbf{e}_k(t) = \begin{cases} y_k(t) - f_k(\mathbf{x}(t)), & k = 0, 1, 2, \ldots, N_0, \\ \mathbf{y}(t) - \mathbf{f}(\mathbf{x}(t)), & k = N_0 + 1, \ldots, N_0 + N \end{cases} \tag{7}$$

and $\mathbf{K}_k(t)$ is the Kalman gain, which is given by

$$\mathbf{K}_k(t) = \mathbf{P}_k(t-1)\mathbf{B}_k^{\mathrm{T}}(t)[\mathbf{R}_k(t) + \mathbf{B}_k(t)\mathbf{P}_k(t-1)\mathbf{B}_k^{\mathrm{T}}(t)]^{-1} \tag{8}$$

$\mathbf{P}_k(t)$ is the error covariance matrix, and is updated by

$$\mathbf{P}_k(t) = [\mathbf{I} - \mathbf{K}_k(t)\mathbf{B}_k(t)]\mathbf{P}_k(t-1) + q\mathbf{I}, \tag{9}$$

where $y_k(t)$ is the kth component of $\mathbf{y}(t)$ in training data $(\mathbf{x}(t), \mathbf{y}(t))$, $\mathbf{B}_k(t)$ is the submatrix of derivatives of network outputs with respect to the kth group's parameters at tth learning step. $\mathbf{R}_k(t)$ is the variance of the measurement noise, and is set to be diag(λ) (λ is a constant) in this paper. q is a scalar that determines the allowed random step in the direction of the gradient vector.

The computational complexity of the node-decoupled EKF is $O(N_0(N+1)^2 + N(N_i+1)^2)$, which is much lower than the global EKF: $O(N_0(N_iN + N)^2)$.

In the Kalman gain calculation, matrix inversion is involved. This is further simplified by applying the matrix inversion lemma. In the following, we present the simplified formulae of Kalman gain in (8) and error covariance matrix in (9) for network weight parameters and hidden unit parameters separately [34].

3.3.1. For the weight parameters

For the weight parameters, $\mathbf{w}_k = [a_{0k}, a_{1k}, \ldots, a_{Nk}]$ $(k = 1, 2, \ldots, N_0)$, as the weights for each output are completely independent of those of other outputs; hence, the system learns the weights for each individual output separately. Therefore we have

$$\mathbf{B}_k(t) = [1, \phi_1, \phi_2, \ldots, \phi_N] \tag{10}$$

and the Kalman gain is

$$\mathbf{K}_k(t) = \frac{\mathbf{P}_k(t-1)\mathbf{B}_k^{\mathrm{T}}(t)}{\eta_k + \lambda}, \tag{11}$$

where $\eta_k = \mathbf{B}_k(t)\mathbf{P}_k(t-1)\mathbf{B}_k^{\mathrm{T}}(t)$, it is a scalar.

The error covariance matrix becomes

$$\mathbf{P}_k(t) = \mathbf{P}_k(t-1) - \frac{\mathbf{P}_k(t-1)\mathbf{B}_k^{\mathrm{T}}(t)\mathbf{B}_k(t)\mathbf{P}_k(t-1)}{\eta_k + \lambda} + q\mathbf{I}. \tag{12}$$

3.3.2. For the hidden units

For the parameters of $\mathbf{w}_k = [\boldsymbol{\mu}_k^{\mathrm{T}}, \sigma_k]^{\mathrm{T}} = [\mu_{k_1}, \mu_{k_2}, \ldots, \mu_{k_{N_i}}, \sigma_k]^{\mathrm{T}}$, the gradient matrix $\mathbf{B}_k$ becomes

$$\mathbf{B}_k(t) = \boldsymbol{\Delta}_k(t)\boldsymbol{\Psi}_k(t), \tag{13}$$

where

$$\boldsymbol{\Delta}_k(t) = \begin{bmatrix} \dfrac{2a_{k_1}\phi_k}{\sigma_k^2} \\ \dfrac{2a_{k_2}\phi_k}{\sigma_k^2} \\ \vdots \\ \dfrac{2a_{k_{N_0}}\phi_k}{\sigma_k^2} \end{bmatrix}, \tag{14}$$

$$\boldsymbol{\Psi}_k(t) = \left[(x_1 - \mu_{k_1}), \ldots, (x_{Ni} - \mu_{k_{Ni}}), \frac{\|\mathbf{x} - \boldsymbol{\mu}_k\|^2}{\sigma_k}\right], \tag{15}$$

where $x_1, x_2, \ldots, x_{Ni}$ are the components of input signal $\mathbf{x}$.

The Kalman filter in (8) can be rewritten as

$$\begin{aligned}\mathbf{K}_k(t) &= \frac{\mathbf{P}_k(t-1)\boldsymbol{\Psi}_k^{\mathrm{T}}(t)\Delta_k^{\mathrm{T}}(t)}{\lambda}\left(\mathbf{I} - \frac{\alpha_k(t)\Delta_k(t)\Delta_k^{\mathrm{T}}(t)}{\lambda + \alpha_k(t)\beta_k(t)}\right) \\ &= \frac{\mathbf{P}_k(t-1)\boldsymbol{\Psi}_k^{\mathrm{T}}(t)\Delta_k^{\mathrm{T}}(t)}{\lambda + \alpha_k(t)\beta_k(t)}\end{aligned} \tag{16}$$

and therefore $\mathbf{P}_k(t)$ can be written as

$$\mathbf{P}_k(t) = \mathbf{P}_k(t-1) - \frac{\beta_k(t)\mathbf{P}_k(t-1)\boldsymbol{\Psi}_k^{\mathrm{T}}(t)\boldsymbol{\Psi}_k(t)\mathbf{P}_k(t-1)}{\lambda + \alpha_k(t)\beta_k(t)} + q\mathbf{I}, \tag{17}$$

where $\alpha_k(t) = \boldsymbol{\Psi}_k(t)\mathbf{P}_k(t-1)\boldsymbol{\Psi}_k^{\mathrm{T}}(t)$, and this is a scalar. $\mathbf{R}_k(t) = \mathrm{diag}(\lambda)$. $\beta_k(t) = \boldsymbol{\Delta}_k^{\mathrm{T}}(t)\boldsymbol{\Delta}_k(t)$.

So by using Eqs. (6), (11), (12), (16), and (17), the matrix inversion operations in ND-EKF are avoided, this forms the SDEKF algorithm which was used to train GRBF of layer 0 and the subnetworks of higher layers in our learning system shown in Fig. 1.

