Front. Comput. Sci.
DOI 10.1007/s11704-016-5171-9

RESEARCH ARTICLE

Advancing the incremental fusion of robotic sensory features using online multi-kernel extreme learning machine

Lele CAO (✉)[1,2], Fuchun SUN[1], Hongbo LI[1], Wenbing HUANG[1]

1 Department of Computer Science and Technology, Tsinghua University, State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China
2 Department of Computing and Information Systems, The University of Melbourne, Parkville 3010 VIC, Australia

Abstract Robot recognition tasks usually require multiple homogeneous or heterogeneous sensors which intrinsically generate sequential, redundant, and storage demanding data with various noise pollution. Thus, online machine learning algorithms performing efficient sensory feature fusion have become a hot topic in robot recognition domain. This paper proposes an online multi-kernel extreme learning machine (OM-ELM) which assembles multiple ELM classifiers and optimizes the kernel weights with a p-norm formulation of multi-kernel learning (MKL) problem. It can be applied in feature fusion applications that require incremental learning over multiple sequential sensory readings. The performance of OM-ELM is tested towards four different robot recognition tasks. By comparing to several state-of-the-art online models for multi-kernel learning, we claim that our method achieves a superior or equivalent training accuracy and generalization ability with less training time. Practical suggestions are also given to aid effective online fusion of robot sensory features.

Keywords multi-kernel learning, online learning, extreme learning machine, feature fusion, robot recognition

1 Introduction

From the days when robotic arms were applied to industrial automation, to today, when space rovers, surgical robots, and personal service robots join the advanced dexterous robotics league, the research of robotics is booming in recent years. The advances of machine learning techniques and low-cost sensors are just a few of the many reasons propelling that boom. Robots usually accomplish their assigned tasks by reasonable interaction with target objects in its surroundings; an efficient learning algorithm that helps recognizing these objects and surroundings (a.k.a. robot recognition) based on sensory data could bring robots one step closer to completing different tasks correctly. Our major contribution is to propose an online multi-kernel extreme learning machine (OM-ELM) approach for incremental fusion of robotic sensory features. The following paragraphs briefly motivate the introduction of OM-ELM algorithm in the domain of robot recognition:

1) multi-kernel feature fusion A single sensor generally perceives merely limited and partial information about the target object and environment, so multiple similar and/or dissimilar sensors are commonly required to provide overcomplete and heterogeneous information to aid the robot recognition tasks. The data from different sensors is further combined with feature fusion methods to obtain synthetic observations. It is known that sensory output often comes with various levels of noise; and feature fusion can help to reduce the negative impact of noise from any standalone sensor, bringing about more accurate and robust recognition results. The research of multi-kernel learning (MKL) [1–4] has greatly inspired the study of feature fusion by solving a joint optimization problem to learn the optimal hyper kernel weights. MKL is an appropriate and convenient framework for feature fusion of robotic sensors, because any feature descriptor for

Received April 30, 2015; accepted November 12, 2015

E-mail: caoll12@mails.tsinghua.edu.cn

any sensory data can be easily represented by multiple kernels calculated with different kernel parameters and distance measures.

2) ELM-based online learning Batch learning methods assume i.i.d. sampling of training instances, which is in conflict with the nature of robotic sensors. The robotic sensory output is intrinsically sequential, highly redundant, and storage demanding, which impedes the straightforward application of batch learning. It requires online learning (i.e., incremental learning) [5, 6] algorithms, which are primed on a small dataset and keep updating its current model as more sensory data arrives. Furthermore, it is often desirable for the robot to make progressive guesses (even if rather imperfect ones) of targets as early as possible. In the extreme case, we expect the model to evolve upon the arrival of each sample transmitted from a certain sensor; however, the sampling frequencies of most robotic sensors are relatively high (at least 15Hz). Therefore, the model update time is preferably small and nearly linear in the number of samples. Among many state-of-the-art learning algorithms, extreme learning machine (ELM) [7,8] is selected as core classifiers, since it is known to be fast, have low demand on memory, and requires little user intervention, which is attributed to its single-hidden -layer structure requiring no iterative process.

The rest of this paper is organized as follows. In Section 2, we review the recent work of online multi-feature fusion. Section 3 presents our proposed approach OM-ELM (online multi-kernel extreme learning machine). A comprehensive evaluation on four real datasets collected from several robots is conducted in Section 4. Finally, conclusions are drawn in Section 5.

2 Related work

Feature learning and selection, as a fundamental problem of constructing suitable feature descriptors from the original high-dimensional sensory data, is the basis for robotic feature fusion. In many computer vision applications, some widely used feature descriptors, such as SIFT (scale invariant feature transform) [9], HoG (histogram of oriented gradients) [10], and local hue histograms [11], have become predominant; but Wang et al. [12] empirically summarized that there is no universally best hand-engineered feature for all datasets, and suggested that learning features directly from the dataset itself may be more advantageous. So several state-of-the-art approaches (e.g., Refs. [13–15]) of obtaining features directly from raw sensory data have been invented recently. However, this work emphasizes the study of methods that automatically or semi-automatically diversify off-the-shelf feature descriptors and convert them into a joint feature representation to support effective robot recognition, which is formally termed "feature fusion" in Refs. [16, 17].

Feature fusion, together with "data fusion" and "decision fusion", constitutes the complete concept of "information fusion" [16, 17]. A constantly seen schema of decision fusion is a two-layered structure where a classifier is trained on each feature descriptor and their outputs are aggregated by another classifier [18]. The output of each individual classifier is treated as a sort of abstracted feature space in this case. If a linear classifier is used in the second layer, it is equivalent to a linear combination of multiple base classifiers. But this kind of approach does not have guaranteed global optimality due to that the optimization problem is solved in two separate steps. On the other hand, fusion on raw data level (i.e., data fusion) is time-demanding and prone to noises. Therefore, we concentrate on feature-level fusion and try to investigate its effectiveness in robot recognition tasks. Recent years have witnessed an increasing interest in developing feature fusion approaches (e.g., [19, 20]); but most of them belong to the batch learning methodology that are incompatible with real -time recognition tasks and hard to scale-up for big data [21].

As such, online learning (incremental learning) [5, 6] has began to be used in robotic researches for dynamics modeling (e.g., Refs. [22, 23]), behavior learning (e.g., Refs. [24, 25]), and object/ environment recognition (e.g., Refs. [26–28]). The heart of online learning is an efficient model-updating mechanism applied after each new sample is received. Online learning embodies both training and testing in each iteration: the algorithm is presented with a new input vector at each incremental step; uses the current model to predict the corresponding label; and updates the model based on the correct label received. The primary objective is to minimize every prediction mistake, regardless of the data generation mechanism. Therefore, online learning algorithms do not have any information or assumption of sampling distribution or input sampling range, resulting in a more difficult and general algorithm than the ones trained in batch.

MKL and online learning have been merged with ELM during the past decade, producing different versions of multi -kernel ELM (e.g., Refs. [29–31]) and online ELM (e.g., Refs. [32–34]). But for multi-kernel ELM algorithms, there has been no proved theoretical guarantees on the maximum number of iterations and convergence rate; and the training is often ceased long before attaining the global optimal solution [35]. The online ELM algorithms are essentially built upon

the sequential implementation of ELM's least-squares solution $\beta = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{Y}$ (cf. Section 3 for notation definitions). But this solution is intrinsically difficult to represent the kernel matrix $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ in itself, which could be the reason that there has been no multi-kernel online ELM algorithm proposed as far as we know. Nevertheless, recent studies presented several generic versions of online multi-kernel learning algorithms: OM-2 (online multi-class multi-kernel) [21], OMCL (online multiple cue learning) [36], and OMKC (online multiple kernel classification) [37]. Among them, OMKC and OMCL learn a classifier for each kernel and combine classifiers by linear weights, while OM-2 advances in a way of allowing tuning the level of sparsity for kernel combinations. OM-2 also has a guaranteed mistake bound on any individual data sequence, and achieves a performance close to MKL batch algorithms [21]. But these methods may need more time for processing every succeeding training sample, which could pose problems if the data arrives in a high frequency.

In robot recognition field, there has been little studies that simultaneously apply online machine learning and multiple feature fusion. Luo et al. [26] collected a multimodal dataset from two mobile robots to perform an indoor environment identification task; then they experimented incremental SVM (support vector machine) [26], OMCL [36], and OM-2 [21] algorithms respectively on that dataset. Another representative work carried out by Araki et al. [28] tried to make a robot learn object concepts continuously from everyday operations in conjunction with a small amount of linguistic manual annotations. Their robot acquires multimodal data (i.e., visual, auditory, and haptic readings), which are in turn represented with multimodal latent Dirichlet allocation using Gibbs sampling in an online manner.

Different from previous work [21, 32, 34, 36, 37], our OM-ELM approach exploits robotic feature fusion and recognition in a joint framework that assembles multiple ELM classifiers and incrementally optimizes the kernel weights with a p-norm formulation of MKL problem. Extensive experiments show that our method achieves a superior or equivalent performance with less training time comparing to several state-of-the-art models. Practical suggestions are also given to aid effective implementation of such online robot recognition applications using multiple sensory features.

