Paper:

An Online Incremental Semi-Supervised Learning Method

Furao Shen*, Hui Yu*, Youki Kamiya, and Osamu Hasegawa****

*The State Key Laboratory for Novel Software Technology, and
Jiangyin Information Technology Research Institute, Nanjing University
Nanjing 210093, P.R. China
E-mail: frshen@nju.edu.cn
**The Imaging Science and Engineering Lab., Tokyo Institute of Technology
R2-52, 4259 Nagatsuda, Midori-ku, Yokohama 226-8503, Japan
E-mail: hasegawa@isl.titech.ac.jp
[Received December 21, 2009; accepted May 28, 2010]

Using labeled data and large amounts of unlabeled data, our proposed online incremental semi-supervised learning automatically learns the topology of input data distribution without prior knowledge of numbers of nodes or network structure. Using labeled data, it labels generated nodes and divides a learned topology into substructures corresponding to classes. Node weights used as prototype vectors enable classification. New labeled or unlabeled data is added incrementally to the system during learning. Experimental results for artificial and real-world data show that this learning efficiently learns online incremental tasks even in noisy and non-stationary environments.

Keywords: online incremental learning, labeled data and unlabeled data, semi-supervised learning, non-stationary environment

1. Introduction

Using huge datasets in designing efficient and robust learning algorithms involves continually adding new datasets to already huge databases and introduces two issues. First is the so-called Stability-Plasticity Dilemma [1] – how to learn new knowledge while retaining previous knowledge – is an important target of online incremental learning algorithms [2, 3]. Second is how to learn when labeled instances are difficult, expensive, or time-consuming to obtain because they require experienced human personnel. This is discussed mainly related to semi-supervised learning [4, 5], addressed using large amounts of unlabeled data, together with labeled data, to build better classifiers. Requiring less human effort and yielding higher accuracy, semi-supervised learning is of interest in both theory and practice [6, 7]. However, the former issue has not been considered as sufficient among many types of traditional semi-supervised learning.

Competitive online incremental neural-networks-based learning algorithms using supervised or unsupervised learning have the advantage of operating incrementally with new data information, e.g., Kohonen's Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ) [8], its modification for supervised learning. With SOM and LVQ unsuitable, however, for incremental learning, the number of nodes is predefined. Self-growing SOM [9] and Growing Cell Structures (GCS) [10] improve performance by inserting nodes and gradual increasing structural complexity. Growing Neural Gas (GNG) architecture [11], is a GCS modification, does not predefine the topological structure dimensionality, but is, instead, discovered during training. Self-growing SOM, GCS, and GNG, do not predefine the number of nodes, and learn new information by inserting nodes. They do, however, require that the maximum node number be predefined to prevent interminable node insertion continues as long as new input patterns arise. Once this maximum is reached further input engenders the possibility of destruction. Further proposed modifications such as Life-long Learning Cell Structures (LLCS) [2] for supervised classification and Self-Organizing Incremental Neural Networks (SOINN) [3] for unsupervised clustering automatically stop and restart node insertion, avoiding interminable node increase, balancing stability and plasticity [2, 3].

Incremental Growing Neural Gas (IGNG) [12] realizes semi-supervised clustering, alternating between clustering part of huge datasets and having users correct the network. Human operation thus remains complex, including finding clustering mistakes, correcting the network by edge insertion and deletion, and assigning labels to disjoint subgraphs. Semi-supervised Fuzzy ARTMAP (ssFAM) and semi-supervised Ellipsoidal ARTMAP (ssEAM) are rooted in Adaptive Resonance Theory (ART) [13] as recent extensions of the basic Fuzzy ARTMAP (FAM) architecture [14]. Although they prevent overfitting to training data in achieving zero posttraining error, they require large numbers of labeled data for training [15].

The online incremental semi-supervised learning we propose targets four goals:

1. Finding suitable numbers of nodes automatically and representing the topology structure of large-scale

training data without a priori knowledge such as maximum node number or given network structure.

2. Constructing classifiers based on small amounts of labeled data and large amounts of unlabeled data.
3. Realizing incremental learning by learning new information without destroying old information.
4. Automatically and continuously fine-tuning old learned classes based on new labeled and unlabeled data.

In this paper, Section 2 introduces our proposed semi-supervised online incremental learning algorithm. Section 3 explores experiments using an artificial dataset and real datasets to verify the feasibility of our algorithm and compare our proposal to conventional methods. Section 4 summarizes findings and conclusions.