4. Error-driven active local learning

At the first stage of robot mapping learning such as arm kinematics learning and eye/hand coordination learning, the robot acts randomly. On each step, the network is trained using the data from the robot action. However, the training error in the workspace is not uniform, i.e. some areas have larger training errors than others. This is because the mapping is nonlinear, and the mapping difficulties are different over the workspace.

To reduce the large mapping error clusters, an error-driven active learning approach is presented. We see examples of active learning in human beings such as infants repeating a pattern of actions to gain skills, and athlete training specific actions to gain certain skills in order to overcome a bottleneck of performance. Error clustering and local learning are two key components in active learning. We applied hierarchical error clustering to locate the large error clusters at different levels and used local subnetworks to approximate the residual mapping errors of these clusters, as shown in Fig. 1.

4.1. Hierarchical error clustering

Based on error locations and their amplitudes, hierarchical error clustering groups the mapping errors simultaneously over a variety of scales by creating a cluster tree. At each step of the hierarchical clustering, only two mapping errors with the nearest distance are joined. The distance between two mapping errors is defined as

$$d_{rs} = \sqrt{\sum_{i=1}^{n}(x_{ri} - x_{si})^2 w_i}, \tag{18}$$

where r and s are two error indexes, n is the number of components of each error (location components (x, y) and error amplitude in our application). x_{ri} and x_{si} are error components. w_i is the weight for the ith component.

The hierarchical clustering algorithm has the following steps:

```
Algorithm: hierarchical clustering
Initialize: assign each mapping error to
            its own cluster and ID.
```

```
Repeat
        1. Compute distance between pairs of
        clusters according to formula (18).
        2. Merge the two clusters that are
        closest to each other and assign
        a new ID.
Until there is only one cluster left.
```

The cluster tree stores multilevel set of clusters, where two closest clusters at one level are joined as a cluster of the next higher level. The hierarchical structure allows us to choose what scale of clusters to use according to at which size the local error clusters are chosen and approximated by local subnetworks.

4.2. Local learning

Once the hierarchical error cluster tree is generated, the robot is driven to move to the areas with large average errors, and capture the data relevant to the learning task, for example, the arm endpoint coordinates and the arm joint values for the task of learning arm kinematics. As shown in Fig. 1, the obtained data are firstly compared with the predicted value from the lower mapping network(s), and then the residual mapping errors are used to train local subnetworks. Local subnetworks are also built by growing radial basis function networks trained by SDEKF. Although global growing radial basis function networks, which cover the whole work space, can adjust the number of neurons, and the location and size of the receptive field of each neuron, these adjustments are usually affected by the distribution of the training data, i.e. the network attempts to insert new neurons, or attracts more neurons/adjusts existing neurons to satisfy the area with density data. If a global network is used in active learning, the active learning data, which is usually from some local areas, will affect the network mapping in other areas. Therefore in this paper, we used a multilevel network structure to learn the rough mapping at the base layer and to approximate the large error clusters by local subnetworks at higher levels. For a training sample $(\mathbf{x}(t), \mathbf{y}(t))$ at time t, the residual error $\mathbf{r}_j$ for the input of the subnetwork at the jth layer is computed

$$\mathbf{r}_j(\mathbf{x}(t)) = \mathbf{y}(t) - \sum_{l=0}^{j-1} \sum_{i=1}^{N_l^m} \mathbf{a}_{li}^m \phi_{li}^m(\mathbf{x}(t)), \tag{19}$$

where $\mathbf{a}_{li}^m$ is the weights from the ith neuro to the output of the mth subnetwork which covers the current input point $\mathbf{x}(t)$ at the lth layer ($l = 0, 1, \ldots, j-1$), while $\phi_{li}^m(\mathbf{x}(t))$ is ith neuron of the mth subnetwork. N_l^m is the number of the neurons in the mth subnetwork at the lth layer. So given the training example $(\mathbf{x}(t), \mathbf{y}(t))$, the training pair for the mth subnetwork at jth layer is $(\mathbf{x}(t), \mathbf{r}_j(\mathbf{x}(t)))$.

5. Experimental results

To test the proposed error-driven active learning in growing radial basis function networks for early robot learning, the nonlinear robot arm inverse kinematic problem was used as a testbed, in particular, we used a two-link arm which consists of forearm and upper-arm. The system attempts to predict the joint angles (j_1, j_2) given an arm end position (p, θ) in polar coordinates, where j_1 is the angle between the upper-arm and the body baseline, j_2 is the angle between the upper-arm and the axis of the forearm, p is the effective length of the arm axis from shoulder (arm base point) to the arm end position and θ is the angle the axis makes at the shoulder. The following formulae models the arm:

$$p = \sqrt{l_1^2 + l_2^2 + 2l_1 l_2 \cos j_2} \quad \text{and} \quad \theta = j_1 + \arctan \frac{l_2 \sin j_2}{l_1 + l_2 \cos j_2},$$

where l_1 and l_2 are the lengths of the upper-arm and forearm, respectively. This is also referred as a shoulder encoding scheme. In the following experiments, $l_1 = 15$, and $l_2 = 17$ was used.

The system started to build the mapping by random movements in the whole work space and utilized a growing RBF network as a base layer to support the first stage learning. The following parameters were used for the based layer and subnetworks of higher layers: $e_1 = 0.05$, $e_2 = 0.005$, $e_3 = \max\{0.4 * 0.999^i, 0.07\}$ (i is the learning step); field overlap factor $k = 1.0$, size of sliding window in equation (5) $m = 150$, size of sliding window in pruning node $M = 100$, pruning threshold $\delta = 0.001$. During the learning process, neurons were added to or removed from the mapping network, and the position and width of each neuron in the network were adjusted to reflect the training data, therefore the mapping error was reduced. However, after a period of training, the mapping error changes effectively cease, i.e. produce no further significant error reduction, as shown in Fig. 2. There are two main reasons for this problem: (a) Although each neuron in RBF networks only covers a local area, its width and position are adjusted in the learning process, therefore the adjustments may affect the mapping network(s) already set up before this trial; and (b) The training is driven by random movements of the robot arm, and the robot arm inverse kinematic problem is highly nonlinear which is mainly caused by the geometry of the robot arm. The random training samples and the global mapping network cause the

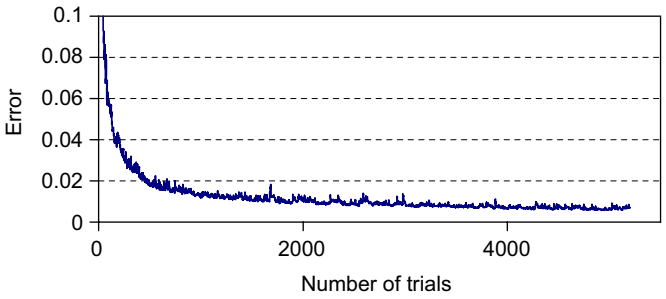

Fig. 2. Mapping error without active learning for the robot arm inverse kinematic problem. The mapping error does not change significantly after a certain number of training steps.

mapping error of the workspace to be nonuniform, as shown in Fig. 3 for the 2-link arm inverse kinematics problem.

