Incremental Learning from Stream Data

Haibo He, *Senior Member, IEEE*, Sheng Chen, *Student Member, IEEE*, Kang Li, *Member, IEEE*, and Xin Xu, *Member, IEEE*

***Abstract*— Recent years have witnessed an incredibly increasing interest in the topic of incremental learning. Unlike conventional machine learning situations, data flow targeted by incremental learning becomes available continuously over time. Accordingly, it is desirable to be able to abandon the traditional assumption of the availability of representative training data during the training period to develop decision boundaries. Under scenarios of continuous data flow, the challenge is how to transform the vast amount of stream raw data into information and knowledge representation, and accumulate experience over time to support future decision-making process. In this paper, we propose a general adaptive incremental learning framework named ADAIN that is capable of learning from continuous raw data, accumulating experience over time, and using such knowledge to improve future learning and prediction performance. Detailed system level architecture and design strategies are presented in this paper. Simulation results over several real-world data sets are used to validate the effectiveness of this method.**

***Index Terms*— Adaptive classification, concept shifting, data mining, incremental learning, machine learning, mapping function.**

I. Introduction

INCREMENTAL learning has recently attracted growing attention from both academia and industry. From the computational intelligence point of view, there are at least two main reasons why incremental learning is important. First, from data mining perspective, many of today's data-intensive computing applications require the learning algorithm to be capable of incremental learning from large-scale dynamic stream data, and to build up the knowledge base over time to benefit future learning and decision-making process. Second, from the machine intelligence perspective, biological intelligent systems are able to learn information incrementally throughout their lifetimes, accumulate experience, develop spatial-temporal associations, and coordinate sensory-motor pathways to accomplish goals (goal-oriented behavior). In this paper, we mainly focus on the first issue and propose a general incremental learning framework that is able to learn from stream data for classification purpose. Certainly, we would like to note that such a framework could also benefit the research community to advance the understanding of the second issue, and hopefully provide useful suggestions to bring the community much closer to a biologically alike incremental learning capability in the long term.

Among the recent efforts on incremental learning from knowledge discovery and data analysis points of view, numerous new algorithms and architectures have been developed and successfully applied to different domains. For instance, an incremental linear discriminant analysis (ILDA) was proposed in [1] to handle the inverse of the within-class scatter matrix issue. Based on ILDA, a new algorithm, namely GSVD-ILDA, the generalized singular value decomposition LDA, was proposed and successfully applied to the face recognition problem. In [2] and [3], incremental learning for autonomous navigation systems was presented. Various experiments with mobile robots and a vision-based autonomous land vehicle (ALV) in the indoor learning environment were used to demonstrate the effectiveness of such learning methods. In [4], a system named SwiftFile was proposed to help different users to organize their e-mail messages into folders, which can be dynamically adjusted according to users' mailing habits. Some other works on incremental learning and its applications include the incremental learning fuzzy neural (ILFN) network for fault detection and classification [5], incremental learning for multi-sensor data fusion [6], incremental genetic learning for data classification [7], incremental semi-supervised learning [8], incremental learning for human-robot interaction [9], and others.

There is a controversy regarding the definition of incremental learning in the community. For instance, in [10] and [11], whether the previous data can be accessed by the current learning process in the scenario of incremental learning was debated. Besides, in [12], whether the incremental learning should be motivated to handle the unexpected emergent new class was discussed. Recently, it was presented in [13] and [14] that incremental learning should be capable of learning the new information and retaining the previously acquired knowledge, without having access to the previously seen data. Along this direction, the incremental learning framework discussed in this paper mainly focuses on two important issues: how to adaptively pass the previously learned knowledge to the presently received data to benefit learning from the new

Manuscript received November 18, 2010; revised July 22, 2011; accepted September 28, 2011. Date of publication October 31, 2011; date of current version December 1, 2011. This work was supported in part by the National Science Foundation under Grant ECCS 1053717 and the Defense Advanced Research Projects Agency Mathematics of Sensing, Exploitation, and Execution under Grant FA8650-11-1-7148.

H. He is with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881 USA (e-mail: he@ele.uri.edu).

S. Chen is with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030 USA (e-mail: schen5@stevens.edu).

K. Li is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT7 1NN, U.K. (e-mail: k.li@qub.ac.uk).

X. Xu is with the Institute of Automation, College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China (e-mail: xinxu@nudt.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2171713

raw data, and how to accumulate experience and knowledge over time to support future decision-making processes. We consider these two characteristics as the most critical aspects to understand the foundation of the adaptive incremental learning from computational intelligence point of view.

Motivated by the successful application of IMORL for video and image processing [13], in this paper we propose a general framework ADAIN that is capable of incremental learning from stream data. Different base classifiers can be integrated into this framework according to different application requirements, which provides the flexibility of using this approach across a wide range of domains. Mapping function is the key component of ADAIN that can effectively transform the knowledge from the current data chunk into the learning process of the future data chunks. Its design can be accomplished through different means such as nonlinear function approximators instead of the Euclidean distance function as used in many of the existing approaches. We also investigate the issue of incremental learning from new concepts and long sequence of data stream to provide an in-depth understanding of how the proposed approach can effectively handle such situations. Comparative study of the proposed framework with existing techniques assisted with detailed assessment metrics and significance tests are also presented in this paper.

The rest of this paper is organized as follows. In Section II, we formulate the incremental learning problem targeted by this paper. In Section III, we present the proposed ADAIN framework in detail. System-level architecture and detailed mapping function design are presented in this section. In Section IV, we present a convergence analysis of ADAIN. Section V describes the simulation configurations and presents the corresponding simulation results. In Section VI, we provide a detailed analysis of learning from long sequences of stream data using the proposed framework. Finally, Section VII concludes this paper and discusses several future research directions.

II. Problem Formulation

Considering the following learning scenario, let $\mathcal{D}_{j-1}$ represent the data chunk received between time t_{j-1} and t_j, and h_{j-1} be a hypothesis developed on $\mathcal{D}_{j-1}$. In this paper, we mainly focus on the classification task, therefore the hypothesis here specifically refers to a type of base classifier, such as a neural network, a decision tree, or any other kind of model adopted for classification. To achieve the incremental learning capability over stream data, conventionally there are two major categories of approaches.

The first group of methods employs a simple data accumulation strategy. In these methods, whenever a chunk of data is received, one simply develops a new hypothesis h_j based on all the available data sets accumulated so far $\{\ldots, \mathcal{D}_{j-1}; \mathcal{D}_j\}$ and discards the previously trained hypothesis h_{j-1}. This is a very straightforward approach without using the existing learned knowledge in h_{t-1} to help learning from new data in h_t. We would like to point out that for some memory-based approaches such as the locally weighted linear regression method, certain level of previous experience can be accumulated to avoid the "catastrophic forgetting" problem. Nevertheless, this group of methods generally require the storage of all accumulated data sets. Therefore it may not be feasible in many data-intensive real applications due to limited memory and computational resources.

The second approach employs ensemble learning methodology. The key idea of this category of approaches is to develop multiple hypotheses along the stream data, and use a combination strategy to integrate *all* or *part* of the existing hypotheses whenever a decision is needed. Briefly speaking, whenever a new chunk of data is available, either a single new hypothesis, h_j, or a set of new hypotheses $\mathcal{H}$: h_j, $j = 1, \ldots, L$, are developed based on the new data. Finally, a combination mechanism can be used to integrate all the decisions from different hypotheses to reach the final prediction. The major advantage of this approach is that storage or access to the previously observed data is not required. Instead, the knowledge has been stored in a series of hypotheses developed along the learning life. Although this category of approaches have been successfully applied to many domains, it also has it's own limitations. For instance, the knowledge learned in time period of $[t_{j-1}, t_j]$, i.e., the hypothesis h_{j-1}, cannot be directly used to benefit the learning process in $[t_j, t_{j+1}]$ though both hypotheses will participate in the final decision integration process. This means knowledge integration process exclusively happens in the final voting stage instead of the learning period. Therefore, an essential problem of incremental learning, i.e., the adaptive accumulation of experience over time and its usage in facilitating future learning process, is poorly addressed by this category of approaches.

We would also like to note that besides these two major groups of approaches, *online learning* methods have also been well studied to address the incremental learning problem. For instance, an online version Boosting and Bagging was proposed in [15] to adapt the original AdaBoost and Bagging algorithms for learning from stream data, which was customized in [16] to handle feature selection problems in image processing. In [17], the authors analyzed a family of online learning methods based on "Passive-Aggressive (PA)" algorithms. The general idea of "PA" is that upon arrival of a new data point, the new support vector machine (SVM) classifiers should be constructed in a way that it remains as close as possible to the old one, while in the meantime it should at least achieve a unit margin against the most recent example. Due to noise in dataset, the two conditions are hardly achievable simultaneously. Therefore, the authors proposed and analyzed variants of PA methods in efforts to strike a balance between the conditions [17]. In [18], a stochastic gradient-based online learning method was proposed to address the non-convex Neyman-Pearson (NP) classification problem, which is particularly well-suited for processing large-scale data sets. In [19], several online SVM algorithms, including LASVM-G, LASVM-NC, and LASVM-I, were proposed. These algorithms, while generating accurate intermediate models in its iterative steps by leveraging the duality gap, are more attractive in that computational efficiency is progressively increasing for both time and space. Another efficient SVM optimizer, PEGASOS, for online learning was proposed in [20], which decreases the number of iterations needed for SVM training

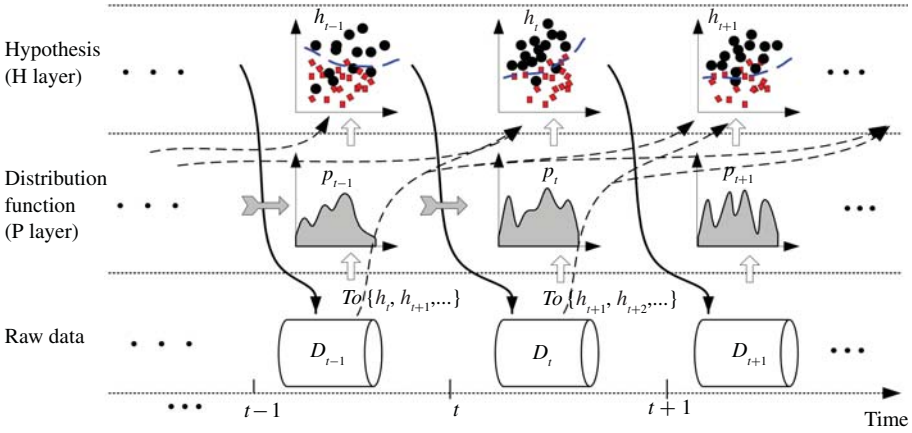

Fig. 1. ADAIN: adaptive incremental learning for classification.

significantly, and has been successfully applied to process large text corpus. The success of online learning methods can also be recognized by many high-quality implementations, including the fast out-of-core learning system by the Vowpal Wabbit project [21], which basically implements a generalized linear model using stochastic gradient descent.