3 The proposed approach

We first formally define some notations used throughout this paper. We use bold uppercase characters to denote matrices, bold lowercase characters to denote vectors. Let $\aleph$ represent the sequential sensory samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)|\boldsymbol{x}_i \in \mathbb{R}^d, i = 1, 2, \ldots, N\}$, where $\boldsymbol{y}_i \in \{0, 1\}^M$ indicating M possible classes; and let $\mathbf{Y}$ be the label matrix $[\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N]^{\mathrm{T}}_{N\times M}$.

The **score function** $\boldsymbol{f}(\boldsymbol{x})$ generates an M-element vector $[f(\boldsymbol{x}, 1), f(\boldsymbol{x}, 2), \ldots, f(\boldsymbol{x}, M)]$ indicating the probability that sample $\boldsymbol{x}$ belongs to each class. We employ the linear[1)] score function $f(\boldsymbol{x}, m) = \mathbf{W} \cdot \boldsymbol{\hbar}(\boldsymbol{x}, m)_{,m=\{1,2,\ldots,M\}}$, where $\mathbf{W}$ are the linear weights and function $\boldsymbol{\hbar}(\boldsymbol{x}, m)$ maps the feature space into a kernel space (of L dimensions) whose inner product $\mathbf{K}((\boldsymbol{x}, \boldsymbol{y}'), (\boldsymbol{x}', \boldsymbol{y}'))$ is defined as the associated **kernel**. There exist many widely adopted kernels options such as $exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$ and so on.

With F sensory cues, we get the feature maps $[\boldsymbol{\hbar}_1(\cdot), \boldsymbol{\hbar}_2(\cdot), \ldots, \boldsymbol{\hbar}_F(\cdot)]$ and kernels $[\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_F]$, which are defined as

$$\begin{aligned}
&\boldsymbol{\hbar}(\boldsymbol{x}, m) = (\boldsymbol{\hbar}_1(\boldsymbol{x}, m), \boldsymbol{\hbar}_2(\boldsymbol{x}, m), \ldots, \boldsymbol{\hbar}_F(\boldsymbol{x}, m)),\\
&\boldsymbol{\hbar}_j(\boldsymbol{x}, m) = [\mathbf{0}, \ldots, \mathbf{0}, \underbrace{\boldsymbol{h}_j(\boldsymbol{x})}_{m\text{th block}}, \mathbf{0}, \ldots, \mathbf{0}]^{\mathrm{T}}_{M\times L},\\
&\mathbf{K}_j((\boldsymbol{x}, \boldsymbol{y}), (\boldsymbol{x}', \boldsymbol{y}')) = \boldsymbol{\hbar}_j(\boldsymbol{x}, y) \cdot \boldsymbol{\hbar}_j(\boldsymbol{x}', y'),\\
&j = 1, 2, \ldots, F, \quad y^{(')} = \arg\max_{y^{(')}\in\{1,2,\ldots,M\}} \boldsymbol{y}^{(')},
\end{aligned} \tag{1}$$

where $\boldsymbol{h}_j(\boldsymbol{x})$ is an ELM feature map that is label-independent. The **weight matrix** $\mathbf{W}$ consists of M blocks:

$$\begin{aligned}
&\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M]_{L\times M},\\
&\mathbf{w}_m = \sum_{j=1}^{F} w_{m,j}\boldsymbol{\beta}_{m,j} = \sum_{j=1}^{F} w_{m,j}[\beta_1, \beta_2, \ldots, \beta_L]_{m,j},\\
&\mathbf{w}_{\cdot j} = [w_{1,j}, w_{2,j}, \ldots, w_{M,j}]^{\mathrm{T}}_{j=1,2,\ldots,F},\\
&\mathbf{B}_j = [\boldsymbol{\beta}_{1,j}, \boldsymbol{\beta}_{2,j}, \ldots, \boldsymbol{\beta}_{M,j}]_{M\times L},
\end{aligned} \tag{2}$$

where $\boldsymbol{\beta}_{m,j}$ is the weight component for the jth cue and mth class, which can be efficiently obtained by an ELM-like ridge regression (RR) solver introduced in Section 3.1; notation $w_{m,j}$ is a scalar factor of $\boldsymbol{\beta}_{m,j}$; and we can formally define $\mathbf{W} \cdot \boldsymbol{\hbar}_j(\boldsymbol{x}, m) = \mathbf{w}_m \cdot \boldsymbol{h}_j(\boldsymbol{x})$ by its construction. The **generic norm** of $\mathbf{w}_{\cdot j}$ is indicated by $\|\mathbf{w}_{\cdot j}\|$. We also denote $\|\mathbf{w}_{\cdot j}\|_p$ as ***p*-norm** of $\mathbf{w}_{\cdot j}$, where $\|\mathbf{w}_{\cdot j}\|_p = \left(\sum_{m=1}^{M} |w_{m,j}|^p\right)^{1/p}$. In the rest part of this paper, p and q represent dual coefficients of **dual norm** $\|\cdot\|_p$ and $\|\cdot\|_q$, indicating $p^{-1} + q^{-1} = 1$. We use $\mathbf{W}'$ to denote the matrix composed by the concatenation of the F **scalar vectors** $\mathbf{w}_{\cdot j}$, therefore $(2, p)$-**group norm** is defined below as in [21, 41, 42]:

$$\begin{aligned}
\|\mathbf{W}'\|_{2,p} &= \|(\|\mathbf{w}_{\cdot 1}\|_2, \|\mathbf{w}_{\cdot 2}\|_2, \ldots, \|\mathbf{w}_{\cdot F}\|_2)\|_p\\
\mathbf{W}' &= (\mathbf{w}_{\cdot j})_1^F = (\mathbf{w}_{\cdot 1}, \mathbf{w}_{\cdot 2}, \ldots, \mathbf{w}_{\cdot F}).
\end{aligned} \tag{3}$$

[1)] To simplify the optimization constrains, bias is not considered, which is also consistent with ELM theories [7, 38–40]

When optimizing the MKL problem (Section 3.2), we consider the **multi-class loss function** adapted from [43]:

$$\begin{aligned}\mathscr{L}(\mathbf{W}', \mathbf{B}, \boldsymbol{x}, \boldsymbol{y}) &= \max_{m \neq y} |1 - \mathbf{W}' \cdot \mathbf{B} \cdot [\hbar(\boldsymbol{x}, y) - \hbar(\boldsymbol{x}, m)]|_+ \\ y &= \arg \max_{y \in \{1,2,\ldots,M\}} \boldsymbol{y}, \quad \boldsymbol{y} \in \{0,1\}^M,\end{aligned} \tag{4}$$

where $|\cdot|_+ = max\{\cdot, 0\}$ that is convex and upper bounds the misclassification loss; and $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_F)$.

3.1 Multi-class extreme learning with kernels

The multi-class ELM has the following decision function [39] for predicting the label of sample $\boldsymbol{x}$ using the jth cue:

$$label_j(\boldsymbol{x}) = \arg \max_{m \in \{1,\ldots,M\}} [\boldsymbol{\beta}_{m,j} \cdot \boldsymbol{h}_j(\boldsymbol{x})], \ \forall_{j \in \{1,2,\ldots,F\}}, \tag{5}$$

where $\boldsymbol{\beta}_{m,j} \cdot \boldsymbol{h}_j(\boldsymbol{x})$ is the score function when training sample $\boldsymbol{x}$ is assigned to class m; similarly to previous definitions, $\boldsymbol{\beta}_{m,j} = [\beta_1, \beta_2, \ldots, \beta_L]_{m,j}$ is the row vector of the weights between L hidden neurons and the mth output node; and $\boldsymbol{h}_j(\boldsymbol{x}) = [G_j(\mathbf{a}_1, b_1, \boldsymbol{x}), \ldots, G_j(\mathbf{a}_L, b_L, \boldsymbol{x})]^{\mathrm{T}}$ is the map of the hidden layer with respect to sample $\boldsymbol{x}$. In practice, activation functions $G(\mathbf{a}_k, b_k, \boldsymbol{x})_{k=1,2,\ldots,L}$ are irrelevant to class label m, creating a so called ELM feature space $\mathbf{H}_j$ for the jth cue, which is only related to the training samples and the value of L (the number of hidden neurons) [39]:

$$\mathbf{H}_j = \begin{bmatrix} \boldsymbol{h}_j^{\mathrm{T}}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}_j^{\mathrm{T}}(\boldsymbol{x}_N) \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} G_j(\mathbf{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G_j(\mathbf{a}_1, b_1, \boldsymbol{x}_N) \\ \vdots & \ldots & \vdots \\ G_j(\mathbf{a}_L, b_L, \boldsymbol{x}_1) & \cdots & G_j(\mathbf{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}. \tag{6}$$