2. Proposal Method

As staled in Section 1, we are targeting among other things learning from labeled and unlabeled training data and incrementally inputting new training data. To do so, we have separated our proposal into three parts:

1. Topology learning – representing the topology structure of all labeled and unlabeled training data.
2. Learned nodes labeling – labeling all learned nodes using labeled training data.
3. Classifier construction – constructing classifiers using labeled nodes.

Our architecture has three layers – input, competitive, and output – as shown in **Fig. 1**. In the input layer, labeled and unlabeled data are mixed to form training data input to the competitive layer and competitive are happened between nodes in the competitive layer, such nodes and their connection represent the topology structure for inputting training data. Labeled data is used to label competitive layer nodes, and display results in the output layer, which is structured the same as the competitive layer but has labeled nodes. Labeled weight vectors nodes are used as prototypes to build classifiers.

Steps 1–10 of **Fig. 2** learn the input data topology structure, such data are input incrementally to the system. Steps 11–14 use labeled input data to label learned nodes from Steps 1–10. Weights of these nodes are used as different class prototypes to build classifiers. Step 15 decides whether learning is finished, and if so, outputs all class prototypes. If not,it continues online incremental learning.

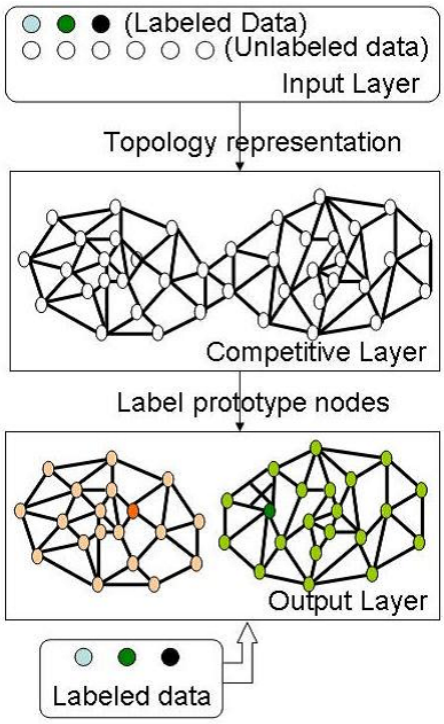


Fig. 1. Architecture of the proposed method.

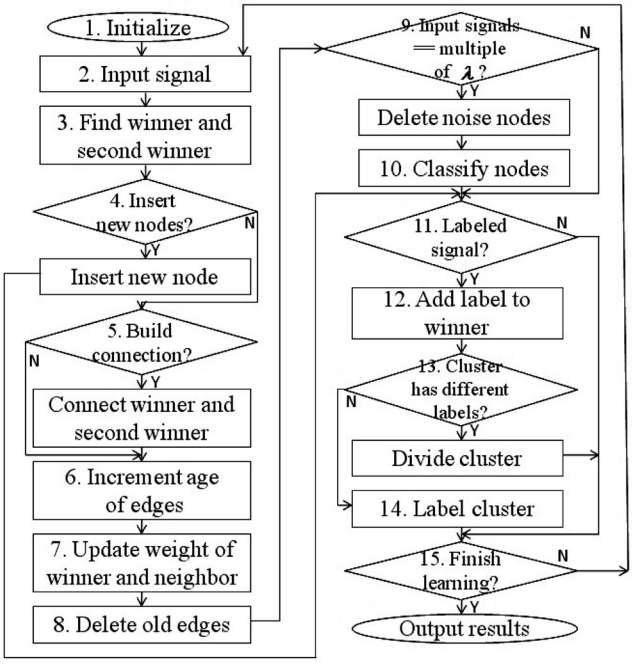


Fig. 2. Proposed learning.

2.1. Topology Learning

To realize the targets stated in Section 1, competitive networks such as SOM can learn the input data topology structure, and some related incremental learning is proposed. SOINN, an incremental competitive learning neural network, learns input data topology incrementally, finds suitable numbers of nodes automatically, and reports the numbers of classes. SOINN's two-layer network enables unsupervised competitive learning. With enhanced-SOINN [16], single-layer SOINN is sufficient for topology structure learning, as detailed in the sections that follow.