To overcome the above problems, an error-driven active learning approach with local mapping subnetworks was used. Fig. 4 gives the dendrogram of the error clusters of the first layer after 2000 trials of training. It is a hierarchical tree structure showing the links of two nearest error clusters with relevant distance on y axis. From this figure, different scales of clustering can be obtained by cutting off clusters with small distance, which produce different size of error clusters. The selection of the cutoff point depends on the requirements of how big the cluster is or how big the difference is within a cluster. Generally, the higher the value of the cutoff point, the bigger the size of each cluster/subnetwork generated from the hierarchical tree. Fig. 5 shows the eight clusters with large average error of the first layer. A threshold was used to remove clusters with small number of error points, i.e. the system ignores the error clusters with the number of error locations less than this threshold. In this paper, we set this threshold to 10.

Fig. 6 shows the error distribution after active learning in each of the eight cluster areas in Fig. 5, 400 trials for each

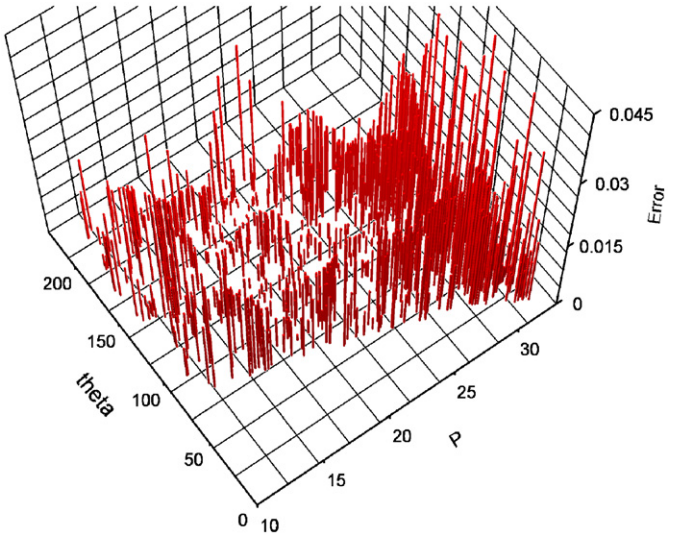

Fig. 3. Error distribution before active learning. The mapping error is large and nonuniform.

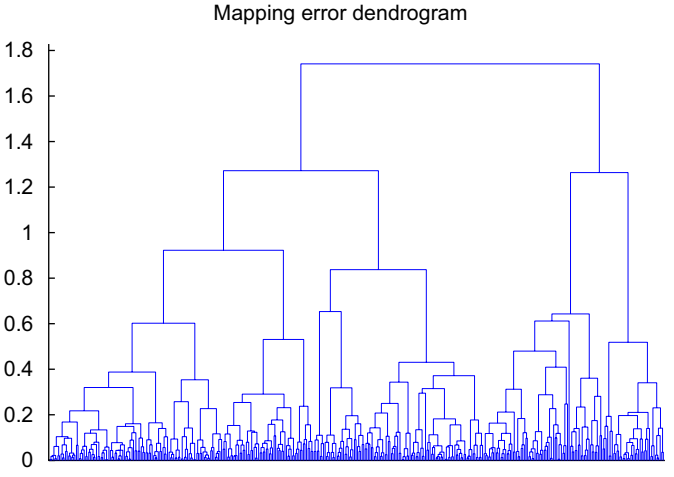

Fig. 4. Mapping error dendrogram. The horizontal axis is the error cluster index, and the vertical axis is the distance between the two error clusters it links. Large values on y axis indicate that the two linked clusters are quite different; while small values mean that the two linked clusters are similar.

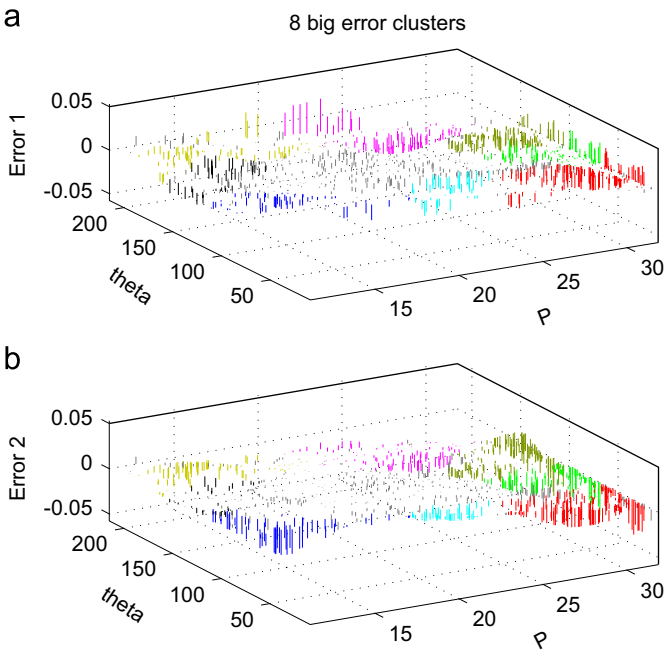

Fig. 5. Large error clusters located by the hierarchical clustering technique. The large error clusters are displayed in two figures of mapping errors, each for one output component. The small errors are plotted in gray, while the large error clusters are plotted in other colors.