Due to the importance of learning from data streams for different domains, there are also some other efforts to address this problem from different angles. For instance, an online dynamic value system for machine learning was proposed in [22]. A data-driven learning model was developed in [23]–[25], and its applications to different classification problems were reported. Growing neural gas algorithm [26], [27] was also developed to adapt the original neural gas model [28] to work in an incremental learning scenario. Based on growing cell structure and competitive Hebbian learning, it can dynamically add and delete nodes and edges to learn the topological relations in a given set of input vectors. A neural network architecture called fuzzy ARTMAP was developed for incremental supervised learning [29], in which a minimal number of new recognition categories are learned. Based on the success of the ARTMAP architecture, various modified algorithms were created for different situations, such as the Gaussian ARTMAP for incremental learning of noisy data [30], the hybrid architecture of fuzzy ARTMAP and probabilistic neural network for on-line learning and probability estimation [31], the life-long learning cell structures [32], the ellipsoid ART and ARTMAP for incremental clustering and classification [33], among others. Other related works include nearest generalized exemplar [34], generalized fuzzy min-max neural networks [35], and incremental learning based on function decomposition [36]. Interested readers can refer to [37] for further details. Data-driven adaptive learning and optimization based on adaptive dynamic programming has also been investigated in [38]–[44], which not only provide critical fundamental insight about machine intelligence research, but also provide many useful techniques and solutions for a wide range of applications domains. Finally, it is important to note that the imbalanced learning (i.e., learning from imbalanced data) over data streams has also attracted significant growing attention in the community [45]–[47]. A comprehensive and critical review of the foundations and principles of imbalanced learning, the state-of-the-art research, the current assessment metrics, as well as the major research opportunities and challenges can be found in [45].

III. Adaptive Incremental Learning from Stream Data

Motivated by the adaptive boosting principle and ensemble learning methodology [48], [49], we propose an adaptive incremental learning framework to enable knowledge accumulation and transformation to benefit learning from continuous data stream. Unlike traditional learning approaches, the objectives here are two-fold: integration of previously learned knowledge into currently received data to improve learning from new raw data, and accumulation of experience over time to support future decision-making processes.

A. Proposed Adaptive Incremental Learning Framework

Assume a learner is presented with a data flow over time. At time t, a new set of training data $\mathcal{D}_t$ is received. The previous knowledge in this case includes the hypothesis h_{t-1}, which was developed at time $t-1$ from the distribution function $\boldsymbol{P}_{t-1}$ applied to the data set $\mathcal{D}_{t-1}$. Here the distribution function can be either a sampling probability function or weight distribution function for different instances in the data. Difficult examples that are hard to learn will carry higher weights compared to those examples that are easy to learn [48]. For the first chunk of the received data, the initial distribution function $\boldsymbol{P}_1$ can be set to a uniform distribution because nothing has been learned yet. In context of this, the system-level framework of ADAIN is illustrated in Fig. 1. To demonstrate how a particular data set $\mathcal{D}_t$ received at t is learned to be integrated into the final knowledge integration process, ADAIN is algorithmically described as shown in the next page.

With the continuous stream data, a series of weight distribution functions will be developed to represent the learning capability of data in each chunk (the P layer in Fig. 1). Based on such a distribution function $\boldsymbol{P}_t$, a hypothesis h_t will be developed, in which the decision boundary is automatically forced to be more focused on the difficult-to-learn regions.

Algorithm 1 The ADAIN framework

Previous knowledge at time $(t-1)$:

− Data set, $\mathcal{D}_{t-1}$, with m instances: $\{\boldsymbol{x}_i, y_i\}, (i = 1, \ldots, m)$, where $\boldsymbol{x}_i$ is an instance in the n dimensional feature space $\mathcal{X}$ and $y_i \in \boldsymbol{Y} = \{1, 2, \ldots, c\}$ is the class identity label associated with $\boldsymbol{x}_i$.

− Distribution function: $\boldsymbol{P}_{t-1}$.

− A hypothesis, h_{t-1}, developed by the data based on $\mathcal{D}_{t-1}$ with $\boldsymbol{P}_{t-1}$, where $\boldsymbol{P}_{t-1} = [w_1^{t-1}, w_2^{t-1}, \ldots, w_m^{t-1}]$.

Current Input at time t:

− A new data set, $\mathcal{D}_t$, with m' instances, where m' may or may not be the same size as m, and can be represented as $\{\boldsymbol{x}_j, y_j\}, (j = 1, \ldots, m')$.

Learning procedure:

(1) Estimate the initial distribution function for $\mathcal{D}_t$.

$$\hat{\boldsymbol{P}}_{t-1} = \varphi(\mathcal{D}_{t-1}, \mathcal{D}_t, P_{t-1}) \tag{1}$$

where φ is a mapping function.

(2) Apply hypothesis h_{t-1} to $\mathcal{D}_t$, calculate the pseudo-error of h_{t-1}

$$\varepsilon_{t-1} = \sum_{j:h_{t-1}(\boldsymbol{x}_j)\neq y_j} \hat{\boldsymbol{P}}_{t-1}(j) \tag{2}$$

(3) Set $\beta_{t-1} = \varepsilon_{t-1}/(1-\varepsilon_{t-1})$.

(4) Update the distribution function for $\mathcal{D}_t$:

$$\boldsymbol{P}_t(j) = \frac{\hat{\boldsymbol{P}}_{t-1}(j)}{Z_t} \times \begin{cases} \beta_{t-1} & \text{if } h_{t-1}(\boldsymbol{x}_j) = y_j \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

where Z_t is a normalization constant so that $\boldsymbol{P}_t$ is a distribution, and $\boldsymbol{P}_t$ can be represented as $\boldsymbol{P}_t = [w_1^t, w_2^t, \ldots, w_{m'}^t]$.

(5) A hypothesis h_t is developed by the data example based on $\mathcal{D}_t$ with $\boldsymbol{P}_t$.

(6) Repeat the procedure when the next chunk of new data sets $\mathcal{D}_{t+1}$ is received.

Output: The final hypothesis:

$$h_{final}(\boldsymbol{x}) = \arg\max_{y\in Y} \sum_{t:h_T(\boldsymbol{x})=y} \log\left(\frac{1}{\beta_t}\right) \tag{4}$$

where T is the set of incrementally developed hypotheses in the learning life.

After $\boldsymbol{P_t}$ and h_t have been obtained, the system uses its knowledge to facilitate learning from the next chunk of raw data, $\mathcal{D}_{t+1}$. This is achieved by the top-down and horizontal signal flow, as illustrated in Fig. 1. The objective here is to inherit the adaptive boosting characteristic to improve incremental learning.

There are two mechanisms in the proposed ADAIN framework to facilitate adaptive incremental learning. First, a mapping function φ (1) is used to estimate the initial distribution function $\hat{\boldsymbol{P}}_{\boldsymbol{t-1}}$ for $\mathcal{D}_t$. This function could be customized in accordance with the requirements of specific applications.

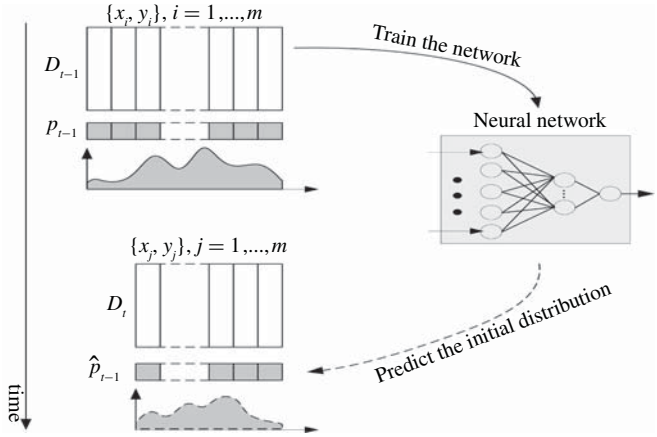

Fig. 2. Mapping function based on MLP.

The objective of the φ function is to provide a quantitative estimation of the learning capability of the new chunk of data based on previously trained classifier. We will discuss this in detail in Section III-B. Second, the initial estimation $\hat{\boldsymbol{P}}_{\boldsymbol{t-1}}$ is applied to the new chunk of data $\mathcal{D}_t$ to calculate the pseudo-error (2), which represents the goodness-of-learning when the previous knowledge h_{t-1} is applied to the new data. Similar to AdaBoost [48], [49], β_t is calculated as a function of ε_t. This in turn is used to refine the distribution function in (3). In this way, misclassified instances (difficult examples) will receive higher weights, and the learning algorithm will adaptively push the decision boundary to focus on those hard-to-learn instances. Furthermore, since the hypothesis developed at the previous time step is used to evaluate its performance over the current data chunk, this framework implicitly takes into consideration all previous domain data sets for the current hypothesis for knowledge accumulation and transformation without requiring access to previous data.