In other words, $\mathbf{H}_j$ maps the feature dimension from d to L. Due to the fact that all parameters for hidden neurons (i.e., $\{\mathbf{a}_i, b_i\}_{i=1\cdots L}$) can be randomly produced as long as it obeys a certain continuous probability distribution, the hidden neuron parameters could already be decided before the training data arrives. For instance, Sigmoidal and Gaussian functions are among the many commonly used functions. According to [7, 39], the output weight $\mathbf{B}_j$ can be calculated by equation

$$\mathbf{B}_j = [\boldsymbol{\beta}_{1,j}, \boldsymbol{\beta}_{2,j}, \ldots, \boldsymbol{\beta}_{M,j}]_{M \times L} = \mathbf{H}_j^{\mathrm{T}} (\frac{\mathbf{I}}{C} + \mathbf{H}_j \mathbf{H}_j^{\mathrm{T}})^{-1} \mathbf{Y}, \tag{7}$$

where $\mathbf{I}/C$ is a set of small positive values added to the diagonal of $\mathbf{H}_j \mathbf{H}_j^{\mathrm{T}}$ to make the resulting solution more stable and have better generalization performance; the parameter C is a tradeoff between the distance of the separating margin and the training error [39, 44]. In most real applications, the feature map $\boldsymbol{h}_j(\boldsymbol{x})$ is unknown; one then may apply Mercer's conditions; and define a kernel for ELM as Eq. (1)

$$\mathbf{K}_j(\boldsymbol{x}, \boldsymbol{x}') = \mathbf{H}_j \mathbf{H}_j^{\mathrm{T}} = \boldsymbol{h}_j(\boldsymbol{x}) \cdot \boldsymbol{h}_j(\boldsymbol{x}'), \tag{8}$$

where the number of hidden neurons need not be given either. Kernel $\mathbf{K}_j$ is neither relevant to the number of output nodes m nor to the training label y_m's. [39] Substituting Eq. (8) into Eq. (7), we thus obtain the kernel-based solution:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \mathbf{K}_j(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ \mathbf{K}_j(\boldsymbol{x}, \boldsymbol{x}_N) \end{bmatrix}^{\mathrm{T}} (\tfrac{\mathbf{I}}{C} + \mathbf{K}_j)^{-1} \mathbf{Y}. \tag{9}$$

3.2 Problem of multi-kernel learning

We consider F kernels $[\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_F]$ representing F feature cues/modalities respectively, and simplify the MKL problem by decomposing it into two separate tasks, i.e., learning 1) a classifier for each individual kernel, and 2) scalar weights $\mathbf{w}_{\cdot j}$ that combine the outputs of individual kernel classifier to form the final prediction. This simplification allows us to have a multi-kernel multi-class ELM classifier based on Eq. (9).

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{x}) &= \sum_{j=1}^{F} \mathbf{w}_{\cdot j} \left(\mathbf{B}_j^{\mathrm{T}} \cdot \boldsymbol{h}_j(\boldsymbol{x}) \right)^{\mathrm{T}} \\ &= \sum_{j=1}^{F} \mathbf{w}_{\cdot j} \left(\begin{bmatrix} \mathbf{K}_j(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ \mathbf{K}_j(\boldsymbol{x}, \boldsymbol{x}_N) \end{bmatrix}^{\mathrm{T}} (\frac{\mathbf{I}}{C} + \mathbf{K}_j)^{-1} \mathbf{Y} \right)^{\mathrm{T}},\end{aligned} \tag{10}$$

where $\mathbf{w}_{\cdot j}$ is a scalar factor of $\mathbf{B}_j$, synthesizing the confidence degree for each class and each kernel. According to the MKL theory [3, 4, 45], the optimization problem can be written as

$$\begin{aligned}\min_{\mathbf{w}_{\cdot j} \in \Delta^F} : \ & \frac{\lambda}{2} \left(\sum_{j=1}^{F} \left\| \mathbf{w}_{\cdot j} \right\|_2 \right)^2 + \frac{1}{N} \sum_{i=1}^{N} \xi_i \\ \text{s.t.} : \ & \mathbf{W}' \cdot \mathbf{B} \cdot [\hbar(\boldsymbol{x}_i, y_i) - \hbar(\boldsymbol{x}_i, y)] \geqslant 1 - \xi_i, \\ & i = 1, 2, \ldots, N, \quad y_i \neq y,\end{aligned} \tag{11}$$

where $\hbar(\cdot)$ embodies F ELM feature maps $\boldsymbol{h}_{j=1,2,\ldots,F}(\cdot)$ in the way as Eq. (1); the coefficients $\mathbf{w}_{\cdot j}$ are constrained to the simplex $\Delta^F \triangleq \{\mathbf{W}' \geqslant 0 | \|\mathbf{W}'\| \approx 1\}$; and it weighs the regularization term by λ and divides the loss term by N, which is equivalent to the proposed formulation in Ref. [4] by setting $\lambda = 1/(CN)$. [42] Considering the definition of group norm Eq. (3) and multi-class loss function Eq. (4), we can generalize Eq. (11) into

$$\min_{\mathbf{w}_{\cdot j} \in \Delta^F} : \ \frac{\lambda}{2} \|\mathbf{W}'\|_{2,p}^2 + \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}(\mathbf{W}', \mathbf{B}, \boldsymbol{x}_i, \boldsymbol{y}_i). \tag{12}$$

The parameter p ($1 < p \leqslant 2$) is used to adjust the sparsity level. When p approaches 1, the solution gets closer to the

sparse solution, but the strong convexity vanishes, making it hard to optimize. [21,41]

3.3 Online optimization of weights

The online learning algorithm learns a model by processing one data sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ at a time sequentially. At each round $t = 1, 2, \ldots, N$, the algorithm receives the training example $\boldsymbol{x}_t$ and measures the current model's prediction quality by loss function $\mathscr{L}$ defined in Eq. (4). Based on the deviation between the predicted label $\arg\max_{m\in\{1,2,\ldots,M\}} \boldsymbol{f}(\boldsymbol{x}_t)$ (Eq. (10)) and the actual label $\arg\max_{m\in\{1,2,\ldots,M\}} \boldsymbol{y}_t$, the model might be updated to improve the chances of making more accurate predictions in the subsequent steps [21]. Unlike the criteria used in Refs. [37,46] saying that the algorithm makes a "mistake" each time the prediction is not the same as the true label $\boldsymbol{y}_t$, the OM-2 [21] algorithm updates the model as long as $\mathscr{L}(\mathbf{W}', \mathbf{B}, \boldsymbol{x}_t, \boldsymbol{y}_t) > 0$ (even if the prediction is correct). This criteria of entering the model update process is also applied in OM-ELM.

According to the framework of "Follow the Regularized Leader" [5], $\mathbf{W}'$ in problem Eq. (12) at round t can be updated in an online manner with the following formula

$$\mathbf{W}'_{t+1} = \arg\min_{\mathbf{W}'} \mathcal{R}(\mathbf{W}'_t) + \mathbf{W}'_t \cdot \eta \sum_{i=1}^{t} \partial\mathscr{L}(\mathbf{W}'_i, \mathbf{B}^{(i)}, \boldsymbol{x}_i, \boldsymbol{y}_i), \quad (13)$$

where $\mathcal{R}(\mathbf{W}')$ is the regularizer that is defined in a similar way as [21]: $\mathcal{R}(\mathbf{W}') = \frac{q}{2}\|\mathbf{W}'\|^2_{2,p}$; variable η is a dynamic (changes at each round) positive real value representing the learning rate. Compared to Eq. (12), the multi-class loss function has been replaced by its subgradient $\partial\mathscr{L}(\mathbf{W}', \mathbf{B}, \boldsymbol{x}, \boldsymbol{y})$:

$$\begin{aligned} &\partial\mathscr{L}(\mathbf{W}', \mathbf{B}, \boldsymbol{x}, \boldsymbol{y}) = \mathbf{B} \cdot (\hbar(\boldsymbol{x}, y) - \hbar(\boldsymbol{x}, m)) \\ &y = \arg\max_{y\in\{1,2,\ldots,M\}} \boldsymbol{y}, \ m = \arg\max_{m\neq y} \boldsymbol{f}(\boldsymbol{x}, m). \end{aligned} \quad (14)$$

By using the Fenchel conjugate (noted as $\mathcal{R}^*$) of $\mathcal{R}$, we can derive the solution of Eq. (13) for each update as in [21,41,42]:

$$\begin{aligned} (\mathbf{W}')_{t+1} &= \nabla\mathcal{R}^*\left(-\eta \sum_{i=1}^{t} \partial\mathscr{L}(\mathbf{W}'_i, \mathbf{B}^{(i)}, \boldsymbol{x}_i, \boldsymbol{y}_i)\right), \\ \mathcal{R}^*(\boldsymbol{\gamma}) &= \frac{1}{2q}\|\boldsymbol{\gamma}\|^2_{2,q}, \\ \nabla\mathcal{R}^*(\boldsymbol{\gamma}_i) &= \frac{1}{q}\left(\frac{\|\boldsymbol{\gamma}_i\|_2}{\|\boldsymbol{\gamma}\|_{2,q}}\right)^{q-2} \boldsymbol{\gamma}_i, \quad i = 1, 2, \ldots, F. \end{aligned} \quad (15)$$