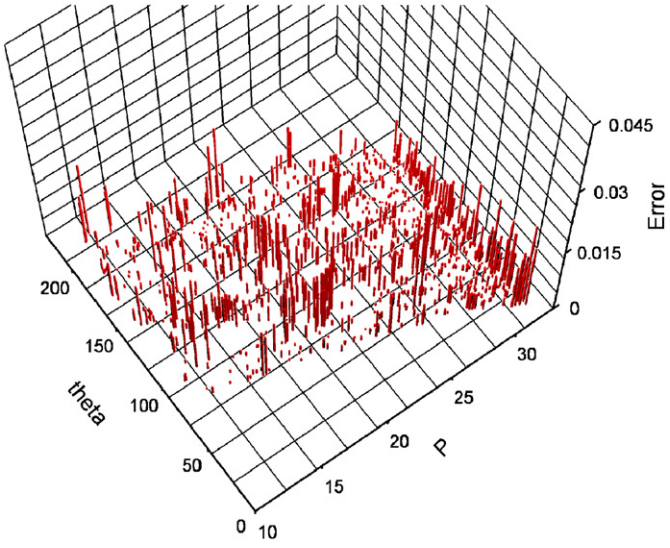

Fig. 6. Error distribution after active learning. The overall mapping errors are reduced greatly, and become uniform.

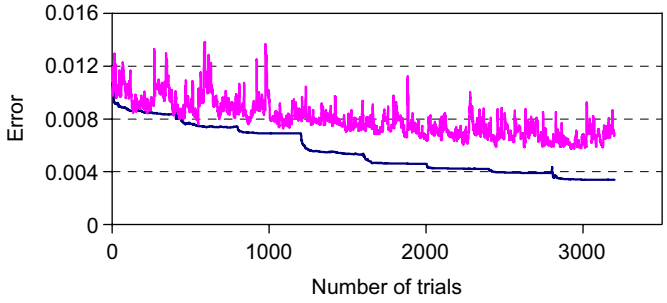

Fig. 7. Comparison between mapping errors with passive and active learning. The upper curve is the error with passive learning while the bottom one is for active learning. The figure only shows the results from the beginning of active learning and ignores the training of the first layer of the mapping network.

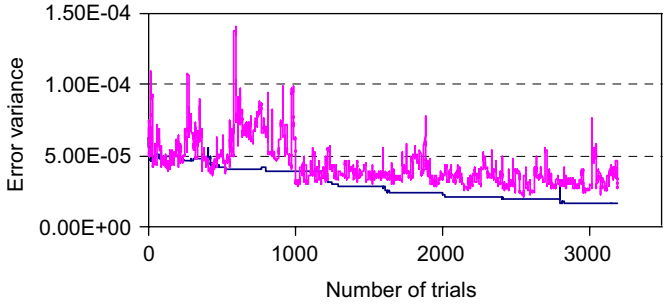

Fig. 8. Comparison between error variances with passive and active learning. The upper curve is passive learning while the bottom one is for active learning.

cluster. From Figs. 3 and 6, we see that the error after active learning becomes more uniform and smaller.

Fig. 7 shows a comparison of mapping errors against the number of trials with passive and active learning, and Fig. 8 gives their error variance comparison results. At each trial, the error and error variance in the figure are the average mapping error over the whole working space (1000 points across the workspace) and the variance of these mapping errors, respectively. From these two figures, we see that the system reduced the error and its variance significantly by using active learning and local RBF subnetworks at different levels. During active learning, the robot explored one large error cluster and set up the local subnetwork before turning to another. At the beginning of training each subnetwork for a cluster, the mapping error for the local subnetwork was large due to lack of experience and neurons in the subnetwork. This is demonstrated by the transitions between each local subnetwork in the active learning curves in Figs. 7 and 8. For the eight subnetworks, the following numbers of neurons were achieved automatically by growing RBF network algorithm: 10, 13, 14, 14, 13, 9, 7, and 12. Figs. 9 and 10 demonstrate the average error and error variance changes for each large mapping error cluster by using active learning with local subnetworks. The average error was reduced greatly for each cluster and the error variance was also improved. The error reductions in these large error

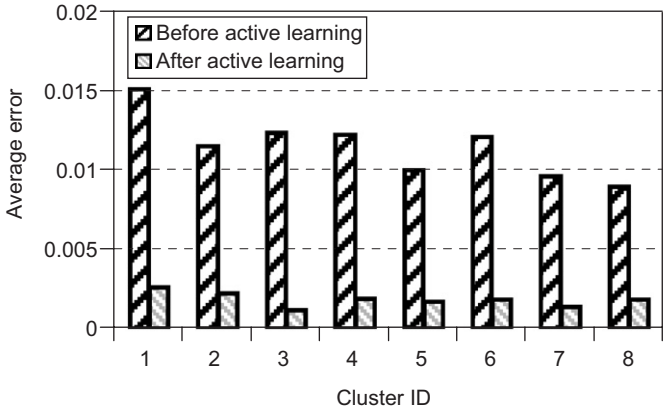

Fig. 9. Comparison between mapping errors before and after active learning for each large mapping error cluster.

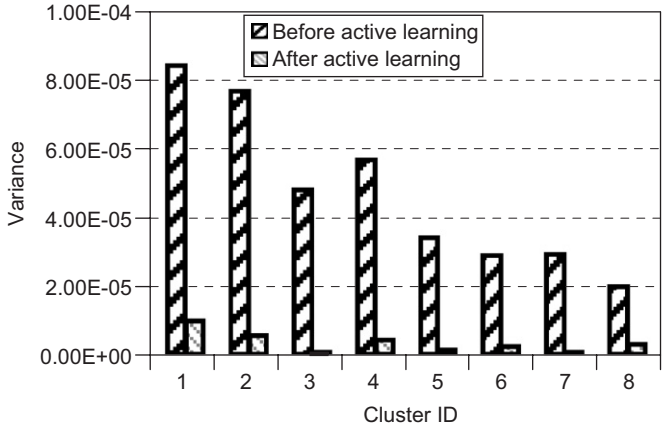

Fig. 10. Comparison between error variances before and after active learning for each large mapping error cluster.

clusters make the mapping error to be more uniform across the whole workspace as shown in Fig. 6 and the overall error to become smaller as shown in Fig. 7. It should be noted that in this paper, only two layers of network are used, but it is straightforward to add more layers of subnetworks to reduce any large error clusters using the same mechanism.

5.1. Under noise condition

In order to test the algorithm's robustness under noise, we applied zero-mean gaussian noise to the polar coordinates (p,θ) of the robot arm endpoint and the two joints. The noise is controlled by the standard deviation of the gaussian noise function, which is scaled by the noise level ψ, and sensory data range R of each joint, p or θ, respectively, as shown in (20).

$$N|_{j1,j2,p,\theta} = N(0, \psi R|_{j1,j2,p,\theta}). \tag{20}$$

It should be noted that for the gaussian noise, if we add, say, 10% noise level to a joint, then the probability of the noise between 10% and 20% of the related joint range is about 27%, and there is about 5% probability of the noise

data being larger than 20% of the joint range. Another issue is that we applied gaussian noise to the two joints and polar coordinates (p,θ) simultaneously, and this greatly increases the overall noise level compared to each individual modality noise level. The active learning method reduced the mapping errors and their variance under noise conditions. Fig. 11 shows a comparison result between mapping errors with passive learning and active learning under noise level $\psi = 2\%$.