B. Design of the Mapping Function

In the proposed learning framework, the mapping function φ (1) provides a connection from past experience to the newly received data, and adapts such knowledge to data sets received in future. Therefore, the design of the mapping function φ is of critical importance to this framework. When the classic boosting idea is applied to traditional static learning problem [48], [49], the weights can be updated iteratively based on the static training data in a sequential manner. However, in incremental learning scenario, one cannot directly obtain/update such weights when a new chunk of the data flow is received. In this section, we propose to use nonlinear regression models as the mapping function to achieve the goal.

Let us take the general function approximator of neural network with multilayer perceptron as an example here (we abbreviate this as "MLP" in the rest of this paper). Fig. 2 shows a high-level structure on this. Based on the previous data information, $\mathcal{D}_{t-1}$, and its associated distribution function, $\boldsymbol{P}_{t-1}$, one can develop an MLP model to learn the relationships between the feature space and its corresponding numerical weight function, $\boldsymbol{P}_{t-1}$. Then, when the new chunk

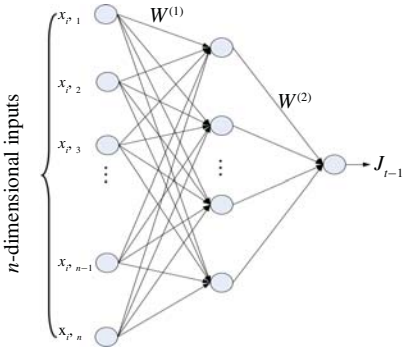

Fig. 3. Implementation details of MLP for distribution function estimation.

of data, $\mathcal{D}_t$ is received, one can use the trained MLP to obtain the initial estimation of the distribution function.

To show this idea clearly, Fig. 3 illustrates a detailed design structure with MLP. Here $\{x_{i,q}\}$, $(i = 1, \ldots, m; q = 1, \ldots, n)$ represents the n-dimensional feature space of an instance $\boldsymbol{x}_i$ in data chunk $\mathcal{D}_{t-1}$. J_{t-1} is the currently estimated distribution function output. $\mathcal{W}^{(1)}$ represent the connection weights of input to hidden layer neurons, and $\mathcal{W}^{(2)}$ represent the connection weights from hidden layer neurons to output neuron. Backpropagation [50], [51] is the key to tune the parameters of $\mathcal{W}^{(1)}$ and $\mathcal{W}^{(2)}$ to learn the relationship between the inputs and the distribution function. To do so, the error function can be defined as

$$e(k) = J_{(t-1)}(k-1) - P_{(t-1)}(k-1); \qquad E(k) = \frac{1}{2}e^2(k) \quad (5)$$

where k represents the backpropagation training epoch, $J_{t-1}(k-1)$ and $\boldsymbol{P}_{t-1}(k-1)$ represent the estimated value and target value of the distribution function for data $\mathcal{D}_{t-1}$, respectively. For clear presentation, in the following discussion we drop the subscript $(t-1)$ to derive the update rule of $\mathcal{W}^{(1)}$ and $\mathcal{W}^{(2)}$.

To calculate backpropagation, one can define the neural network output as (see Fig. 3)

$$J(k) = \frac{1 - e^{-\upsilon(k)}}{1 + e^{-\upsilon(k)}} \quad (6)$$

$$\upsilon(k) = \sum_{f=1}^{N_h} w_f^{(2)}(k) g_f(k) \quad (7)$$

$$g_f(k) = \frac{1 - e^{-h_f(k)}}{1 + e^{-h_f(k)}}, \qquad f = 1, \ldots, N_h \quad (8)$$

$$h_f(k) = \sum_{q=1}^{n} w_{f,q}^{(1)}(k) x_{i,q}(k), \qquad f = 1, \ldots, N_h \quad (9)$$

where h_f is the fth hidden node input of the neural network and g_f is the corresponding output of the hidden node, υ is the input to the output node of the network before the sigmoid function, N_h is the number of hidden neurons of the network, and n is the total number of inputs to the network.

Therefore, one can apply backpropagation to update the weights of the neural network to learn the relationship between the feature space and corresponding distribution function. We detail this procedure in the following paragraph.

Weight adjustment for the hidden to the output layer $\Delta w^{(2)}$

$$\Delta w_f^{(2)} = \alpha(k) \left[-\frac{\partial E(k)}{\partial w_f^{(2)}(k)} \right], \quad (10)$$

$$\frac{\partial E(k)}{\partial w_f^{(2)}(k)} = \frac{\partial E(k)}{\partial J(k)} \frac{\partial J(k)}{\partial \upsilon(k)} \frac{\partial \upsilon(k)}{\partial w_f^{(2)}(k)} = e(k) \cdot \frac{1}{2}(1 - (J(k))^2) \cdot g_f(k). \quad (11)$$

Weight adjustments for the input to the hidden layer $\Delta w^{(1)}$

$$\Delta w_{f,q}^{(1)} = \alpha(k) \left[-\frac{\partial E(k)}{\partial w_{f,q}^{(1)}(k)} \right], \quad (12)$$

$$\frac{\partial E(k)}{\partial w_{f,q}^{(1)}(k)} = \frac{\partial E(k)}{\partial J(k)} \frac{\partial J(k)}{\partial \upsilon(k)} \frac{\partial \upsilon(k)}{\partial g_f(k)} \frac{\partial g_f(k)}{\partial h_f(k)} \frac{\partial h_f(k)}{\partial w_{f,q}^{(1)}(k)} = e(k) \cdot \frac{1}{2}(1 - (J(k))^2) \cdot w_f^{(2)}(k) \cdot \frac{1}{2}(1 - g_f^2(k)) \cdot x_{i,q}(k) \quad (13)$$

where $\alpha(k)$ is a learning rate. Once the neural network is trained, it can be used to predict the estimated initial distribution function $\hat{\boldsymbol{P}}_t$ for $\mathcal{D}_t$. This will only require the feedforward propagation in the MLP based on the received new data feature space in $\mathcal{D}_t$.

We would like to note that alternative strategies might be adopted in the P layer as well. For instance, technically speaking, other types of the regression models, such as support vector regression (SVR) [52] and classification and regression tree (CART) [53], can also be integrated into the proposed learning framework for the P layer design. This will provide the flexibility of using the proposed approach as a general learning framework according to different application requirements. Let us take SVR as an example here. Suppose $y = f(\boldsymbol{x})$ is the estimated initial weight for instance $\boldsymbol{x} \in \mathcal{R}^n$

$$f(\boldsymbol{x}) = \langle \boldsymbol{s}, \boldsymbol{x} \rangle + b \quad (14)$$

where $\boldsymbol{s}$ and b are the slope and intercept of linear estimation function $f(\boldsymbol{x})$, and $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{R}^n$, e.g., $||\boldsymbol{s}||^2 = \langle \boldsymbol{s}, \boldsymbol{s} \rangle$. To obtain an accurate estimation $f(\boldsymbol{x})$, (14) can be transformed into a convex optimization problem

$$\arg\min_{\boldsymbol{s}} \tfrac{1}{2} \|\boldsymbol{s}\|^2$$
$$\text{s.t.} \; \|y - \langle \boldsymbol{s}, \boldsymbol{x} \rangle - b\| < \varepsilon.$$

This assumes that such a $f(\boldsymbol{x})$ actually exists that can approximate $\langle y, \boldsymbol{x} \rangle$ with ε precision, which in other words makes the convex optimization feasible. Using the final distribution function in current data chunk as the target value y for the estimation of SVR, we can solve the convex optimization problem and obtain $\boldsymbol{s}$ and b, which can be kept in memory until the arrival of the next data chunk for estimating its initial distribution function using (14). Interested readers can refer to [54] for details on SVR.

Also, in our previous work in [13], a Euclidean distance function was used to estimate the distance between different data sets to facilitate weight updates, and we will compare the performance of the proposed strategy in this paper with

that approach as well. Indeed, Euclidean distance is a straightforward implementation of the proposed incremental learning framework. The rationality behind this idea is that the similarity between data residing in consecutive data chunks could be quantized by their Euclidean distance in feature space. While it is not as competitive as nonlinear mapping function such as neural network, it has the advantage of processing/learning the data in a significantly faster fashion. We will further discuss and analyze this in the simulation section of this paper.

We would also like to note that different mapping function designs will have different levels of data accessibility requirement. For instance, in the aforementioned neural network-based mapping function design, the proposed approach does not require access to any previously received data, which fits naturally into the incremental learning scenario as discussed in many current work [14]. On the other hand, if Euclidean distance function is adopted for the mapping function design, one will need to access limited amount of previous data to build the quantitative relationship between two consecutive data chunks. Even under such a situation, we consider this assumption is still valid and in fact has also been adopted in the community. For instance, in [55], a small portion of data in the previous data chunk is preserved over time to facilitate the incremental learning process. In such situations, since only a small part of previous data chunk need to be accessed at any given time during the incremental learning period, the so-called "capacity-overflow" issue does not exist.