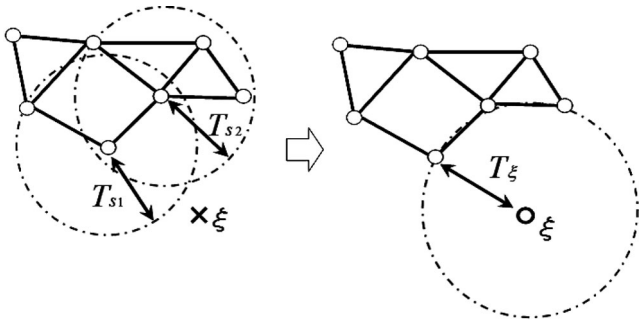


Fig. 3. New node insertion: if at least one of two Euclidean distances between input pattern ξ and either *winner* s_1 or *runner up* s_2 is larger than similarity threshold T_{s_1} or T_{s_2} (left), the input pattern is inserted as a new node (right).

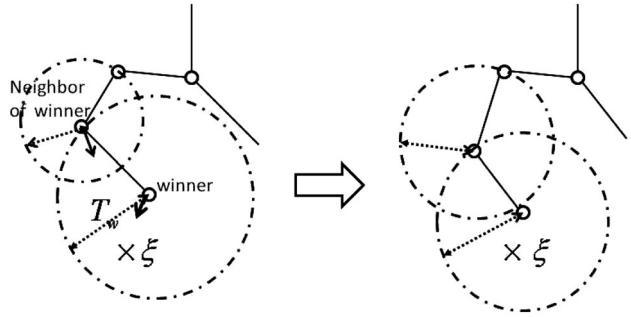


Fig. 4. Node update weights: if Euclidean distances between input pattern ξ and either *winner* s_1 or *runner up* s_2 is smaller than similarity threshold T_{s_1} or T_{s_2} (left), weights of *winner* s_1 and its neighbor are moved toward input pattern ξ (right).

Single-layer SOINN's insertion schemes shown in **Fig. 3**, inserts new nodes to enable incremental learning and topology representation. To insert new nodes, SOINN uses threshold distance T, corresponding nodes in the existing network, to judge whether an input sample and its nearest node belong to the same cluster. If so, the sample is inserted as a new node.

If the distance between input pattern ξ and *winner* s_1 or *runner up* s_2 is less than similarity threshold T_{s_1} or T_{s_2}, weights of s_1 and its neighbors are moved toward ξ, as shown in **Fig. 4**.

Similarity threshold T_i is defined as the Euclidean distance from the boundary to the center of Voronoi region V_i of node i. During learning, node i move to meet input pattern distribution, so when a new pattern is input and node i is the winner, Voronoi region V_i of node i changes, similarity threshold T_i also changes. When new node i is generated, similarity threshold T_i of node i is initialized to $+\infty$. When node i becomes a *winner* – the node nearest the input pattern – or *runner up* – the second nearest – similarity threshold T_i is updated. If node i has a direct topological neighbor – a node linked to node i with an edge – T_i is updated as the maximum distance between

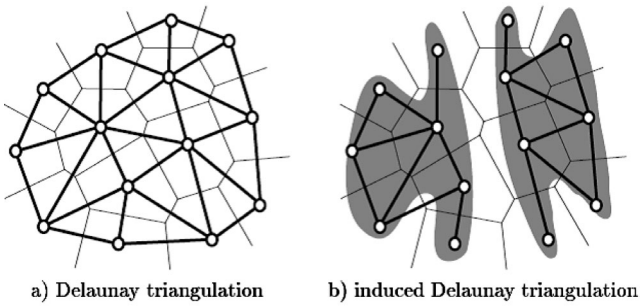


Fig. 5. Two ways to define closeness among a set of points. a) *Delaunay triangulation* (thick lines) connects points having neighboring *Voronoi polygons* (thin lines). b) *Induced Delaunay triangulation* (thick lines) is obtained by masking original Delaunay triangulation with data distribution P(ξ) (shaded) [18].

node i and all of its neighbors:

$$T_i = \max_{c \in N_i} \|\mathbf{W}_i - \mathbf{W}_c\|. \quad \ldots\ldots\ldots\ldots (1)$$

$\mathbf{W}_i$ is the node i weight vector, N_i the node i neighbor set, and $\|\mathbf{W}_i - \mathbf{W}_c\|$ the Euclidean distance between vectors $\mathbf{W}_i$ and $\mathbf{W}_c$. If node i has no neighbor, T_i is updated as the minimum distance between node i and all other nodes in the network A:

$$T_i = \min_{c \in A \setminus \{i\}} \|\mathbf{W}_i - \mathbf{W}_c\|. \quad \ldots\ldots\ldots\ldots (2)$$

A is the node set.