Table 1 summarizes the comparison results of the average mapping error and the variance over the whole workspace after the training process, among passive learning with adaptive RBF networks, active learning with hierarchical adaptive RBF networks, and hierarchical mixtures of experts (HME) [25] under different noise levels: $\psi = 0\%$ and 2%. In the comparison, the system tried the same total number of learning steps for passive learning, active learning and HME training. We see that at each noise level, local active learning achieved smaller mapping errors than passive learning, and the variances were improved too. Two HME structures have been tested under each above noise level, 36 experts and 121 experts for noiseless condition, 100 experts and 365 experts under 2% noise level. All of these four HME structures used 2 levels, and expectation–maximization (EM) algorithm was used as the learning algorithm for the HME architecture. The experiments show that active learning with hierarchical RBF network structure in this paper performs better than HME in both noise levels.

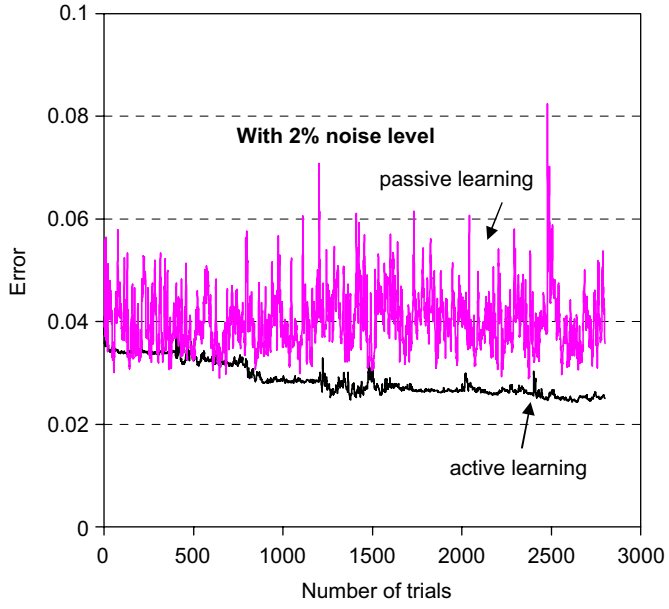

Fig. 11. Comparison between mapping errors with passive learning and active learning under noise. The error is the average across the whole workspace using 1000 test points.

From Table 1, we noticed that under noise condition, active learning did not reduce mapping errors so significantly as noiseless situation. This is because active learning is mainly to reduce the nonuniformity of sensorimotor mapping errors due to the nonlinear of the mapping, i.e. to actively explore the areas with large mapping errors. While the noise is evenly distributed over the whole work space, and these noise may contribute to the mapping error, therefore active learning with local subnetworks may not reduce the error so significantly under noise condition.

6. Extension to 3D space

In this section, the algorithm is applied to inverse kinematic problem of an industrial robot (PUMA 560) in 3D space. Fig. 12 and Table 2 show the robot and its Denavit–Hartenberg (D–H) parameters, respectively. The link frame assignment of PUMA 560 for D–H parameters and the definition of the parameters are the same as those in [10].

In this paper, we consider the PUMA robot inverse kinematic problem, i.e. converting the end effector position in 3D into joint values (the orientation of the end effector

Table 1
Comparison of average errors and variances among active learning, passive learning, and HME[a]

		Average error[b]	Variance[b]	Number of nodes[c]
No noise	Passive	0.0074	3.29E−05	30
	Active	0.0031	1.67E−05	29,10,13,14,14,13,9,7,12
	HME	0.0186	2.52E−04	36 experts
		0.0196	2.61E−04	121 experts
2% noise[d]	Passive	0.0345	6.12E−04	103
	Active	0.0250	4.74E−04	100,22,35,27,45,33,50,44
	HME	0.0403	6.58E−04	100 experts
		0.0412	6.61E−04	365 experts

[a]The system tried the same total number of learning steps for passive learning as for active learning approach. In active learning, the base layer tried 2000 learning steps to reach a stage of steady mapping error. Then local active learning starts, each local subnetwork covers one large error cluster and was trained 400 steps.

[b]The average error across the whole workspace and the variance of the mapping errors over the workspace after learning process. 1000 test points over the workspace were used.

[c]The number of nodes with active learning has the format: (the number of nodes in base layer, and number of nodes of each cluster in second layer).

[d]The number is the noise level in (20). We added gaussian noise to polar coordinates (p,θ) of the robot arm endpoint and the two joints.

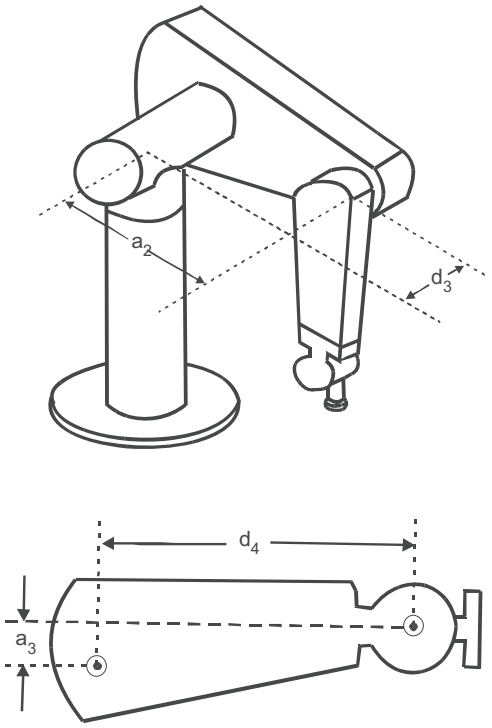

Fig. 12. PUMA 560 industrial robot.