Here, variable $\boldsymbol{\gamma}$ essentially defines the mixture of kernel classifiers. It can be proved that the accumulated number of mistakes for $\aleph$ in any order is approximately equal to the one for Eq. (12). The proofs can be found in Appendix of Ref. [21]. The learning rate η for round t is calculated with

$$\eta_t = \min\left\{1 - 2\frac{\mathbf{W}'_t \cdot \partial\mathscr{L}(\mathbf{W}'_t, \mathbf{B}^{(t)}, \boldsymbol{x}_t, \boldsymbol{y}_t)}{\|\partial\mathscr{L}(\mathbf{W}'_t, \mathbf{B}^{(t)}, \boldsymbol{x}_t, \boldsymbol{y}_t)\|^2_{2,q}}, \ 1\right\}. \quad (16)$$

Provided that kernel matrices are normalized and variable p is initialized with $\frac{2\ln F}{2\ln F - 1}$, then for any $\mathbf{W}'$, the number of mistakes is bounded by $a + b + \sqrt{ab} - \sum_{t\in I}\eta_t$, in which $a = 4e\ln F\|\mathbf{W}'\|^2_{2,1}$, $b = \sum_{t=1}^{N}\mathscr{L}(\mathbf{W}', \mathbf{B}^{(t)}, \boldsymbol{x}_t, y_t)$, and I denotes the set of rounds that model update occurs without any prediction error; the proof can be found in Ref. [21].

3.4 Pseudo code and computational complexity

To provide an integral picture of our OM-ELM algorithm, the pseudo code is briefed as follows.

Algorithm OM-ELM

Require: $p = 2\ln F/(2\ln F - 1)$, $q = 1/(1 - 1/p)$, $\mathbf{W}'_1 = 0$, $\boldsymbol{\gamma}_1 = 0$, and F kernel types and their parameter(s)

Ensure: weight matrix $\mathbf{W}'$, sample set S that once had $\mathscr{L} > 0$

1: **for** each $t \in \{1, 2, \ldots, N\}$ **do**
2: Receive an instance: $\boldsymbol{x}_t$
3: Receive its label: $\boldsymbol{y}_t$, $y_t = \arg\max_{y_t\in\{1,2,\ldots,M\}} \boldsymbol{y}_t$
4: Predict with F assembled ELM classifiers: Eq. (5), Eq. (10)
$label(\boldsymbol{x}_t) = \arg\max_{m\in\{1,2,\ldots,M\}} \boldsymbol{f}(\boldsymbol{x}_t)$
5: Calculate loss function: Eq. (4)
$\mathscr{L} = \max\{\max_{m\neq y_t}\{1 - \mathbf{W}' \cdot \mathbf{B} \cdot [\hbar(\boldsymbol{x}_t, y_t) - \hbar(\boldsymbol{x}_t, m)]\}, 0\}$
6: **if** $\mathscr{L} > 0$ **then**
7: Calculate sub-gradient: (14)
$\partial\mathscr{L} = \mathbf{B} \cdot (\hbar(\boldsymbol{x}_t, y_t) - \hbar(\boldsymbol{x}_t, \arg\max_{m\neq y} \boldsymbol{f}(\boldsymbol{x}_t, m)))$
8: Calculate learning rate: (16)
$\eta = \min\left\{1 - 2(\mathbf{W}' \cdot \partial\mathscr{L})/(\|\partial\mathscr{L}\|^2_{2,q}), \ 1\right\}$
9: Update variable: $\boldsymbol{\gamma}^{t+1} = \boldsymbol{\gamma}^t + \eta\partial\mathscr{L}$
10: Calculate weights from $\boldsymbol{\gamma}$: (15)
$\mathbf{w}^{t+1}_{\cdot j} = \frac{1}{q}\left(\frac{\|\boldsymbol{\gamma}^{t+1}_i\|_2}{\|\boldsymbol{\gamma}^{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\gamma}^{t+1}_i, \quad i = 1, 2, \ldots, F$
11: **else**
12: Disgard $(\boldsymbol{x}_t, \boldsymbol{y}_t)$
13: **end if**
14: **end for**

Observed from the pseudo code of OM-ELM approach, the overall computational complexity for processing each sample (i.e., lines 2–10) can be roughly measured by $O(s_t \times M \times F)$, where s_t is the number of updates actually happened (i.e., $\mathscr{L} > 0$) before the tth round. This complexity is dominated by the kernel calculation [i.e., $\mathbf{K}_j$ in Eq. (10)] involved in the prediction step (i.e., line 4), which might seem costly in some cases, but it can be programmed cost-effectively by applying some kernel caching methods. More importantly, when s_t reached a rather large value, instead of $\mathbf{H}_j\mathbf{H}_j^{\mathrm{T}}$ (cf. Eq. (8), size: $s_t \times s_t$), $\mathbf{B}_j$ can get another equivalent solution:

$(\frac{\mathbf{I}}{C} + \mathbf{H}_j^{\mathrm{T}}\mathbf{H}_j)^{-1}\mathbf{H}_j^{\mathrm{T}}\mathbf{Y}$, where $\mathbf{H}_j^{\mathrm{T}}\mathbf{H}_j$ (cf. Eq. (6), size: $L \times L$) is used to compute kernels [39]. As in most applications, the number of hidden neurons L is much smaller than s_t: $L \ll s_t$, the computational cost reduces dramatically. As a result, the computation complexity of each model update becomes $O(L\times M\times F)$ upon condition $L \ll s_t$, which implies a trend of better computational scalability (L is irrelevant to $s_t \leqslant t \leqslant N$) over the long run and leads to less model-update time in average (cf. Table 2).

4 Experiments with robot recognition tasks

In this section, we will consecutively describe our real-world datasets collected by different robots, experimental settings, and finally comprehensive result analysis. The performance of OM-ELM is compared with several state-of-the-art online multi-kernel learning models (i.e., OM-2 [21], OMCL [36], and OMKC [37, 46]) and OS (online sequential)-ELM [32] ensembles. This experimental study covers both real-time and non-real-time tasks, addresses both object recognition and location identification problems, and handles both multi- modal and mono-modal feature fusion schemas. The overall experimental objectives are to find out: 1) the dynamic of average error rate (AER) over the number of training samples received; 2) the connection between testing accuracies and training passes (epochs); 3) the computational efficiency of OM-ELM compared to other algorithms.

4.1 Benchmark datasets for robot recognition tasks

Four datasets representing different application scenarios of autonomous robot recognition problems are evaluated. The first scenario (BDH-KDC-10 dataset) [47, 48] is to identify target objects using sequential tactile and depth data; the second scenario (KTH-IDOL2 dataset) [26] is to recognize the location of a mobile robot using sequential images and laser data; the third scenario (SD-5 dataset) [49, 50] was originally designed for assessing grasp stability of a robotic hand; the last scenario (BDH-5 dataset) [51] tries to judge if a plastic bottle contains water or not. It is worth noting that SD-5 and BDH-5 datasets only contain tactile data collected from sensor arrays mounted on fingertips of robotic hands. The dataset specification is shown in Table 1.

Dataset BDH-KDC-10 is collected on a dexterous robotic operation platform to perform grasping tasks (Fig. 1). This platform is equipped with a 4-degree-of-freedom (4-DoF) barrett dexterous hand (BDH) mounted on a 7-DoF Schunk manipulator. For the robotic hand to choose an appropriate gesture and exert proper grasping force, it has to distinguish the category of target objects first. Tactile and depth data are collected at every grasp action under different environmental conditions (i.e., illumination condition and object placement). Tactile readings are stored throughout every valid grasping process starting from the initial object contact (cf. Fig. 1(b)) until the lifting movement ceases (cf. Fig. 1(c)). The tactile sensor patches are mounted on the three fingers and the palm. The depth data collected by a Kinect Depth Camera (KDC) would greatly alleviate the dilemma that tactile sensors could not provide discriminative data before getting physical touch with the target objects. This dataset has 50 time-series data sequences for grasping objects belonging to 10 classes. The depth images (50-by-50 pixels) are described with SIFT [9] and local hue histograms [11]. As is suggested in Ref. [52], both representations are also calculated over a 3-by-1 horizontal decomposition to code parts of the spatial layout of frames, which makes the total number of depth features add up to 4. For tactile data, we concatenate the readings from the fingers and the palm, forming a large matrix as raw input. We select about 65% samples from each category as training/validation dataset, and use the rest for testing.