To build connections among neural nodes, SOINN uses the competitive Hebbian rule [17], for input signals, connecting the two closest nodes – the *winner* and *runner up* measured using the Euclidean distance – with an edge to forms a network whose edges are in the area suggested by input data distribution (**Fig. 5(b)**). The network is a subgraph of original Delaunay triangulation (**Fig. 5(a)**). Based on the competitive Hebbian rule, the resulting graph approximates input data distribution shape [17]. During online learning, nodes change locations slowly but permanently, so neighbors early on may not be neighbors later. The competitive Hebbian rules *edge aging* removes connections not recently refreshed [18]. SOINN removes edges whose age exceeds threshold parameter a_d.

To delete noise-generated nodes, if the number of input signals generated is an integer multiple of parameter λ, SOINN removes nodes that having only one or no topological neighbor assuming, that such a node is highly likely noise-generated.

Two nodes linked by a series of edges are said to have a path between them [3], i.e., given series of nodes $x_i \in A$, $i = 1,2,\ldots,n$, makes (i,x_1), (x_1,x_2),..., (x_{n-1},x_n), $(x_n, j) \in C$, so a "path" exists between nodes i and j. A is the node set and C the connection or edge set.

Martinetz and Schulten [18] conclude that the competitive Hebbian rule is suitable for forming a path that preserves representations of a given manifold. Network connectivity corresponds to induced Delaunay triangulation, defining both a perfectly topology-preserving map and

path-preserving representation if the node number is sufficient for dense distribution. With path-preserving representation, the competitive Hebbian rule determines parts of a given pattern manifold separated and forming different clusters. Two nodes linked by one path belong to one cluster. We use the same algorithm as SOINN to classify nodes.

Algorithm 2.1: Classify nodes in different classes:

1. Initialize all nodes as unclassified.
2. Randomly choose one unclassified node i from node set A. Mark node i classified and label it class C_i.
3. Search A to find all unclassified nodes connected to node i by a "path." Mark these nodes classified and label them the same class as node i.
4. Go to Step 2 to continue classification until all nodes are classified, if unclassified nodes exist.

Both single and two-layer SOINN are designed for unsupervised learning. To separate over lapping classes, SOINN defines node density and designs techniques to separate overlapped classes. Because, we target semi-supervised learning, we simplify SOINN using labeled data to label all learned nodes and separate overlapped classes, as detailed in Section 2.2. To summarize, topology structure we follow basic SOINN for online incrementally learning the input data as detailed in Steps 1–10 in **Fig. 2**.

2.2. Labeling Learned Nodes

Steps 11–14 in **Fig. 2** show the basic learned-node labeling flow. If a labeled vector with weight $\mathbf{W}_s$ is input, node i nearest to the vector is found as follows:

$$i = \arg\min_{i \in N} \|\mathbf{W}_i - \mathbf{W}_s\|. \quad \ldots\ldots\ldots \quad (3)$$

Give node i the same label as the input vector. A node labeled directly with an input vector label is called a "teacher node."

Steps 1–10 of **Fig. 2** generate nodes and the topology structure based on input data density. Data belonging to the same class is allocated to the same cluster. We give these nodes the same label if generated nodes belong to the same cluster, so given teacher nodes, we label unlabeled nodes with the same label as the teacher node within the same cluster.

More than one teacher node with a different label belonging to one cluster means an overlap exists between different classes that are linked to form one cluster. To label learned nodes suitably, overlapped classes must be divided using teacher nodes. Density is generally used to define the amount of data in an area, e.g., a high-density area has a large amount of data. Further, if nodes learned by Steps 1–10 in **Fig. 2** lie in a high-density area, nodes will be close together, and vice versa. To divide a cluster with different labeled nodes, we use approximation radius R_i:

$$R_i = \frac{1}{|N_i|} \sum_{c \in N_i} ||\mathbf{W}_i - \mathbf{W}_c||. \quad \ldots\ldots\ldots \quad (4)$$

N_i is the neighbor set of node i, $|N_i|$ the number of elements in N_i, and the approximation radius the mean distance between the node and its neighbors.