Table 2
D–H Parameters of PUMA 560

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	0	0	θ_2
3	0	a_2	d_3	θ_3
4	-90°	a_3	d_4	θ_4
5	90°	0	0	θ_5
6	-90°	0	0	θ_6

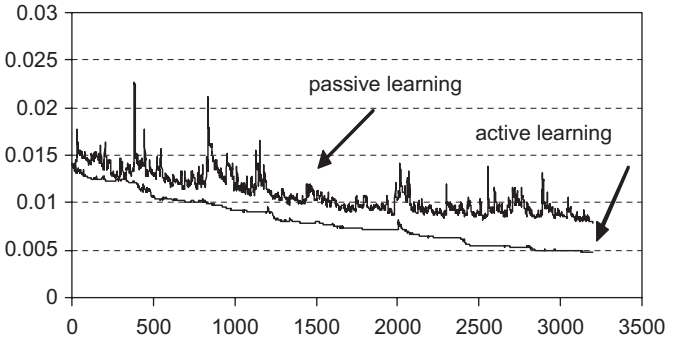

Fig. 13. Comparison between mapping errors with passive learning and active learning for PUMA robot.

are not considered). Spherical polar coordinates are used to represent a position in 3D space. In this study, the first three joints are limited to: $[-180, 0]$, $[-86, 86]$ and $[-180, 86]$. Fig. 13 illustrates the comparison between mapping errors with passive learning and active learning for PUMA 560 robot. The active learning with local networks overcomes the instability of the passive learning and reduces the overall mapping errors.

Table 3 gives the error comparison of each large error cluster before active learning and after active learning, together with the number of nodes for each cluster. The error of each cluster was reduced significantly and reached the error level of other areas in the workspace, and hence the non-uniform error distribution was improved.

7. Related work

7.1. Biological plausibility

Our long goal is to investigate biologically inspired learning algorithms for robots at their early stages in order to not only control artificial machines, but also verify and help us further understand some models and theories of biology and psychology [59,35]. In this paper, we focus on error-driven active learning based on an adaptive and self-growing neural network with hierarchical structure and local subnetworks. Recent neurophysiological, psychological and neuropsychological research provides strong evidence to support the key components in the approach present in this paper.

Evidence reveals that humans use basis functions to perform sensorimotor transformations [41]; the plasticity of the network in this paper in terms of growing/shrinking is similar to that reported in neuroscience [16]. An extended Kalman filter was used to update the system parameters during learning. Recent research findings provide evidence that Kalman filtering occurs in human visual information processing [43,44], motor coordination control [55], and spatial learning and localization in the hippocampus [7,51]. In hippocampus studies, a Kalman filtering framework has been mapped to the entorhinal-hippocampal loop in a biologically plausible way [7,51]. According to the mapping, region CA1 in the hippocampus holds the system reconstruction error signal, and the internal representation is maintained by Entorhinal Cortex (EC) V–VI. The output of CA1 corrects the internal representation, which in turn corrects the reconstruction of the input at EC layers II–III. We used matrix inversion lemma to simplify the EKF

Table 3
Error comparison of each cluster before active learning and after active learning

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
Before active learning	0.0274	0.0288	0.0335	0.0212	0.0141	0.0147	0.0170	0.0166
After active learning	0.0052	0.0047	0.0062	0.0045	0.0043	0.0044	0.0039	0.0056
Number of nodes	83	91	86	93	62	88	68	82

calculations, and this has been widely used in computational neuroscience [24]. O'Keefe also provided a biologically plausible mechanism by which matrix inversions might be performed by the CA1 layer through an iterated update scheme and in conjunction with the subiculum [39].

Subnetworks were used to learn sensorimotor mapping locally, similar modular decomposition of sensorimotor learning exists in human beings to learn visuomotor maps [17], and other kinematic and dynamic transformations [13]. These evidence suggest that the brain/CNS may use multiple local internal modules or experts for different context, and integrate these local modules by learning.

7.2. Hierarchical neural networks and active learning

The network grows and shrinks according to the novelty of each example the robot receives, and the algorithm is based on the RAN [40] and MRAN [30,31] algorithms. MRAN greatly improved its performance by introducing network pruning into RAN and utilizing EKF to adjust network parameters. In this paper, we simplify the EKF algorithm by decoupling the parameters into two parts: weights and hidden units, and therefore reduce the computational cost for real-time applications with reasonable good accuracy [34]. Similar decoupled EKF technique was also recently applied to FGAP-RBF algorithm for classification problems [60]. In our paper, the node-decoupled EKF for RBF networks is further simplified by applying the matrix inversion lemma, and new formulae are derived for the simplified node-decoupled EKF. Nishida et al. used similar localized extended Kalman filter to train hyper basis function networks [37]. In order to increase the learning speed in incremental construction of feedforward neural networks, Huang et al. [29,22,21] proposed a novel online sequential extreme learning algorithm which all the parameters of new added hidden nodes can be chosen randomly and the learning is only used to adjust the output weights linking the hidden layer and the output layer.

Regarding hierarchical neural network structure with local subnetworks, Jordan and Jacobs presented hierarchical mixtures of experts (HME) [25]. HME uses a tree-structure architecture for supervising learning, in which the gating networks sit at the nonterminals of the tree. The input space is divided into a nested set of regions and each region is represented by an expert. The main differences between HME and the approach presented in this paper are: HME uses a fixed tree branching factor for the whole space (for example, binary tree structure), the sub-regions in the whole workspace are determined by the tree branching factor, and are not actively selected based on the training error feedback. Another similar hierarchical RBF network was presented in [12]. However, there are two main differences: the first is that we use a local subnetwork for each large error cluster, while [12] used one network for each layer; the second difference is that we use an automatic growing RBF network for each subnetwork, while [12] used a fixed network structure for each layer: the number of neurons in each layer, and the position and the coverage of each neuron were all fixed and predefined.

In terms of active learning, most of the approaches are based on complicated statistical analysis [9,57,15], which are usually not applicable to on-line applications. In robotic learning, the majority of active learning research comes from the reinforcement learning community [45,27]. The approach of combining interval estimation [27] exploration heuristic with the ID-3 inductive learning algorithm [42] was investigated in [45]. The actions were selected by choosing the leaves on the ID-3 tree that had a high expected probability of success (reward) conditioned on the current perceived attributes. Some researchers have also developed a variety of active exploration heuristics that tradeoff exploration and exploitation during adaption [54,5]. The prediction error was used as curiosity rewards in reinforcement learning in [50]. In this paper, we use the largest predictor error clusters to drive active learning to build a hierarchical mapping structure for sensorimotor learning. Active learning was also integrated into self-organizing maps (SOM) for classification [19], the sample query criterion is based on Bayesian decision theory and aims at selecting the data items of maximum expected sample value, a quantity that is measured by the expected change in the clustering cost function. Another technique related to active learning is Boosting [3,48,46]. Boosting algorithms produce an accurate prediction rule by combining rough and moderately inaccurate rules [46]. These inaccurate prediction rules or weak rules are created by repeatedly calling the base learning algorithm, but each time with different subsets of the training examples which most often misclassified by preceding weak rules (i.e. generate largest prediction errors). The principle of placing the most weight on the samples with large prediction error is similar to our error-driven active learning, however, the learning structure and algorithm are different. Here we use a hierarchical and adaptive RBF structure, each subnetwork in this structure only approximates the residual errors. Furthermore, we use prediction error information to actively drive robots to explore and gain more data from areas with large errors, while basic Boosting algorithms repeatedly reuse sub-sets of the training examples.