IV. Theoretical Analysis of ADAIN

Inspired by the decision theoretic generalization analysis of the AdaBoost algorithm [49], we present a theoretical convergence analysis of the ADAIN framework in this section. Here we consider a two-class classification problem with stream data, in which the class label can be represented as either 0 or 1 [48], [49]. We assume that the data chunks presented to the learning system are all of the same size m. We would like to note that this assumption is mainly for clear presentation purpose rather than a limitation of the theoretical analysis in this section. Furthermore, for clear presentation, we assume the output of the mapping function of (1) can be represented in the following way:

$$\hat{\boldsymbol{P}}_{t-1} = \varphi(\mathcal{D}_{t-1}, \mathcal{D}_t, P_{t-1}) = \boldsymbol{w}_j^{t-1} \cdot \boldsymbol{\alpha}_j^{t-1}. \tag{15}$$

In this way, we can consider any element of $\boldsymbol{\alpha} = [\alpha_1^{t-1}, \alpha_2^{t-1}, \cdots, \alpha_m^{t-1}]$ as a real number in the range of $(0,\ 1]$. This can be understood in an intuitive way: when two examples in two consecutive data chunks are significantly far away from each other in feature space, then the value of mapping coefficient α should be very close to 0. On the other hand, when these two examples are exactly matched with each other, then the α should be equal to 1. We would also like to point out that (15) in fact reflects a quantitative expression as presented by the mapping function design in (1). Conceptually speaking, any mapping function design as discussed in Section III will be able to be represented in this way. For instance, when Euclidean distance function is used as the mapping function design, (15) can be easily justified since the Euclidean distance function can directly estimate the distances between different data sets, which can be used to modify the previous weights to obtain the initial estimation of the new weights for the new data set. When regression models such as neural network MLP is used as the mapping function design, (15) is still effective because the neural network basically is used to find the mapping from the new data set feature space to the weight space by properly tuning the neural network weights through backpropagation training rule. In this case, one can consider the $\boldsymbol{\alpha}$ to be represented by the neural network weights (i.e., a nonlinear function approximator).

For two-class problems, we can rewrite the mathematical representation of several equations in ADAIN framework for our analysis convenience as follows. Note that the modified equations are in essence the same with the original ones.

The error calculation as defined in (2) can be rewritten as

$$\varepsilon_{t-1} = \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} |h_{t-1}(\boldsymbol{x}_j) - y_j|. \tag{16}$$

The weight update (3) can be rewritten as

$$w_j^t = w_j^{t-1} \beta_{t-1}^{1-|h_{t-1}(\boldsymbol{x}_j)-y_j|} \alpha_j^{t-1}. \tag{17}$$

Based on (17), we can get

$$\sum_{j=1}^{m} w_j^t = \sum_{j=1}^{m} w_j^{t-1} \beta_{t-1}^{1-|h_{t-1}(\boldsymbol{x}_j)-y_j|} \alpha_j^{t-1}. \tag{18}$$

And finally, the h_{final} defined in (4) can be rewritten as

$$h_{final}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \displaystyle\sum_{t=1}^{T} \left(\log\left(\frac{1}{\beta_t}\right) h_t(\boldsymbol{x}) \right) \geq \frac{1}{2} \sum_{t=1}^{T} \log\left(\frac{1}{\beta_t}\right) \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Given the inequation $\theta^\gamma \leq 1 - (1-\theta)\gamma$ for $\theta \in [0,\ 1]$, $\gamma \in [0,\ 1]$ [49], we have

$$\begin{aligned} \sum_{j=1}^{m} w_j^t &\leq \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} (1 - (1-\beta_{t-1})(1 - |h_{t-1}(\boldsymbol{x}_j) - y_j|)) \\ &= \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} (\beta_{t-1} + |h_{t-1}(\boldsymbol{x}_j) - y_j|) \\ &\quad - \beta_{t-1} \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} |h_{t-1}(\boldsymbol{x}_j) - y_j| \\ &= \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} \beta_{t-1} - \sum_{j=1}^{m} w_j^{t-1} (\beta_{t-1} - 1) \varepsilon_{t-1} \\ &\leq \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} 2\beta_{t-1} - \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} (\beta_{t-1} - 1) \varepsilon_{t-1} \\ &= \sum_{j=1}^{m} w_j^{t-1} \alpha_j^{t-1} (1 - (1-\beta_{t-1})(1-\varepsilon_{t-1})). \end{aligned} \tag{20}$$

Since $\alpha \in (0,\ 1]$, from (20) one can get

$$\sum_{j=1}^{m} w_j^t \le \left(\sum_{j=1}^{m} w_j^{t-1}\right)(1-(1-\varepsilon_{t-1})(1-\beta_{t-1})). \quad (21)$$

In this way, one can write (21) recursively

$$\begin{aligned}\sum_{j=1}^{m} w_j^t &\le \left(\sum_{j=1}^{m} w_j^{t-1}\right)(1-(1-\varepsilon_{t-1})(1-\beta_{t-1}))\\ &\le \left(\sum_{j=1}^{m} w_j^{t-2}\right)(1-(1-\varepsilon_{t-1})(1-\beta_{t-1}))\\ &\quad\cdot(1-(1-\varepsilon_{t-2})(1-\beta_{t-2}))\\ &\le \cdots\\ &\le \prod_{i=1}^{t-1}(1-(1-\varepsilon_i)(1-\beta_i)). \end{aligned} \quad (22)$$

This means

$$\sum_{j=1}^{m} w_j^{T+1} \le \prod_{t=1}^{T}(1-(1-\varepsilon_t)(1-\beta_t)). \quad (23)$$

Similar to the AdaBoost analysis, the final hypothesis makes a mistake on instance $\boldsymbol{x}_i$ only if the following condition is satisfied [49]

$$\prod_{t=1}^{T}\beta_t^{-|h_t(\boldsymbol{x}_i)-y_i|} \ge \left(\prod_{t=1}^{T}\beta_t\right)^{-\frac{1}{2}}. \quad (24)$$

To further understand this, we discuss two situations when the final hypothesis makes a mistake. First, suppose the true class label for instance $\boldsymbol{x}_i$ is $y_i = 0$. Then based on (24), one can get

$$\begin{aligned}\prod_{t=1}^{T=1}\beta_t^{-h_t(\boldsymbol{x}_i)} &\ge \left(\prod_{t=1}^{T}\beta_t\right)^{-\frac{1}{2}}\\ &\Rightarrow \sum_{t=1}^{T}\left(\log\frac{1}{\beta_t}\right)h_t(\boldsymbol{x}_i) \ge \frac{1}{2}\sum_{t=1}^{T}\log\frac{1}{\beta_t}\\ &\Rightarrow h_{final}(\boldsymbol{x}_i) = 1. \end{aligned} \quad (25)$$

This is consistent with the conjecture made by (24) that h_{final} here makes a mistake on predicting the class label of $\boldsymbol{x}_i$.

On the other hand, if the true class label for instance $\boldsymbol{x}_i$ is $y_i = 1$. Based on (24), it can be inferred that

$$\begin{aligned}&\prod_{t=1}^{T}\beta_t^{-(y_i-h_t(\boldsymbol{x}_i))} \ge \left(\prod_{t=1}^{T}\beta_t\right)^{-\frac{1}{2}}\\ &\Rightarrow \sum_{t=1}^{T}\log\frac{1}{\beta_t} - \sum_{t=1}^{T}\left(\log\frac{1}{\beta_t}\right)h_t(\boldsymbol{x}_i) \ge \frac{1}{2}\sum_{t=1}^{T}\log\frac{1}{\beta_t}\\ &\Rightarrow \frac{1}{2}\sum_{t=1}^{T}\log\frac{1}{\beta_t} \ge \sum_{t=1}^{T}\left(\log\frac{1}{\beta_t}\right)h_t(\boldsymbol{x}_i)\\ &\Rightarrow h_{final}(\boldsymbol{x}_i) = 0. \end{aligned} \quad (26)$$

TABLE I
INFORMATION REGARDING THE SIMULATION DATA SETS

name	# feature	# example	# class
Spambase	57	4601	2
Magic	10	19020	2
Waveform	40	5000	3
Sat	36	6435	6

This is also consistent with the conjecture made by (24) that h_{final} has made a mistake on predicting the class label of $\boldsymbol{x}_i$.

According to (17), at any given time during the incremental learning period, the present weight can be represented by applying (17) recursively

$$\begin{aligned}w_i^{T+1} &= d(i)\cdot\prod_{t=1}^{T}\alpha_i^t\beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|}\\ &= d(i)\cdot\prod_{t=1}^{T}\alpha_i^T\cdot\prod_{t=1}^{T}\beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|}. \end{aligned} \quad (27)$$

Based on (24), one can get

$$\prod_{t=1}^{T}\beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|} \ge \left(\prod_{t=1}^{T}\beta_t\right)^{\frac{1}{2}}. \quad (28)$$

It is also straightforward to have

$$\sum_{j=1}^{m} w_j^{T+1} \ge \sum_{j:\ h_{final}(\boldsymbol{x}_j)\neq y_j} w_j^{T+1}. \quad (29)$$

Therefore, from (27)–(29), one can get

$$\sum_{j=1}^{m} w_j^{T+1} \ge \left(\sum_{j:\ h_{final}(\boldsymbol{x}_j)\neq y_j} d(j)\cdot\prod_{t=1}^{T}\alpha_j^t\right)\cdot\left(\prod_{t=1}^{T}\beta_t\right). \quad (30)$$

Let us define $k = 1/(\prod_{t=1}^{T}(\min_i \alpha_i^t))$, based on (30), we have

$$\sum_{j=1}^{m} w_j^{T+1} \ge \frac{1}{k}\cdot\sigma\left(\prod_{t=1}^{T}\beta_t\right)^{\frac{1}{2}} \quad (31)$$

where $\sigma = \sum_{j:\ h_{final}(\boldsymbol{x}_j)\neq y_j} d(j)$ is the training error of the final hypothesis h_{final}.

By combining together (23) and (31), one can get

$$\sigma \le \prod_{t=1}^{T} k\cdot\frac{1-(1-\varepsilon_t)(1-\beta_t)}{\sqrt{\beta_t}}. \quad (32)$$

Similar in AdaBoost method, if we assume the base classifier can do better than random guess, we can define $\varepsilon_t = (1/2) - \gamma_t$, where $\gamma_t \in (0,\ (1/2))$, then

$$\sigma \le \prod_{t=1}^{T}\frac{\sqrt{1-4\gamma_t^2}}{\prod_{t=1}^{T}(\min_i \alpha_i^t)}. \quad (33)$$

Equation (33) provides an upper bound for the training error of the final hypothesis h_{final}.