Table 1 Summary of main properties of four real-world robot recognition datasets

Dataset [Ref.]	Platform	Sensor spec.	# Modality	# Classes	# Train	# Test	# Kernels	Real-world problem
BDH-KDC-10 [47,48]	4-DoF BDH 7-DoF Schunk Mani. Kinect Depth Cam.	Tf(×3): 3×8 Tp: 24 Di: 50×50	2 multi-modal	10	4 856	2 614	5	Object recognition
KTH-IDOL2 [26]	PeopleBot Minnie PowerBot Dumbo	Cam.: 5fps Laser Scanner	2 multi-modal	5	5 665	5 765	4	Location identification
SD-5 [49,50]	3-finger SDH	Tf(×3): 13×6 Tf(×3): 14×6	1 mono-modal	5	10 632	7 135	6	Object recognition
BDH-5 [51]	4-DoF BDH 7-DoF Schunk Mani.	Tf(×3): 3×8	1 mono-modal	2	18 827	7 833	3	Distinguish bottle with/without water

Note: Abbreviations: BDH — Barrett Dexterous Hand, SDH — Schunk Dexterous Hand, Tf — Tactile sensor on finger, Tp — Tactile sensor on palm, Ref. — Reference, Spec. — Specification, Cam. — Camera, Mani. — Manipulator

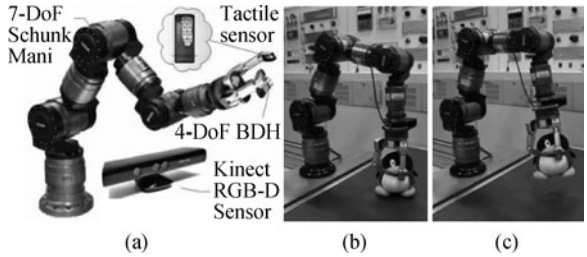


Fig. 1 The dexterous robotic operation platform where the BDH-KDC-10 dataset is collected. (a) Hardware components of the grasping system [47]; (b) start state of grasping process; (c) end state of grasping process

The KTH-IDOL2 dataset contains 24 image sequences captured by a perspective camera installed on two robots moving in an indoor environment under various weather and illumination conditions and across a time span of 6 months (cf. Fig. 2). Those images can be categorized into 5 classes representing 5 different locations (i.e., corridor, printer area, kitchen, 1-person office, and 2-person office) respectively. For our test, only 12 image sequences acquired by the same robot are considered; and they are divided into training and testing sets using the same way described in Refs. [21, 36]. We use three visual descriptors (exponential chi-square kernel: $\exp -\chi^2$) for images and one geometric feature (RBF kernel) for the laser scan sensor.

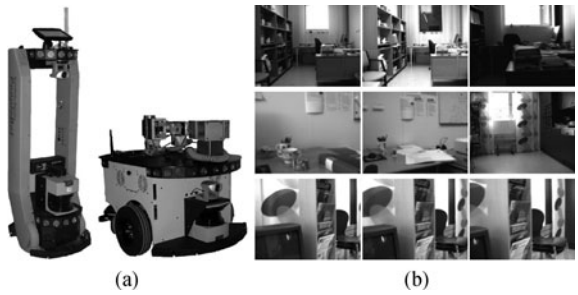


Fig. 2 The illustration of KTH-IDOL2 dataset. Figures are reproduced from [26]. (a) Mobile robot platforms employed to collect the dataset; (b) sample images showing the variations captured in the dataset

The SD-5 dataset [49, 50] is collected with the 3-finger schunk dexterous hand (SDH) which is equipped with two tactile sensor patches on each finger. An exemplary grasping execution of this hand is presented in Fig. 3(a). As presented in Fig. 3(b), there are five daily objects used for grasping: *white bottle, black cylinder, bleach cylinder, spray bottle*, and *can cylinder*. Each grasp execution provides a sequence of tactile sensor readings that is labeled as a stable/unstable grasp. The sequence consists of tactile measurements from the first physical contact with an object until the fully closed hand configuration is reached or no more changes in tactile readings occur for a specified time period. Each object was rotated after it is lifted; and the initial position of objects was not precisely aligned with respect to the hand [53]. SD-5 dataset is also annotated with the name of the corresponding object to which the grasping was applied. To compare with BDH-KDC-10 dataset, we will also use this dataset for the purpose of object instance recognition.

Fig. 3 The collection of SD-5 dataset. Figures are reproduced from [49, 50, 53]. (a) Example of grasp execution with SDH (Schunk Dexterous Hand) applied to the *can cylinder*; (b) the five target objects to be grasped

The BDH-5 dataset is collected using a 3-finger BDH, two fingers of which can rotate synchronously and symmetrically about the base joint in a finger-spreading action. Every finger has 24 tactile sensor units; and it is used to grasp 5 different bottles with water (cf. Fig. 4(b)) or without water (cf. Fig. 4(a)), whose visual appearances are quite similar and non-discriminant. Though they cannot be discriminated by visual appearance, their tactile sequences appear to be different. During data collection, each bottle is grasped for 10 times in a standing position. Each grasp action lasts 8.2 to 18 seconds. The sampling frequency is 25Hz and therefore the length of the obtained sequence is 205–450 frames. In Fig. 4(c), we show the representative grasping pose of the hand. From such a setting, we manage to construct a 10-object dataset, which includes exactly 100 tactile sequences.

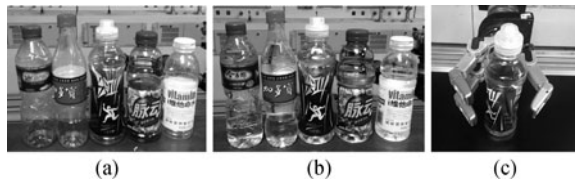


Fig. 4 The experimental setting to obtain BDH-5 dataset. (a) Five bottles without water; (b) the same five bottles filled with water; (c) an illustration of grasp pose of the manipulator

4.2 Experimental settings and parameters

The simulated experiments of OS-ELM[2)], OM-2, OMCL[3)], and OMKC[4)] are carried out with Matlab 2014a (maci64) run on Intel Core i7 2.3GHz CPU with 16GB 1333MHz RAM and 250GB SATA 6GB/s SSD. The implementation of OM-

[2)] http://www.ntu.edu.sg/home/egbhuang/elm_codes.html
[3)] http://dogma.sourceforge.net
[4)] http://www.cais.ntu.edu.sg/~chhoi/OMKC/

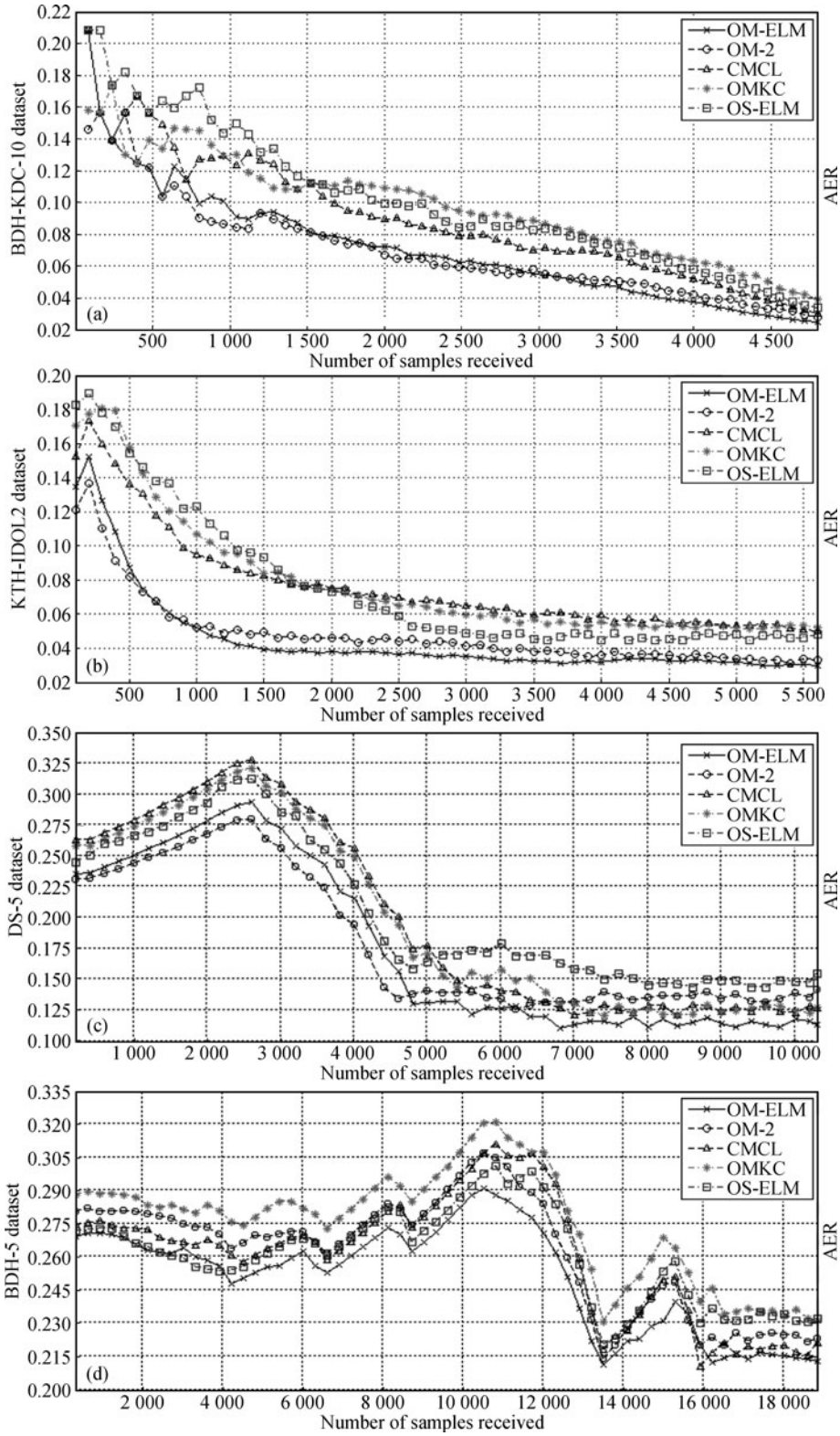


Fig. 5 Average online training error rate (AER) as a function of the number of training samples on four selected robot recognition datasets. (a), (b) Multi-modal datasets: BDH-KDC-10 and KTH-IDOL2; (c), (d) Mono-modal datasets: SD-5 and BDH-5

ELM, OM-2, and OMCL algorithms uses the Dogma toolbox [54], which is dedicated for discriminative online machine learning.