The error-driven learning approach in this paper is closely related to the divide-and-conquer principle [6]. The divide-and-conquer methods divide the input domain into subdivisions in order to simplify the learning problem. However, traditional divide-and-conquer methods suffer from the fact that either the division has to be given in advance or are restricted to a predefined method such as iteratively splitting the input domain by octrees [38]. Regarding the network structure of active learning, we see that a local subnetwork for each subdivision of the input domain has been widely used [9,14,2]. Solving problems locally using local models is an application of the general divide-and-conquer principle, and has been

extensively investigated and accepted in numerical mathematics [11] and geometric modeling [38].

8. Conclusion

Self-growing and shrinking neuronal mapping networks exist in our human brains especially during early infant cognitive development. Error-driven active learning is used very often in our daily life. In this paper, we describe a biologically inspired error-driven active learning approach in hierarchical adaptive RBF networks for early robot learning. The biological evidence was revealed to support the main mechanisms in our approach. A hierarchical clustering technique is introduced to group mapping errors and generate large error clusters to drive the active learning. Local subnetworks are used for active learning to approximate the residual errors from previous mapping network levels. Plastic radial basis function networks construct the substrate of the learning system and a simplified node-decoupled EKF algorithm is presented to train these radial basis function networks. Experimental results demonstrate that our approach of error-driven active learning with growing RBF networks significantly reduces mapping errors and improves the nonuniformity of the mapping errors across the whole working space.

Acknowledgments

We are very grateful for the support of EPSRC through Grant GR/R69679/01.

References

[1] M. Asada, K. MacDorman, H. Ishiguro, Y. Kuniyoshi, Cognitive developmental robotics as a new paradigm for the design of humanoid robots, Robotics Autonomous Syst. 37 (2) (2001) 185–193.
[2] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning, Artif. Intell. Rev. 11 (1–5) (1997) 11–73.
[3] R. Avnimelech, N. Intrator, Boosted mixture of experts: an ensemble learning scheme, Neural Comput. 11 (2) (1999) 483–497.
[4] A. Batista, C. Buneo, L. Snyder, R. Andersen, Reach plans in eye-centered coordinates, Science 285 (1999) 257–260.
[5] R.A.C. Bianchi, C.H.C. Ribeiro, A.H.R. Costa, Heuristic selection of actions in multiagent reinforcement learning, in: 20th International Joint Conference on Artificial Intelligence (IJCAI-07), India, 2007, pp. 690–695.
[6] G. Bontempi, M. Birattari, From linearization to lazy learning: a survey of divide-and-conquer techniques for nonlinear control, Int. J. Comput. Cognit. 3 (1) (2005) 56–73.
[7] O. Bousquet, K. Balakrishnan, V. Honavar, Is the hippocampus a Kalman filter?, in: Pacific Symposium on Biocomputing, Hawaii, 1998, pp. 655–666.
[8] R.A. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, M. Williamson, The Cog project: building a humanoid robot, in: C. Nehaniv (Ed.), Computation for Metaphors, Analogy, and Agents, Lecture Notes in Artificial Intelligence, vol. 1562, Springer, New York, 1999, pp. 52–87.
[9] D.A. Cohn, Z. Ghahbramani, M.I. Jordan, Active learning with statistical models, J. Artif. Intell. Res. 4 (1996) 129–145.
[10] J.J. Craig, Introduction to Robotics Mechanics and Control, third ed., Prentice-Hall, Englewood Cliffs, NJ, 2005.
[11] J. Fan, I. Gijbels, Local Polynomial Modelling and its Applications, Chapman and Hall, London, 1996.
[12] S. Ferrari, M. Maggioni, N.A. Borghese, Multi-scale approximation with hierarchical radial basis functions networks, IEEE Trans. Neural Networks 15 (1) (2004) 178–188.
[13] J.R. Flanagan, E. Nakano, H. Imamizu, R. Osu, T. Yoshioka, M. Kawato, Composition and decomposition of internal models in motor learning under altered kinematic and dynamic environments, J. Neurosci. 19 (RC34) (1999) 1–5.
[14] D. Fox, V. Heinzeand, K. Möller, S. Thrun, G. Veenker, Learning by error-driven decomposition, in: O. Simula (Ed.), Proceedings of the International Conference on Artificial Neural Networks, ICANN-91, 1991.
[15] K. Fukumizu, Statistical active learning in multilayer perceptrons, IEEE Trans. Neural Networks 11 (1) (2000) 17–26.
[16] M.S. Gazzaniga, R.B. Ivry, G.R. Mangun, Cognitive Neuroscience: The Biology of the Mind, W. W. Norton & Company, 2002.
[17] Z. Ghahramani, D.M. Wolpert, Modular decomposition in visuo-motor learning, Nature 386 (1997) 392–395.
[18] M. Graziano, Is reaching eye-centered, body-centered, hand-centered, or a combination?, Rev. Neurosci. 12 (2) (2001) 175–186.
[19] M. Hasenjager, H. Ritter, K. Obermayer, Active learning in self-organizing maps, in: E. Oja, S. Kaski (Eds.), Kohonen Maps, Elsevier, Amsterdam, 1999, pp. 57–70.
[20] S. Haykin, Kalman Filtering and Neural Networks, Wiley-Interscience, New York, 2001.
[21] G.B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing, in press.
[22] G.B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.
[23] G.-B. Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation, IEEE Trans. Neural Networks 16 (1) (2005) 57–67.
[24] Q.J. Huys, R.S. Zemel, R. Natarajan, P. Dayan, Fast population coding, Neural Comput. 2006, in press.
[25] M.I. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Comput. (1994) 181–214.
[26] T. Kohonen, Self-organization and Associative Memory, third ed., Springer, Berlin, 1993.
[27] A.M. Leslie, P. Kaelbling, M. Littman, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1996) 237–285.
[28] Y. Li, N. Sundararajan, P. Saratchandran, Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems, IEE Proc.—Control Theory Appl. 147 (4) (2000) 476–484.
[29] N.Y. Liang, G.B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate on-line sequential learning algorithm for feedforward networks, IEEE Trans. Neural Networks 17 (6) (2006) 1411–1423.
[30] Y. Lu, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks, Neural Comput. 9 (2) (1997) 461–478.
[31] Y. Lu, N. Sundararajan, P. Saratchandran, Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm, IEEE Trans. Neural Networks 9 (2) (1998) 308–318.
[32] M. Lungarella, G. Metta, R. Pfeifer, G. Sandini, Developmental robotics: a survey, Connection Science 15 (4) (2003) 151–190.
[33] Q. Meng, M.H. Lee, Error-driven active learning in growing radial basis function networks for early robot learning, in: 2006 IEEE International Conference on Robotics and Automation (IEEE ICRA 2006), Orlando, FL, USA, 2006, pp. 2984–2990.
[34] Q. Meng, M.H. Lee, Biologically inspired automatic construction of cross-modal mapping in robotic eye/hand systems, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS, 2006), Beijing, China, 2006, pp. 4742–4749.