TABLE II
AVERAGED PREDICTION ACCURACY

Data sets	Methods	Prediction accuracy						
		class 1	class 2	class 3	class 4	class 5	class 6	Overall
Spambase	ADAIN.MLP	0.8820	0.9352	–	–	–	–	0.9142
	ADAIN.SVR	0.8990	0.9205	–	–	–	–	0.9120
	IMORL	**0.9106**	0.8929	–	–	–	–	0.9000
	Accumulation	0.8803	0.9190	–	–	–	–	0.9038
	Learn^{++}	0.8532	**0.9561**	–	–	–	–	**0.9143**
Magic	ADAIN.MLP	0.9315	0.7137	–	–	–	–	0.8549
	ADAIN.SVR	0.9319	0.7395	–	–	–	–	**0.8644**
	IMORL	0.8404	**0.7836**	–	–	–	–	0.8205
	Accumulation	0.8670	0.7410	–	–	–	–	0.8268
	Learn^{++}	**0.9523**	0.6786	–	–	–	–	0.8547
Waveform	ADAIN.MLP	0.7843	0.8230	0.8193	–	–	–	0.8132
	ADAIN.SVR	0.7576	0.8198	0.8474	–	–	–	0.8077
	IMORL	0.7575	0.8000	0.8009	–	–	–	0.7814
	Accumulation	0.7070	0.7558	0.7534	–	–	–	0.7384
	Learn^{++}	**0.7870**	**0.8360**	**0.9072**	–	–	–	**0.8428**
Sat	ADAIN.MLP	0.9602	**0.9131**	0.9169	0.4837	0.6417	0.8494	0.8387
	ADAIN.SVR	0.9584	0.8889	0.9305	0.5180	0.7235	0.8345	0.8471
	IMORL	0.9000	0.8918	0.8566	**0.5653**	0.6841	0.7897	0.8079
	Accumulation	0.9452	0.9473	0.8697	0.5316	**0.7971**	0.8499	0.8454
	Learn^{++}	**0.9696**	0.8860	**0.9327**	0.5651	0.6958	**0.8545**	**0.8558**

TABLE III
RUNNING TIME FOR ADAIN.SVR AND LEARN^{++} (*in seconds*)

Data set	ADAIN.SVR		Learn^{++}	
	training	*testing*	*training*	*testing*
Spambase	128.42	2.19	600.26	246.62
Magic	7256.18	14.36	107 256.33	1623.74
Waveform	163.51	3.39	548.84	238.1016
Sat	352.31	4.95	1246.62	346.53

V. SIMULATION ANALYSIS

In order to validate the performance of the proposed framework, four real-world data sets with varied size and number of classes from UCI machine learning repository [56] are employed for empirical study in this research. The detailed information of these data sets can be found in Table I.

In this simulation, each data set is initially randomly sliced into 20 chunks with identical size. At each run, one chunk is randomly selected to be the testing data, and the remaining 19 chunks are sequentially fed to the proposed framework. Simulation results for each data sets are averaged across 20 runs. CART is employed as the base learner in our current study. For the nonlinear mapping function design, we adopted the MLP structure with ten hidden layer neurons and one output neuron. The number of input neurons is set to be equal to the number of features for each data set. The training epochs of the MLP is set to be 1000. As we mentioned at the end of Section III-B, other regression models can also be integrated into the proposed framework. To that end, we also employ the polynomial SVR model as the mapping function for the proposed framework. For clear presentation, we use ADAIN to abbreviate the proposed adaptive incremental learning framework. Therefore, these two design strategies are represented as "ADAIN.MLP" and "ADAIN.SVR" in the remaining of this paper.

In our experiment, we have included Learn^{++} [57], IMORL [13], and the data accumulation strategy as discussed in Section II (abbreviated as "Accumulation" in the rest of this paper). Our major focus here is to demonstrate that the proposed framework can automatically accumulate experience over time, and use such knowledge to benefit future learning and prediction process to achieve competitive results.

Table II shows the numerical accuracy for these data sets, including both the overall classification performance as well as the prediction accuracy for each individual class for each data set. It can be intuitively observed that the proposed approach can remarkably improve the learning performance compared to the method in [13] and data accumulation learning strategy. As expected, Learn^{++} seems to be able to provide most of the competitive results across these data sets. This is mainly because Learn^{++} is built on the "Ensemble-of-Ensemble" strategy. However, this "Ensemble-of-Ensemble" strategy also means Learn^{++} requires much more computational resources. Table III shows the one round running time of ADAIN.SVR and Learn^{++} for all data sets. Obviously, Learn^{++} spends much more time than ADAIN.SVR in learning from streamed data chunks. Particularly, when the size of the data set is large, e.g., magic data set, the time consumption of Learn^{++} grows exponentially, which limits its scalability to handle large-scale stream data. Fig. 4 visualizes the overall prediction accuracy tendency over time for algorithms under comparison, where Fig. 4(a)–(d) represents the data sets "spambase," "magic," "waveform," and "sat," respectively. Learn^{++} has been excluded from these figures, since it creates multiple hypotheses for each data chunk. From these figures, one can clearly see that the ADAIN can improve its learning performance over time, which means the system can adap-

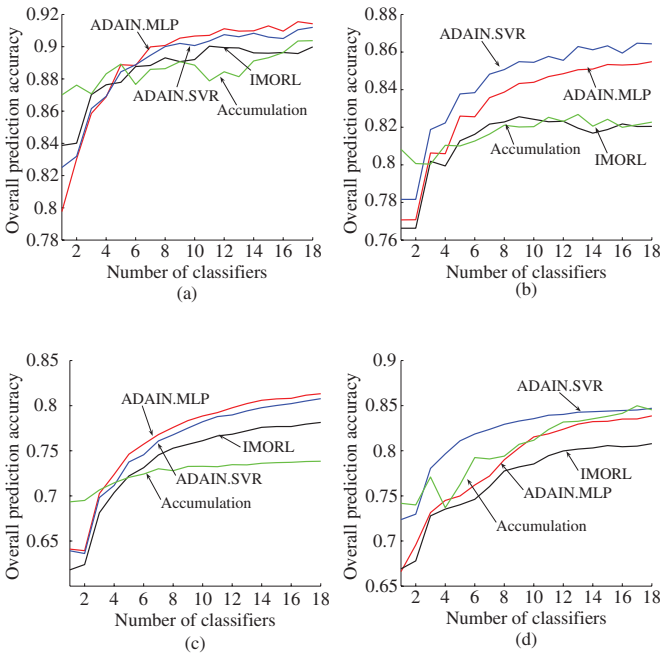

Fig. 4. Prediction overall accuracy. (a) Spambase. (b) Magic. (c) Waveform. (d) Sat.

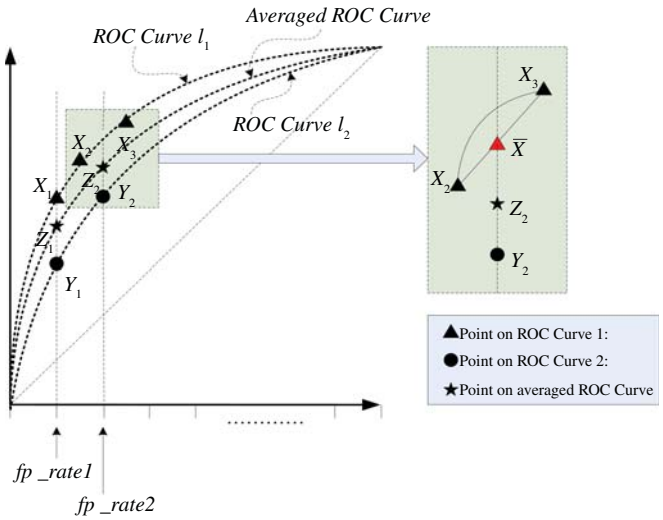

Fig. 5. Vertical averaging approach of ROC curves.

TABLE IV
t-TEST FOR PREDICTION ACCURACY

Data set	ADAIN.MLP		IMORL		$\|Z\|$	Accept or reject H_0
	μ	σ	μ	σ		
Spambase	0.9142	0.0230	0.8999	0.0215	2.0312	*Reject*
Magic	0.8549	0.0091	0.8205	0.0168	8.0576	*Reject*
Waveform	0.8132	0.0102	0.7814	0.0117	9.1696	*Reject*
Sat	0.8387	0.0493	0.8079	0.0351	2.2736	*Reject*
Data set	ADAIN.MLP		Accumulation		$\|Z\|$	Accept or reject H_0
	μ	σ	μ	σ		
Spambase	0.9142	0.0230	0.9038	0.0057	1.9628	*Reject*
Magic	0.8549	0.0091	0.8227	0.0131	9.0142	*Reject*
Waveform	0.8132	0.0102	0.7384	0.0073	26.6892	*Reject*
Sat	0.8387	0.0493	0.8454	0.0593	0.3885	*Accept*
Data set	ADAIN.MLP		Learn^{++}		$\|Z\|$	Accept or reject H_0
	μ	σ	μ	σ		
Spambase	0.9142	0.0230	0.9143	0.0195	0.0148	*Accept*
Magic	0.8549	0.0091	0.8547	0.0071	0.0775	*Accept*
Waveform	0.8132	0.0102	0.8428	0.0091	9.6841	*Reject*
Sat	0.8387	0.0493	0.8558	0.0221	1.4155	*Accept*

tively learn from stream data, and accumulate knowledge to facilitate the future learning and decision-making processes. The reason why ADAIN provides a much better prediction results than IMORL is that by employing a strong function approximator other than Euclidean distance-based approach as in IMORL, ADAIN can more accurately estimate the initial data distribution for each data chunk based on the knowledge obtained from the previous one.