OS-ELM method is capable of handling one feature only, but we have multiple features to process at the same time. To build a learning mechanism from OS-ELM that can combine

the classification capability of F features, we use F OS-ELM networks, each of which has L hidden nodes and uses the same activation function. Each OS-ELM is trained with only one sample in each incremental step. The Gaussian activation function $\exp(-b\|x-a\|^2)$ with random parameters (i.e., a and b) is applied for each feature. Since ELM's generalization performance is good as long as the dimensionality of feature space (L) is large enough [7, 38, 39, 44], L=2 000 is set for all OS-ELM simulations. Based on our intuitive intention, we expect that the better performed (with lower average error) classifier is assigned with larger impact on the final output. So the eventual output value $f(x_t)$ of our OS-ELM ensemble at round t is calculated with the formula

$$f(x_t)=\frac{\left(\mathcal{E}_i^{(t)}\right)^{-1}}{\sum_{i=1}^{F}\left(\mathcal{E}_i^{(t)}\right)^{-1}}f^{(i)}(x_t),\ i=1,2,\ldots,F, \qquad (17)$$

where $f^{(i)}(x_t)$ is the output of the ith network at round t; and $\mathcal{E}_i^{(t)}$ denotes the accumulated AER of the ith OS-ELM classifier at round t. The average error rate (AER) is the number of prediction mistakes the online algorithm makes on a given input sequence normalized by the length of the sequence. AER is a widely used performance measure for online algorithms.

OM-2, OMCL, and OMKC belong to the class of methods that can fuse multiple feature descriptors represented with the same number of kernels in an online sequential style. Apart from measuring these algorithms continuously with AER, we also evaluate their generalization performance on a separate testing dataset, which is similar to the classic batch learning environment. It is constantly observed that the generalization performance of an online learning model trained with merely one pass (epoch) acts weaker than a batch-learning counterpart. This phenomenon becomes even more significant when the number of training samples is smaller. The term "epochs" basically means iterations or sweeps of the supervised online training that go through all training samples. That is why we ran through all training instances for 20 epochs exactly. Hoi et al. [37] claims in their tests that: 1) the stochastic updating strategy is able to significantly improve or at least maintain the performance compared to the deterministic approach; 2) the deterministic combination strategy often outperforms the stochastic combination strategy. Therefore we only concern the simulation results for OMKC$_{(S,D)}$ that applies stochastic updating and deterministic combination; and the parameter β in OMKC$_{(S,D)}$ algorithm is simply fixed to 0.8. To make it work for multi-class problems, we simply train one model for each category m at round t and take the most confident output as the final result, i.e., $\arg\max_{m\in\{1,2,\ldots,M\}}|f_m(\boldsymbol{x}_t)|,\ y_t\in\{-1,1\}$.

Motivated by the success of hierarchical matching pursuit (HMP) [55] in discovering expressive features from sensory data, we use HMP for encoding the raw tactile readings. It is claimed in Ref. [53] that only a minor performance gain might be observed when a second layer is added to HMP; and they reason that features that well represent the data are already captured at the first layer due to low dimensionality of tactile data. Hence, our tactile feature encoding also applies 1-layer spatial HMP. To exploit the intrinsic relationships between adjacent fingers, we regard fingers as different sensor input and code them separately for SD-5 and BDH-5 dataset. Since the samples arrive in a sequential manner (one after another over the timeline) in tactile recognition tasks, all of our simulations are executed in chronological order to test each algorithm's in-field effectiveness. The remaining parameters of the methods are pre-selected using a combination of grid search and manual search (e.g., Ref. [56]) on a small validation set.

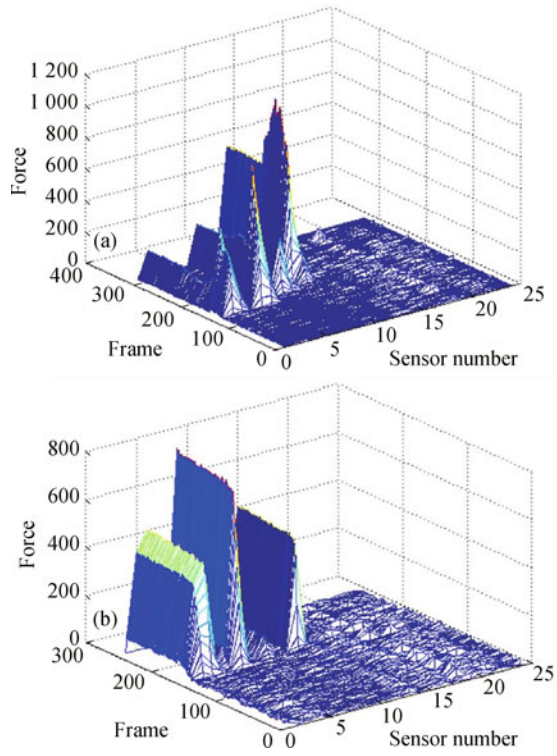


Fig. 6 Sparsity of BDH-5 dataset: visualization of the tactile sequence from one finger when grasping two different bottles. (a) Bottle A (empty); (b) Bottle C (filled with water)

4.3 Performance evaluation

Figures 5(a) and 7(a) report the training performance (AER versus number of samples received) and testing performance (testing accuracy versus number of epochs) respectively for

BDH-KDC-10 dataset. First of all, by comparing OM-ELM and OM-2, both of which adopt the same update strategy, we find that they obtain similar performances in both training (OM-ELM prevails slightly after about 3 300 samples) and testing phases (OM-2 prevails slightly since the third pass). During the training phase, AER fluctuates remarkably within 1 500 samples and starts to stabilize thereafter, which might indicate the "Plateau" [47] (i.e., firm-grasping and lifting) part of tactile data is more helpful in recognizing target objects.

For the location identification problem (i.e., KTH-IDOL2 dataset), Fig. 5(b) shows AER with various online learning algorithms as a function of the number of training samples. We observe that with more training examples received, the performance achieved by the algorithms gradually improves. In particular, OM-2 produces smaller error rates than all of the other algorithms at the beginning; when more training samples arrive, the performance of OM-ELM stabilizes with the lowest AER score. To further investigate this observation, Fig. 7(b) shows the classification performance on test sets as a function of epochs; and it validates the fact that OM-ELM achieves slightly better testing rate than OM-2 after about five epochs, while OM-2 reaches the stable state the most quickly.

SD-5 is a mono-modal dataset which contains only tactile readings; the training performance on this dataset is reported in Fig. 5(c). The AER values start becoming relatively stable when approximately 50% training examples have been used to train the algorithms. OM-ELM outperforms other methods by the end of the first epoch, but the AER difference among the five tested methods consistently lies within a limit of 6%. In contrast to the simulations on multi-modal datasets, the AER values here monotonically increase until a turning point that appears at the 2 600th sample approximately. Because one tactile sequence for a single valid grasp embodies 348 samples in average, we can largely interpret that the first seven tactile sequences failed to provide sufficient discriminative information for any object class. Figure 7(c) illustrates that the generalization ability of OM-ELM, OM-2, and OMCL keeps increasing all the way through 20 epochs; but that of OMKC and OS-ELM almost stops growing after 6 epochs. Although OM-ELM reaches a testing ratio of 86% on SD-5 dataset, Madry et al. [53] has reported their result as 98.9% using an unsupervised batch learning method named spatio-temporal HMP (ST-HMP), which extracts features from consecutive tactile frames and then pools them over the time dimension. However, our tests only focus on the current sample, which essentially means: 1) the previous samples are discarded (e.g., due to insufficient memory); 2) the upcoming samples can not be foreseen in an online learning setup. As a result, most online learning algorithms