[35] G. Metta, G. Sandini, J. Konczak, A developmental approach to visually-guided reaching in artificial systems, Neural Networks 12 (10) (1999) 1413–1427.
[36] A. Moore, J. Schneider, J. Boyan, M.S. Lee, Q2: memory-based active learning for optimizing noisy continuous functions, in: J. Shavlik (Ed.), Proceedings of the 15th International Conference of Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 386–394.
[37] K. Nishida, K. Yamauchi, T. Omori, An on-line learning algorithm with dimension selection using minimal hyper basis function networks, in: 11th International Conference on Neural Information Processing (ICONIP 2004), Lecture Notes in Computer Science, vol. 3316, India, 2004, pp. 502–507.
[38] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level partition of unity implicits, ACM Trans. Graphics (SIGGRAPH 03 Proceedings) 22 (3) (2003) 463–470.
[39] J. O'Keefe, Neural Connections, Mental Computation, MIT Press, Cambridge, MA, 1989.
[40] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.
[41] A. Pouget, L. Snyder, Computational approaches to sensorimotor transformations, Nature Neurosci. (Suppl.) 3 (2000) 1192–1198.
[42] J. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[43] R. Rao, D. Ballard, Dynamic model of visual recognition predicts neural response properties in the visual cortex, Neural Comput. 9 (4) (1997) 721–763.
[44] R. Rao, D. Ballard, Predictive coding in the visual cortex, Nature Neurosci. 2 (1) (1999) 79–87.
[45] M. Salganicoff, L.H. Ungar, R. Bajcsy, Active learning for vision-based robot grasping, Mach. Learn. 23 (1996) 251–278.
[46] R.E. Schapire, The boosting approach to machine learning: an overview, in: D.D. Denison, M.H. Hansen, C.H.B. Mallick, B. Yu (Eds.), Nonlinear Estimation and Classification, Springer, Berlin, 2003.
[47] J. Schmidhuber, Curious model-building control systems, in: Proceedings of International Joint Conference on Neural Networks, vol. 2, Singapore, 1991, pp. 1458–1463.
[48] H. Schwenk, Y. Benggio, Boosting neural networks, Neural Comput. 12 (8) (2000) 1869–1887.
[49] D. Simon, Training radial basis neural networks with the extended Kalman filter, Neurocomputing 48 (2002) 455–475.
[50] A. Stout, G. Konidaris, A. Barto, Intrinsically motivated reinforcement learning: a promising framework for developmental robot learning, in: The AAAI Spring Symposium on Developmental Robotics, 2005, ⟨http://www-anw.cs.umass.edu/pubs/2005/stout_kb_AAAIssdr05.pdf⟩.
[51] G. Szirtes, B. Póczos, A. Lőrincz, Neural Kalman filter, Neurocomputing 65–66 (2005) 349–355.
[52] K.M. Tao, A closer look at the radial basis function (RBF) networks, in: A. Singh (Ed.), Conference Record of The 27th Asilomar Conference on Signals, Systems and Computers, vol. 1, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 401–405.
[53] A. Terrazas, B.L. McNaughton, Brain growth and the cognitive map, Proc. Natl. Acad. Sci. 97 (9) (2000) 4414–4416.
[54] S. Thrun, K. Moller, Active exploration in dynamic environments, in: Neural Information Processing, vol. 4, Morgan-Kauffman, Lost Altos, CA, 1992, pp. 531–538.
[55] E. Todorov, M.I. Jordan, Optimal feedback control as a theory of motor coordination, Nature Neurosci. 5 (11) (2002) 1226–1235.
[56] S. Vijayakumar, Improving generalization ability through active learning, IEICE Trans. Inf. Syst. IEICE-Japan E82-D (2) (1999) 480–487.
[57] S. Vijayakumar, H. Ogawa, Improving generalization ability through active learning, IEICE Trans. Inform. Systems IEICE-Japan E82-D (2) (1999) 480–487.
[58] J.A. Walter, K. Schulten, Implementation of self-organizing neural networks for visuo-motor control of an industrial robot, IEEE Trans. Neural Networks 4 (1) (1993) 86–95.
[59] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, E. Thelen, Autonomous mental development by robots and animals, Science 291 (5504) (2001) 599–600.
[60] R. Zhang, G.B. Huang, N. Sundararajan, P. Saratchandran, Improved GAP-RBF network for classification problems, Neurocomputing, in press.

Qinggang Meng received the B.S. and M.S. degrees in electronic engineering from Tianjin University, China, and the Ph.D. degree in computer science from the University of Wales, Aberystwyth (UWA), UK. From 2002 to 2006 he was a postdoctoral researcher in the Department of Computer Science, UWA.

He is currently a lecturer in the Department of Computer Science, Loughborough University, UK. His main research interests include: biologically and psychologically inspired learning and control algorithms, learning from imitation, machine/human interaction, computer vision, intelligent and service robotics.

Mark Lee is Professor of Intelligent Systems at the University of Wales in the UK. His main research interests cover intelligent robotics and model-based systems. Current research is focussed on developmental robotics and is inspired by early infant psychology. An algorithmic approach is being used to explore fast, incremental learning in sensory-motor systems, drawing on data from psychology and infant development. The results have implications for applications involving autonomous systems in unstructured environments.