We now conduct the significance tests to have an in-depth understanding of the performance across all the methods. The signed test [58] calculates the significance between two algorithms by counting the wins, losses, and ties of all the runs. For 20 random runs, one algorithm should win at least 15 times under a confidence level of 0.05 to reject a null hypothesis, i.e., statistically outperforming the other one [58]. Our test results show that neither ADAIN.MLP nor ADAIN.SVR can win over one another through the signed test. We also use the Hotelling's T-square statistic test, abbreviated as "t-test," to measure the statistical significance of the prediction accuracy between ADAIN.MLP and other approaches. Table IV shows the t-test result with confidence level of 0.05 (i.e., $|Z|$ should be larger than or equal to 1.96 to reject the null hypothesis). From Table IV, one can find that ADAIN.MLP can statistically outperform IMORL and the data accumulation strategy for most of the data sets. On the other hand, Learn^{++} is statistically the same with ADAIN.MLP in terms of the prediction accuracy for three out of the four data sets.

To have a more comprehensive analysis of the performance, we also employ receiver operating characteristics (ROC) curve [59] to demonstrate the effectiveness of the proposed ADAIN framework. The data accumulation strategy is excluded in this case, since a single CART classifier only outputs hard prediction results, i.e., corresponding to just one point in ROC space. The area under ROC curve (AUC) assessments are shown in Table V to quantify the comparison among ADAIN.MLP, ADAIN.SVR, IMORL, and Learn^{++} in experiment. Note here in lieu of simply averaging the AUCs across the 20 runs, the averaged AUCs are derived based on the *vertical averaging* technique as suggested in [59]. For data sets with more than two classes, their averaged AUCs are calculated by summing the averaged AUC of the reference ROC curves weighted by the class ratio [60]. Specifically, our implementation of the vertical averaging method is illustrated in Fig. 5. Assume one would like to average two ROC curves: l_1 and l_2, each of which is formed by a series of points in the ROC space. The first step is to evenly divide the range of fp_rate into a set of intervals. Then at each interval, find the corresponding tp_rate values of each ROC curve and average them. In Fig. 5, X_1 and Y_1 are the points from l_1 and l_2 corresponding to the interval fp_rate1. By averaging their tp_rate values, the corresponding ROC point Z_1 on the averaged ROC curve is obtained. However, there exist some ROC curves which do not

TABLE V
AVERAGED AUC

Data set	Area Under ROC Curve			
	ADAIN.MLP	ADAIN.SVR	IMORL	Learn^{++}
Spambase	0.9566	0.9576	0.9511	**0.9592**
Magic	0.9090	**0.9198**	0.8896	0.9101
Waveform	0.9456	0.942	0.9262	**0.961**
Sat	0.9716	0.9734	0.9665	**0.9761**

have corresponding points on certain intervals. In this case, one can use the linear interpolation method to obtain the averaged ROC points. For instance, in Fig. 5, the point $\bar{X}$ (corresponding to fp_rate2) is calculated based on the linear interpolation of the two neighboring points X_2 and X_3. Once $\bar{X}$ is obtained, it can be averaged with Y_2 to get the corresponding Z_2 point on the averaged ROC curve.

The online version boosting algorithm [15] (abbreviated as "BoostOnline" in the rest of this paper) is an important technique to address incremental learning problems using the idea of AdaBoost. However, the difference between BoostOnline and ADAIN is two-fold. First, BoostOnline maintains a fixed number of classifiers throughout its learning life. Whenever there is a *single* training data point available, *all* classifiers would be applied to learn from it. On the other hand, a new classifier would be created for ADAIN so long as there is a *chunk* of new training data received. Second, BoostOnline requires the base learning algorithm itself to be able to learn incrementally, while ADAIN does not place such restriction on base learning algorithms. This provides the flexibility of adopting many of the standard off-the-shelf base learning algorithms into the ADAIN framework.

Here we adopt neural network MLP with 100 training epochs as the base classifier for both BoostOnline and ADAIN.MLP to facilitate the direct comparison between them. The number of classifiers for BoostOnline is set to be 20, which is consistent with the number of created classifiers after the training stage for ADAIN. We also apply the nonlinear normalization method [61], [62] to normalize the original feature set of all data sets to facilitate training MLP classifier. In order to compare the proposed ADAIN framework with the off-the-shelf implementation of online learning methods, we also apply *Vowpal Wabbit* (VW) platform [21] to learn from the data sets.

Table VI shows the average overall prediction accuracy and AUC of ADAIN.MLP and BoostOnline on all data sets for simulation across 20 random runs. Since it requires some tweaks on class label to facilitate multiclass classification on VW, in this paper we only present the results of "spambase" and "magic" data sets for VW. One can see that ADAIN.MLP outperforms BoostOnline for all data sets. Regarding comparison with VW, ADAIN.MLP performs generally worse than VW on learning from spambase data set [although ADAIN.MLP achieves higher OA than VW (0.9302 versus 0.9212), its AUC is worse than VW (0.8777 versus 0.9736)]. However, it outperforms VW on learning from magic data set in terms of both OA and AUC. Since magic data set has the largest size across all data sets for simulation, we consider

TABLE VI
OVERALL PREDICTION ACCURACY AND AUC FOR ADAIN.MLP AND BOOSTONLINE

Data set	ADAIN.MLP		BoostOnline		VW	
	OA	*AUC*	*OA*	*AUC*	*OA*	*AUC*
Spambase	**0.9302**	0.8777	0.9003	0.8709	0.9212	**0.9736**
Magic	**0.8553**	**0.9100**	0.7448	0.8056	0.8223	0.8649
Waveform	**0.8530**	**0.9649**	0.7393	0.8949	–	–
Sat	**0.8484**	**0.9781**	0.7211	0.9393	–	–

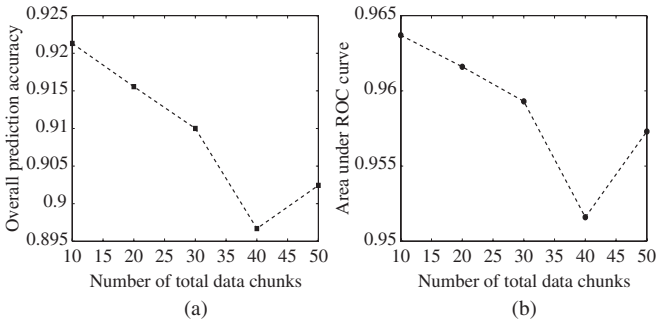

Fig. 6. Simulation results for different number of *total* data chunks for training. (a) Overall prediction accuracy. (b) Area under ROC curve.

ADAIN.MLP still remains competitive against VW in this case.

In order to study the effect of different chunk sizes, here we divide the "spamabase" data set into 10, 20, 30, 40, and 50 chunks, and then apply ADAIN.MLP to learn from these different scenarios. The corresponding prediction accuracy and AUC averaged across 20 random runs are plotted in Fig. 6(a) and (b), respectively. From these results one can see, initially ADAIN's performance is getting worse with the increase of the number of chunks. This could be loosely based on the fact that with the increase of number of chunks, the available data instances in each chunk will be reduced, therefore degrading the classifier's learning and generalization performance. However, when the total number of chunks is set to 50, we notice that the ADAIN can improve its performance in this case. We consider this is probably related to the fact that although the number of data instances in each chunk is reduced, the number of chunks is increased therefore providing a strong opportunity for the proposed approach to catch up and learn from such a relatively large number of chunks. These results suggest there might be a balancing-point with respect to the number of chunks and number of data instances in each chunk to achieve the optimal learning performance.

VI. NEW CONCEPT LEARNING FOR ADAIN

As we have discussed in Section I, incremental learning should be adapted to new concepts, which may be presented in the middle of the learning period. In this section, we would like to focus on demonstrating our proposed framework's capability to handle new concepts. To this end, we dedicate the "waveform" data set to incremental learning with new concepts. To do this, all examples in "waveform" data set are randomly and evenly divided into 40 data chunks, except those belonging to class 3, which are not included into the

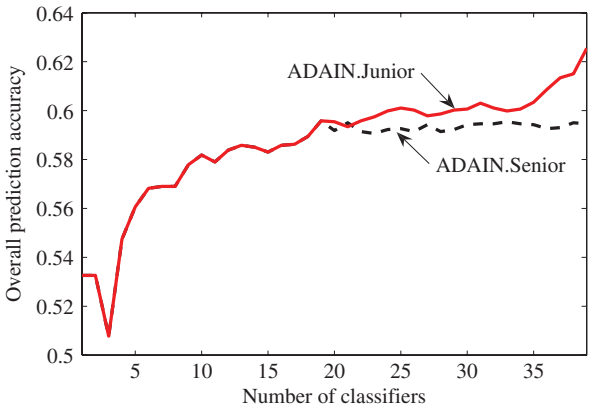

Fig. 7. New concept learning for "waveform" data set.

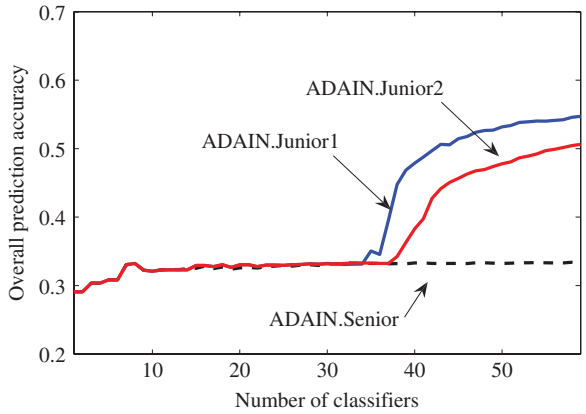

Fig. 8. New concept learning for "sat" data set.