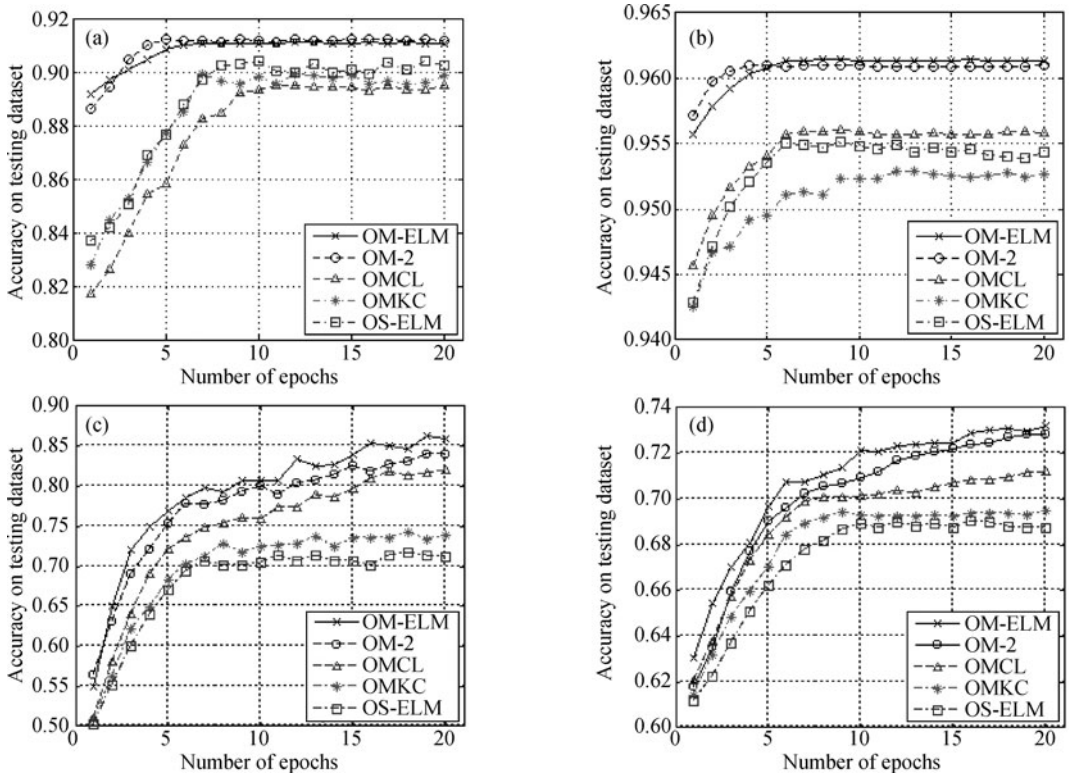


Fig. 7 Accuracy on the testing dataset as a function of the training passes (epochs). (a), (b) Multi-modal datasets: BDH-KDC-10 and KTH-IDOL2; (c), (d) Mono-modal datasets: SD-5 and BDH-5

are incapable of capturing all relevant characteristics over the temporal dimension, which might lead to an inferior training/testing accuracy to some spatio-temporal feature learning methods like ST-HMP and pLDSs (piecewise linear dynamical systems) [47].

Unfortunately in Fig. 5(d), we do not see any wide range of monotone decrease of AER values on BDH-5 dataset. The error rate stops declining after about 4 200 samples (approx. 15 tactile sequences); and starts to fluctuate thereafter until the 16 200th sample (approx. 61st tactile sequence), which we believe is caused by the reasons that 1) mono-modality feature descriptor is relatively weak to bootstrap an online algorithm, which probably also explains the initial monotone increase of AER when training on the SD-5 dataset; 2) the low dimensionality (3×8) of the tactile sensory patches do not generate high fidelity data matrices that could provide strong discriminative ability at each step of model update; 3) the tactile data is quite sparse from both spatial and temporal point of view (Fig. 6), implying that almost half of samples in one grasping sequence might be useless or even harmful to online model evolution; 4) SD-5 is a binary classification problem, in which the tactile reading is affected by not only class labels (i.e., empty/full bottle) but also other non-relevant factors such as shape, material, texture, and hardness of bottles. Figure 7(d) shows that the best performed OM-ELM gets a testing accuracy of 73.19%. We then run two batch learning algorithms (i.e., ST-HMP and pLDSs) on the same split of BDH-5 dataset, and we obtain an average testing rate of 78.43% and 88.71% respectively. Similar to the discussion in the previous paragraph, we would like to argue that it is difficult for the current online learning algorithms to make good use of temporal correlation embodied in time-series data. However, the experiments on BDH-KDC-10 and KTH- IDOL2 dataset suggest that using more than one modality could be a solution to build a strong online learning model.

Observed from experimental results from all datasets, it seems a little surprising that OMCL and ensemble of OMKC and OS-ELM achieve comparable training performances. We believe that it may be attributed to 1) $\text{OMKC}_{(\text{S,D})}$ is able to exploit all the kernel classifiers effectively by the stochastic updating strategy; and 2) the generalization performance of OS-ELM is similar to batch ELM. However, these simplified ensemble models tend to make their training performance fluctuate more than the other three algorithms when dealing with multi-modal data sources. By testing different values of p for OM-ELM and OM-2, we find that larger values ($p \in [1.8, 2]$) are favored in robot recognition tasks, which indicates that little redundant feature exists; this is in line with some of the conclusions in [21].

Table 2 Overall training time cost (measured in seconds) for 20 epochs

Algorithms	BDH-KDC-10	KTH-IDOL2	SD-5	BDH-5
OMKC	38 013	33 978.4	8 925	2 692.3
OS-ELM	10 293.4	9 410.9	3 996.2	3 419.8
OMCL	7 791.7	7 871.4	2 923.7	2 430
OM-2	12 028.2	11 490	3 493.1	2 915.2
OM-ELM	**6 362.6**	**5 926.2**	**2 808.6**	**2 396.6**
Avg./sample	**0.065 51**	**0.052 3**	**0.013 2**	**0.006 36**
Sensor freq./Hz	15	19	75	157

Note: Avg./sample (of OM-ELM) is the average model update time to process one training example. Sensor freq. is the limit of sampling frequency that OM-ELM can handle in each robot recognition problem

In our experiments, we also examine the time efficiency (Table 2) by comparing the time consumption for 20 training epochs. The most cost-efficient method in each experimental group is indicated in bold. Because multiple classifiers are trained for OS-ELM and OMKC tests, their time costs are sometimes significantly higher than the others and hence excluded mostly from our further analysis. In particular, we notice that OM-ELM consumed the least amount of time in every robot recognition problem, which could ascribe to the improved computational scalability of flexible ELM solvers (Section 3.4) with regard to the number of training samples N [39]. The average time needed for OM-ELM to process one robot sensory sample varies from 6.36 milliseconds till 65.51 milliseconds, which means it can handle real-time data flow from sensors with sampling frequencies from 15Hz to 157Hz. It might seem promising, yet we have not considered the time for activities such as kernel pre-computation and unsupervised feature learning from raw sensory readings. Table 2 also shows that OM-ELM is inclined to be more cost-efficient when the input embraces more modalities, less sparsity, and higher dimensions. In other words, OM-ELM has better scalability and runs at a faster speed (in average) than other referenced methods.

5 Conclusions and perspectives

We have proposed OM-ELM approach which is an online multi-kernel learning approach for real-world classification problems. It is essentially several kernel-based ELMs that are assembled by a novel p-norm formulation of the online MKL problem. OM-ELM allows us to tune the level of sparsity and apply an adaptive learning rate to obtain optimal training performance. The time to update the model is linear in the number of kernels, classes, and model updates. Experiments on four robot recognition datasets show that our model achieves

a superior or equivalent performance with less training-time consumption compared to OM-2 and OMCL methods, and much better results than OMKC and OS-ELM ensembles.

We also discovered that online learning is incapable of capturing all the relevant characteristics over the temporal dimension of time-series sensory data, which can lead to an inferior training and testing performance compared to some spatio-temporal batch learning methods. But experiments on multi-modal datasets suggest that incorporating more than one modality might be a solution to build a strong online learning model; it is especially necessary when one or more sensors output sparse signals.

Moreover, OM-ELM tends to consume the least amount of training time among all the tested approaches, which could be attributed to the improved computational scalability of flexible ELM solvers. Provided with pre-processed features and kernels in our setup, OM-ELM is able to handle real-time sensory data with sampling frequencies from approximately 15Hz to 157Hz. In real robot recognition tasks, we have no way to avoid the overhead of feature and kernel calculation; hence parallel computing architectures should be employed. For instance, we can run different tasks over multiple CPUs and GPUs within multiple threads; and this might be one of the valuable fields to look into in our future perspectives.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 613278050 and 61210013). The work of H B Li was supported in part by the National Natural Science Foundation of China (Grant No. 61473161), and by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2015BAK12B03).