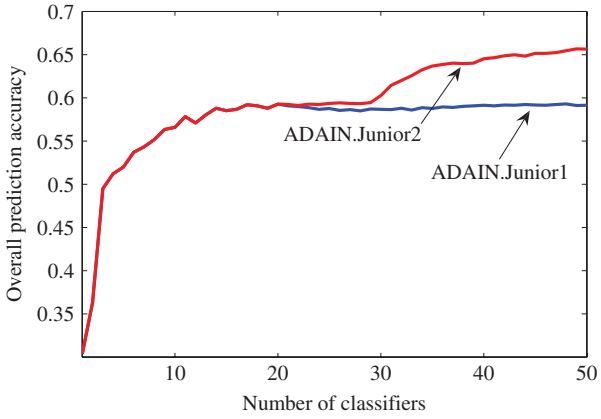

Fig. 9. Pruning hypotheses before the 1st introduction of new concepts.

incremental learning until the 21st data chunk. The base learner and the mapping function are CART and MLP with the same configuration described in Section V.

Fig. 7 shows the overall accuracy tendency of ADAIN for the described learning scenario. The dashed line is for ADAIN sticking to the old concepts during the learning life. We name it "ADAIN.Senior" since this kind of learning is obsessive to the obsolete concepts and the past experience. Its prediction accuracy is understandably poor since its classification on all testing data of class 3 would be wrong. The solid line represents the performance of ADAIN for new concept learning, from which one can see ADAIN quickly makes a difference when the new concept is introduced and shows a significantly improved learning ability till the end of the learning process. This kind of learning can quickly keep updated to the latest emerging new concepts, thus we call it "ADAIN.Junior."

We also would like to explore the new concept learning ability of ADAIN on multiple new concepts. We select the "sat" data set with six classes as an example here. We evenly spread the data of class 1 and class 2 into 60 data chunks. Data belonging to class 3 and class 4 will be introduced since the 11^{th} data chunk, and data from class 5 and class 6 will be introduced since the 31st data chunk. In this case, there are three kinds of ADAIN carried out. The first one, named "ADAIN.Senior," will only learn class 1 and class 2 throughout the entire learning life. The second one, named "ADAIN.Junior1," will face the new concepts of class 3 and class 4 from the 11^{th} data chunk. Finally, the last one, named "ADAIN.Junior2," will face the new concepts of class 3 and class 4 from the 11^{th} data chunk as well as the new concepts of class 5 and class 6 from the 31st data chunk. The learning results are illustrated in Fig. 8. It is apparently astonishing to find out that although ADAIN.Junior2 can still significantly outperform ADAIN.Senior, its performance is worse than ADAIN.Junior1.

This is an interesting observation and we consider the reason might be related to the "aging" mechanism during the long term incremental learning. When a long sequence of data chunks are presented over time, the learning system may need to prune the obsolete hypothesis to better keep tuned on the evolving data stream. Retaining all hypotheses in such scenario not only means excessively occupied memory space but also being over obsessive to the obsolete concepts. By following this philosophy, for ADAIN.Junior1 and ADAIN.Junior2, we abandon the hypothesis created before the class 3 and class 4 are introduced (before the 11^{th} data chunk), and re-test the over-all prediction accuracy tendency in Fig. 9. One can clearly see that ADAIN.Junior2 now can outperform ADAIN.Junior1. From this discussion, we may conclude that by appropriate pruning obsolete hypotheses, ADAIN can significantly be improved for prediction performance particularly for evolving data stream.

The other benefit of pruning obsolete hypotheses is the ecominization of memory space to store hypotheses and the reduction of time for classifying query instances. This is especially important when the proposed ADAIN has to be scaled up to process data stream with large number of data chunks. One cannot keep all hypotheses in this case, and a decision must be made in terms of what is most useful. The Forgetron learning algorithm proposed in [63] is an important effort to explore using fixed budget of memory for online learning from data stream, which is achieved by gradually forgetting active examples. When there is no new class concept presented in the middle of data stream, another method of pruning hypotheses is to maintain a priority queue Q of a fixed capacity H, and follow the procedure specified by Algorithm 2 when a new hypothesis h_n is created.

Fig. 10 illustrates the results of learning from "magic" data set divided into 500 data chunks after applying this pruning

Algorithm 2 Procedure of pruning hypothesis

Require:
priority queue Q of capacity H
a new hypothesis h_n with weight $\log \frac{1}{\beta_n}$
Ensure:
if $(|Q| < H)$ **then**
$Q.insert\_with\_prioity(h_n, \log \frac{1}{\beta_n})$
else
$(h_k, w_k) = Q.pull\_lowest\_priority\_element()$
$\{w_k$ is the associated priority/weight with hypothesis $h_k\}$
if $\log \frac{1}{\beta_n} > w_k$ **then**
$Q.insert\_with\_prioity(h_n, \log \frac{1}{\beta_n})$
else
$Q.insert\_with\_prioity(h_k, w_k)$
end if
end if

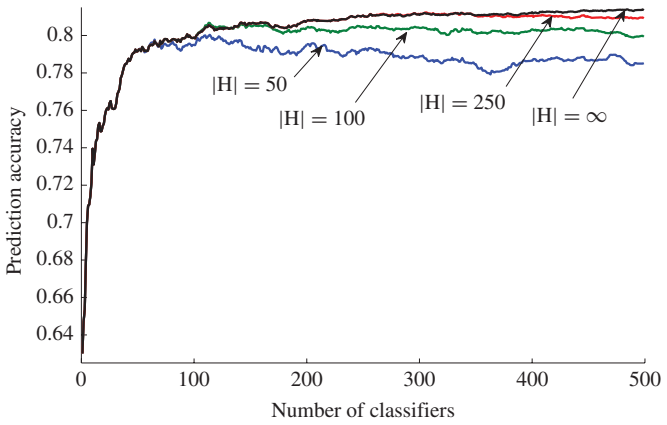

Fig. 10. Performance of ADAIN on "magic" data set using Q of different H.

method when H is set to be 50, 100, 250, and ∞, respectively. Note $H = \infty$ means keeping *all* hypotheses across the learning life. It can be concluded that using a priority queue Q of small capacity ($H = 50$ and $H = 100$) would result in significantly deteriorated performance. However when capacity of Q is set to be large enough ($H = 250$), the performance of ADAIN can approximate that of retaining all hypotheses in learning life. One might need to investigate to find such a *threshold* to achieve best speed-performance trade-off.

VII. Conclusion

In this paper, we proposed an adaptive incremental learning framework for stream data. Based on the adaptive learning and ensemble learning methodology, the proposed ADAIN framework can automatically learn from data stream, accumulating experience over time, and use such knowledge to facilitate future learning and decision-making processes. Simulation results on various data sets and their corresponding statistical tests show the effectiveness of the proposed framework. We would like to note that it is not our intention to compete for the best classification accuracy across all the data sets of the proposed approach with those of existing methods. Instead, our focus in this paper is to investigate the important underlying question of incremental learning for stream data. How to effective integrate previously learned knowledge into currently received data to improve learning from new raw data, and meanwhile accumulate such experience over time to support future decision-making processes. Motivated by the results presented in this paper, we believe that the proposed ADAIN framework can achieve this goal.

As a new learning framework, there are a number of interesting topics along this direction for future research. For instance, it is widely recognized that concept drifting/shifting is a critical issue for incremental learning with stream data. While in our current paper we have had a light touch on this problem, a more in-depth study of this issue from both analytical and empirical analysis point of view will be important to fully understand the behavior of the proposed approach when facing such challenges. For instance, how does the proposed approach handle the trade-off between the outdated experience and new knowledge for extremely long-sequence stream data? How fast can this approach adjust its decision boundary to the new concepts? One possible way to deal with this is to introduce a mechanism with more complicated decay factor and/or momentum term on this. In our current paper, we briefly touched this issue in Section VI and will continue to investigate this problem in a more systematic and principled way. Furthermore, in this paper, we only consider classification problems. It would be interesting to extend this framework to regression learning problems for different real applications. Finally, large-scale empirical study of this approach on different real-world data sets under different configurations is necessary to fully demonstrate the effectiveness of this approach across different application domains. For instance, as we discussed in Section V, the chunk size will have an important impact on the final learning performance. This presents useful considerations in practical applications such as data acquisition, data buffer size, and sampling frequency to achieve the desired learning performance under different resource limitations. Also, as we pointed out in Section II, how to handle the data streams with imbalanced class distributions, a common phenomenon in many data-intensive applications, has been a critical challenge in the community (see the survey paper of [45] for details). It will be important to analyze the performance and potential improvements for the proposed approach to handle such a challenging issue to address real-world needs. We are currently investigating all these issues and their results will be reported in future research. Motivated by our results in this paper, we believe the proposed approach will not only provide important insight into the fundamental problem of incremental learning with stream data, but it can also provide useful techniques and solutions for different real-world stream data analysis applications.

References

[1] H. Zhao and P. C. Yuen, "Incremental linear discriminant analysis for face recognition," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 38, no. 1, pp. 210–221, Feb. 2008.
[2] J. R. Millan, "Rapid, safe, and incremental learning of navigation strategies," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 26, no. 3, pp. 408–420, Jun. 1996.

[3] G. Y. Chen and W. H. Tsai, "An incremental-learning-by-navigation approach to vision-based autonomous land vehicle guidance in indoor environments using vertical line information and multiweighted generalized Hough transform technique," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 28, no. 5, pp. 740–748, Oct. 1998.

[4] R. B. Segal and J. O. Kephart, "Incremental learning in swiftfile," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 863–870.

[5] G. G. Yen and P. Meesad, "An effective neuro-fuzzy paradigm for machinery condition health monitoring," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 31, no. 4, pp. 523–536, Aug. 2001.

[6] J. Su, J. Wang, and Y. Xi, "Incremental learning with balanced update on receptive fields for multi-sensor data fusion," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 34, no. 1, pp. 659–665, Feb. 2004.

[7] S. U. Guan and F. Zhu, "An incremental approach to genetic-algorithms-based classification," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 35, no. 2, pp. 227–239, Apr. 2005.

[8] Y. Cao, H. He, and H. Huang, "Lift: A new framework of learning from testing data for face recognition," *Neurocomputing*, vol. 74, no. 6, pp. 916–929, 2011.

[9] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zollner, "Incremental learning of tasks from user demonstrations, past experiences, and vocal comments," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 37, no. 2, pp. 322–332, Apr. 2007.

[10] A. Sharma, "A note on batch and incremental learnability," *J. Comput. Syst. Sci.*, vol. 56, no. 3, pp. 272–276, Jun. 1998.

[11] S. Lange and G. Grieser, "On the power of incremental learning," *Theor. Comput. Sci.*, vol. 288, no. 2, pp. 277–307, Oct. 2002.

[12] Z.-H. Zhou and Z.-Q. Chen, "Abstract hybrid decision tree," *Knowl.-Based Syst.*, vol. 15, no. 8, pp. 515–528, 2002.

[13] H. He and S. Chen, "IMORL: Incremental multiple-object recognition and localization," *IEEE Trans. Neural Netw.*, vol. 19, no. 10, pp. 1727–1738, Oct. 2008.

[14] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.

[15] N. Oza and S. Russell, "Online bagging and boosting," in *Proc. Artif. Intell. Stat.*, vol. 3. Oct. 2006, pp. 105–112.

[16] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 260–267.

[17] K. Crammar, O. Dekel, J. Keshet, S. Shalev-Shawartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Mar. 2006.

[18] G. Gasso, A. Pappaioannou, M. Spivak, and L. Bottou, "Batch and online learning algorithms for nonconvex Neyman-Pearson classification," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 28–47, Apr. 2011.

[19] S. Ertekin, L. Bottou, and C. L. Giles, "Nonconvex online support vector machines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 368–381, Feb. 2011.

[20] Y. Singer and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for SVM," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 807–814.

[21] *Vowpal Wabbit Project* [Online]. Available: http://github.com/John-Langford/vowpal_wabbit/wiki

[22] H. He and J. A. Starzyk, "Online dynamic value system for machine learning," in *Proc. 4th Int. Symp. Neural Netw.: Adv. Neural Netw.*, vol. 4491. 2007, pp. 441–448.

[23] H. He and J. A. Starzyk, "A self-organizing learning array system for power quality classification based on wavelet transform," *IEEE Trans. Power Delivery*, vol. 21, no. 1, pp. 286–295, Jan. 2006.

[24] J. A. Starzyk, Z. Zhu, and H. He, "Self-organizing learning array and its application to economic and financial problems," in *Proc. Joint Conf. Inf. Syst.*, Cary, NC, 2003, pp. 1–3.

[25] Z. Zhu, H. He, J. A. Statzyk, and C. Tseng, "Self-organizing learning array and its application to economic and financial problems," *Inf. Sci.: Int. J.*, vol. 177, no. 5, pp. 1180–1192, Mar. 2007.

[26] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*. Cambridge, MA: MIT Press, 1995, pp. 625–632.

[27] B. Fritzke, "Incremental learning of local linear mappings," in *Proc. Int. Conf. Artif. Neural Netw.*, 1995, pp. 217–222.

[28] T. M. Martinetz and K. J. Schulten, *Artificial Neural Networks*. Amsterdam, The Netherlands: North-Holland, 1991, pp. 397–402.

[29] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.

[30] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Netw.*, vol. 9, no. 5, pp. 881–897, Jul. 1996.

[31] C. P. Lim and R. F. Harrison, "An incremental adaptive network for on-line supervised learning and probability estimation," *Neural Netw.*, vol. 10, no. 5, pp. 925–939, Jul. 1997.

[32] F. H. Hamker, "Life-long learning cell structures–continuously learning without catastrophic interference," *Neural Netw.*, vol. 14, nos. 4–5, pp. 551–573, May 2001.

[33] G. C. Anagnostopoulos and M. Georgiopoulos, "Ellipsoid ART and ARTMAP for incremental clustering and classification," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2. Washington D.C., Jul. 2001, pp. 1221–1226.

[34] S. Salzberg, "A nearest hyperrectangle learning method," *Mach. Learn.*, vol. 6, no. 3, pp. 277–309, 1991.

[35] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 769–783, May 2000.

[36] A. Bouchachia, "Incremental learning via function decomposition," in *Proc. Int. Conf. Mach. Learn. Appl.*, 2006, pp. 63–68.

[37] A. Bouchachia, B. Gabrys, and Z. Sahel, "Overview of some incremental learning algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, London, U.K., Jul. 2007, pp. 1–6.

[38] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Netw.*, vol. 22, no. 3, pp. 200–212, Apr. 2009.

[39] H. G. Zhang, Y. H. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1490–1503, Sep. 2009.

[40] F. Y. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 24–36, Jan. 2011.

[41] D. Liu, Y. Zhang, and H. G. Zhang, "A self-learning call admission control scheme for CDMA cellular networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1219–1228, Sep. 2005.

[42] J. Si and Y. T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[43] J. Fu, H. He, and X. Zhou, "Adaptive learning and control for MIMO system based on adaptive dynamic programming," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1133–1148, Jul. 2011.

[44] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, 2011, to be published.

[45] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[46] S. Chen, H. He, and E. A. Garcia, "RAMOBoost: Ranked minority oversampling in boosting," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1624–1642, Oct. 2010.

[47] S. Chen and H. He, "Toward incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach," *Evolv. Syst.*, vol. 2, no. 1, pp. 35–50, 2011.

[48] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[49] Y. Freund and R. Schapire, "Decision-theoretic generalization of on-line learning and application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[50] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[51] P. J. Werbos, "Backpropagation: Basics and new developments," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 134–39.

[52] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst. 9*, 1996, pp. 155–161.

[53] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. New York: Chapman & Hall, 1984.

[54] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2003.

[55] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting streams with skewed distribution," in *Proc. SIAM Int. Conf. Data Min.*, Minneapolis, MN, 2007, pp. 3–14.

[56] A. Asuncion and D. J. Newman. (2007). *UCI Machine Learning Repository* [Online]. Available: http://www.ics.uci.edu/∼mlearn/ MLRepository.html
[57] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.
[58] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. New York: Chapman & Hall, 2000.
[59] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," HP Laboratories, Palo Alto, CA, Tech. Rep. HPL-2003-4, Mar. 2004.
[60] F. Provost and P. Domingos, "Well-trained pets: Improving probability estimation trees," in *Proc. CeDER Working Pap.*, 2001, no. IS-00-04, pp. 1–25.
[61] H. He and X. Shan, "A ranked subspace learning method for gene expression data classification," in *Proc. Int. Conf. Artif. Intell.*, 2007, pp. 358–364.
[62] H. He, *Self-Adaptive Systems for Machine Intelligence*. New York: Wiley, 2011.
[63] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The Forgetron: A kernel-based perceptron on a budget," *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, 2008.

Haibo He (SM'11) received the B.S. and M.S. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in electrical engineering from Ohio University, Athens, in 2006.

He is currently an Assistant Professor with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston. From 2006 to 2009, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ. He has published one research book (Wiley), edited six conference proceedings (Springer), and authored or co-authored over 80 peer-reviewed journal and conference papers. His researches have been covered by national and international media such as the IEEE Smart Grid Newsletter, the Wall Street Journal, and Providence Business News. His current research interests include adaptive dynamic programming, machine learning, computational intelligence, very large scale integration and field-programmable gate array design, and various applications such as smart grid.

Dr. He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE TRANSACTIONS ON SMART GRID. He received the National Science Foundation CAREER Award in 2011 and the Providence Business News Rising Star Innovator in 2011.

Sheng Chen (S'06) received the B.S. and M.S. degrees in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer engineering from the Stevens Institute of Technology, Hoboken, NJ, in 2011.

His current research interests include computational intelligence, statistical machine learning, and pattern recognition, as well as their applications such as record matching in database and computational linguistics.

Kang Li (M'05) received the B.Sc. degree from Xiangtan University, Hunan, China, in 1989, the M.Sc. degree from the Harbin Institute of Technology, Harbin, China, in 1992, and the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 1995.

He is currently a Reader with the Intelligent Systems and Control group, Queens University Belfast, Belfast, U.K., and the current Secretary of the IEEE U.K. and the Republic of Ireland Section. He has published over 160 papers in his areas of expertise. His current research interests include nonlinear system modeling, identification and controls, bio-inspired computational intelligence, fault-diagnosis and detection, with recent applications on energy efficiency and pollution reduction techniques in power generation plants and power systems, and polymer extrusion processes. He is also interested in bioinformatics with applications on food safety, healthcare and biomedical engineering.

Xin Xu (M'07) received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, China, in 1996, and the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation (CMA), NUDT, in 2002.

He has been a Visiting Scientist for cooperation research with the Hong Kong Polytechnic University, University of Alberta, Edmonton, AB, Canada, the University of Guelph, Guelph, ON, Canada, and the University of Strathclyde, Glasgow, U.K., respectively. He is currently a Full Professor with the Institute of Automation, CMA. He has co-authored four books and published more than 70 papers in international journals and conferences. His current research interests include reinforcement learning, learning controls, robotics, data mining, autonomic computing, and computer security.

Dr. Xu is one of the recipients of the first class Natural Science Award from Hunan, China, in 2009, and the Fork Ying Tong Youth Teacher Fund of China in 2008. He is a Committee Member of the IEEE Technical Committee on Approximate Dynamic Programming and Reinforcement Learning and the IEEE Technical Committee on Robot Learning. He has served as a PC member or Session Chair in numerous international conferences.