References

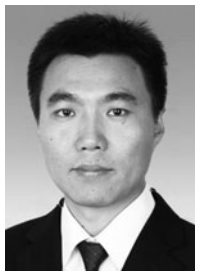
1. Sonnenburg S, Rätsch G, Schäfer C, Schölkopf B. Large scale multiple kernel learning. The Journal of Machine Learning Research, 2006, 7: 1531–1565
2. Lanckriet G R, Cristianini N, Bartlett P, Ghaoui L E, Jordan M I. Learning the kernel matrix with semidefinite programming. The Journal of Machine Learning Research, 2004, 5: 27–72
3. Zien A, Ong C S. Multiclass multiple kernel learning. In: Proceedings of International Conference on Machine Learning. 2007, 1191–1198
4. Rakotomamonjy A, Bach F, Canu S, Grandvalet Y. Simplemkl. Journal of Machine Learning Research, 2008, 9: 2491–2521
5. Shalev-Shwartz S. Online learning and online convex optimization. Foundations and Trends in Machine Learning, 2011, 4(2): 107–194
6. Cesa-Bianchi N, Lugosi G. Prediction, Learning, and Games. Cambridge: Cambridge University Press, 2006
7. Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: theory and applications. Neurocomputing, 2006, 70(1): 489–501
8. Huang G B, Chen L, Siew C K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks, 2006, 17(4): 879–892
9. Lowe D G. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 2004, 60(2): 91–110
10. Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2005, 886–893
11. Van De Weijer J, Schmid C. Coloring local feature extraction. In: Proceedings of European Conference on Computer Vision. 2006, 334–348
12. Wang H, Ullah M M, Klaser A, Laptev I, Schmid C. Evaluation of local spatio-temporal features for action recognition. In: Proceedings of British Machine Vision Conference. 2009, 1–11
13. Taylor G W, Fergus R, LeCun Y, Bregler C. Convolutional learning of spatio-temporal features. In: Proceedings of European Conference on Computer Vision. 2010, 140–153
14. Li Z, Liu J, Yang Y, Zhou X, Lu H. Clustering-guided sparse structural learning for unsupervised feature selection. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(9): 2138–2150
15. Li Z, Liu J, Tang J, Lu H. Robust structured subspace learning for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(1): 2085–2098
16. Kokar M M, Tomasik J A, Weyman J. Formalizing classes of information fusion systems. Information Fusion, 2004, 5(3): 189–202
17. Khaleghi B, Khamis A, Karray F O, Razavi S N. Multisensor data fusion: a review of the state-of-the-art. Information Fusion, 2011, 14(1): 28–44
18. Waske B, Benediktsson J A. Fusion of support vector machines for classification of multisensor data. IEEE Transactions on Geoscience and Remote Sensing, 2007, 45(12): 3858–3866
19. Reiter A, Allen P K, Zhao T. Learning features on robotic surgical tools. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2012, 38–43
20. Campos F M, Correia L, Calado J. Robot visual localization through local feature fusion: an evaluation of multiple classifiers combination approaches. Journal of Intelligent and Robotic Systems, 2015, 77(2): 377–390
21. Jie L, Orabona F, Fornoni M, Caputo B, Cesa-Bianchi N. OM-2: an online multi-class multi-kernel learning algorithm. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2010, 43–50
22. Gijsberts A, Metta G. Incremental learning of robot dynamics using random features. In: Proceedings of IEEE International Conference on Robotics and Automation. 2011, 951–956
23. Nguyen-Tuong D, Peters J. Incremental online sparsification for model learning in real-time robot control. Neurocomputing, 2011, 74(11): 1859–1867
24. Cho S, Jo S. Incremental online learning of robot behaviors from selected multiple kinesthetic teaching trials. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013, 43(3): 730–740
25. Jamone L, Natale L, Nori F, Metta G, Sandini G. Autonomous online learning of reaching behavior in a humanoid robot. International Journal of Humanoid Robotics, 2012, 9(10): 6–8
26. Luo J, Pronobis A, Caputo B, Jensfelt P. Incremental learning for place recognition in dynamic environments. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. 2007, 721–728
27. Ciliberto C, Smeraldi F, Natale L, Metta G. Online multiple instance

learning applied to hand detection in a humanoid robot. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. 2011, 1526–1532

28. Araki T, Nakamura T, Nagai T, Funakoshi K, Nakano M, Iwahashi N. Online object categorization using multimodal information autonomously acquired by robots. Advanced Robotics, 2012, 26(17): 1995–2020
29. Su L j, Yao M. Extreme learning machine with multiple kernels. In: Proceedings of IEEE International Conference on Control and Automation. 2013, 424–429
30. Liu X, Wang L, Huang G B, Zhang J, Yin J. Multiple kernel extreme learning machine. Neurocomputing, 2015, 149: 253–264
31. Cao L L, Huang W B, Sun F C. Optimization-based extreme learning machine with multi-kernel learning approach for classification. In: Proceedings of International Conference on Pattern Recognition. 2014, 3564–3569
32. Liang N Y, Huang G B, Saratchandran P, Sundararajan N. A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Transaction on Neural Networks, 2006, 17(6): 1411–1423
33. Hoang M T T, Huynh H T, Vo N H, Won Y. A robust online sequential extreme learning machine. Lecture Notes in Computer Science, 2007, 4491: 1077–1086
34. Lan Y, Soh Y C, Huang G B. Ensemble of online sequential extreme learning machine. Neurocomputing, 2009, 72(13): 3391–3395
35. Hush D, Kelly P, Scovel C, Steinwart I. QP algorithms with guaranteed accuracy and run time for support vector machines. The Journal of Machine Learning Research, 2006, 7: 733–769
36. Jie L, Orabona F, Caputo B. An online framework for learning novel concepts over multiple cues. In: Proceedings of Asian Conference on Computer Vision. 2010, 269–280
37. Hoi S C, Jin R, Zhao P, Yang T. Online multiple kernel classification. Machine Learning, 2013, 90(2): 289–316
38. Huang G B, Ding X, Zhou H. Optimization-based extreme learning machine for classification. Neurocomputing, 2010, 74(1): 155–163
39. Huang G B, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics: Cybernetics, 2012, 42(2): 513–529
40. Huang G B. What are extreme learning machines? filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle. Cognitive Computation, 2015, 7(3): 263–278
41. Orabona F, Jie L, Caputo B. Multi kernel learning with online-batch optimization. The Journal of Machine Learning Research, 2012, 13(1): 227–253
42. Orabona F, Jie L, Caputo B. Online-batch strongly convex multi kernel learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2010, 787–794
43. Fink M, Shalev-Shwartz S, Singer Y, Ullman S. Online multiclass learning by interclass hypothesis sharing. In: Proceedings of International Conference on Machine Learning. 2006, 313–320
44. Huang G B, Wang D H, Lan Y. Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics, 2011, 2(2): 107–122
45. Bach F R, Lanckriet G R, Jordan M I. Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of International Conference on Machine Learning. 2004, 6
46. Jin R, Hoi S C, Yang T. Online multiple kernel learning: algorithms and mistake bounds. Lecture Notes in Computer Science, 2010, 6331(4): 390–404
47. Xiao W, Sun F C, Liu H P, He C. Dexterous robotic-hand grasp learning using piecewise linear dynamic systems model. In: Proceedings of International Conference on Cognitive Systems and Information Processing. 2014, 845–855
48. Xiao W, Sun F C, Liu H P, Huang C. Manipulation techniques of dexterous robotic hand based on cyber-physical fusion. Journal of Tsinghua University (Science and Technology), 2013, 11: 1601–1608
49. Bekiroglu Y, Kragic D, Kyrki V. Learning grasp stability based on tactile data and hmms. In: Proceedings of IEEE International Symposium on Robot and Human Interactive Communication. 2010, 132–137
50. Bekiroglu Y, Laaksonen J, Jorgensen J A, Kyrki V, Kragic D. Assessing grasp stability based on learning and haptic data. IEEE Transactions on Robotics, 2011, 27(3): 616–629
51. Yang J, Liu H, Sun F, Gao M. Tactile sequence classification using joint kernel sparse coding. In: Proceedings of International Joint Conference on Neural Networks. 2015, 1–6
52. Lazebnik S, Schmid C, Ponce J. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2006, 2169–2178
53. Madry M, Bo L, Kragic D, Fox D. ST-HMP: Unsupervised spatio-temporal feature learning for tactile data. In: Proceedings of IEEE International Conference on Robotics and Automation. 2014, 2262–2269
54. Orabona F. DOGMA: a Matlab toolbox for online learning, 2009
55. Bo L, Ren X, Fox D. Hierarchical matching pursuit for image classification: architecture and fast algorithms. In: Proceedings of Conference on Neural Information Processing Systems. 2011, 2115–2123
56. Hinton G. A practical guide to training restricted boltzmann machines. Momentum, 2010, 9(1): 926

Lele Cao received the MS in Interactive systems engineering from KTH Royal Institute of Technology, Sweden in 2010. He is currently a PhD candidate in Department of Computer Science and Technology, Tsinghua University, China. His research interests include machine learning, neural networks, and human computer interaction.

Fuchun Sun received his PhD from the Department of Computer Science and Technology, Tsinghua University, China in 1998. He is currently a professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include intelligent control, neural networks, fuzzy systems, variable structure control, nonlinear systems, information fusion, and robotics.



Hongbo Li received his PhD from Department of Computer Science and Technology, Tsinghua University, China in 2009. He is currently an assistant professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include networked control systems and intelligent control.

Wenbing Huang received his BS in applied mathematics from Beihang University, China in 2012. He is currently a PhD candidate in Department of Computer Science and Technology, Tsinghua University, China. His research interest is machine learning.