Typically, density of boundary area between classes is more likely less than central area of classes. Thus, if R_i of node i is greater than all of its neighbors, the node is located in a low-density area, probably on the boundary between classes. We therefore use Algorithm 2.2 to label all learned nodes and divide overlapped classes:

Algorithm 2.2: Label nodes and separate cluster D with overlapped classes. Note that some teacher nodes with different labels exist in cluster D:

1. Initialize boundary node set B_{bnd} as the empty set.
2. Initialize all nodes in cluster D as unclassified and $k = 0$. Set node set B_k as follows:

   $$B_k = B_0 = \{a | \forall a \in D, L_a \neq \emptyset\}. \quad \ldots\ldots \quad (5)$$

   L_a is the label of a, meaning that B_0 consists of teacher nodes, i.e., nodes having labels.
3. For all nodes in cluster D, calculate the approximation radius using Eq. (4).
4. For each node $u \in B_k$, if its approximation radius R_u is larger than that of all its unclassified neighbors, move it to B_{bnd}.
5. For each node $u \in B_k$, search unclassified neighbors of node u in cluster D. Set the predicted class label of all neighbors to that of node u. If a neighbor node connects multiple nodes whose labels are mutually different, set multiple predicted class labels for this neighbor to that of all relevant nodes.
6. Set all labeled neighbors to node set B_{k+1}.

   $$B_{k+1} = \{a | \forall a \in D, a \in N_u \ (\forall u \in B_k)\} \quad . \quad (6)$$

   N_u is the neighbor node set of node u.
7. Increment k by 1. If B_k is not the empty set, Go to Step 4.
8. If all nodes in D are labeled, delete the predicted class label of all nodes in B_{bnd} and end. If not, $B_k = B_{bnd}$ and go to Step 4.

Algorithm 2.2 initializes boundary node set B_{bnd} of overlapped classes to empty. Labeled node set B_k is then initialized to B_0, and B_0 consists of teacher nodes. Nodes with different labels are mutually different in B_0. The approximation radius is calculated for all nodes. A labeled node in B_k judged to be a boundary node, is moved to boundary node set B_{bnd}. We use then labeled nodes in B_k to label nodes in cluster D and neighbors of labeled

nodes are labeled with the same label as labeled nodes. Nodes neighboring labeled nodes with different labels are labeled with multiple labels. Labeled node set B_{k+1} is updated to the neighbor node set of B_k, going on moving boundary nodes to B_{bnd} and labeling remaining unlabeled nodes in D until all nodes in D are labeled. Nodes in B_{bnd} lying on the boundary are not used for future classification, so their labels are deleted. **Fig. 1**, for example, shows a two-class task with the two classes overlapping, so the two labeled data belong to two classes. The topology structure is learned based on topology representation process, and results are shown on the competitive layer of **Fig. 1**. Using label prototype nodes (Algorithm 2.2), based on the two labeled data, two nodes are determined to be teacher nodes and overlapped classes are divided into two classes using teacher nodes and results shown in the output layer of **Fig. 1**.

2.3. Complete Algorithm of the Proposal

We then summarize the detailed algorithm of our proposal in Algorithm 2.3. Each algorithm step corresponds to the element number shown in **Fig. 2**.

Algorithm 2.3: topology learning then label learned nodes:

1. Initialize node set A to contain two nodes, c_1 and c_2, with weight vectors chosen randomly from the input pattern. Initialize connection (edge) set C, $C \subset A \times A$, to the empty set:

$$C = \emptyset. \quad \ldots\ldots\ldots\ldots\ldots\ldots (7)$$

2. Input new pattern $\xi \in \mathbb{R}^n$.

   Topology representation

3. Search for nearest node – the *winner* – s_1, and second-nearest node – the *runner up* – s_2 as follows:

$$s_1 = \arg \min_{c \in A} \|\xi - \mathbf{W}_c\| \quad \ldots\ldots (8)$$
$$s_2 = \arg \min_{c \in A \setminus \{s_1\}} \|\xi - \mathbf{W}_c\| \quad \ldots\ldots (9)$$

4. If the distance between ξ and s_1 or s_2 is greater than similarity threshold T_{s_1} or T_{s_2}, the input signal is a new node: add it to A and go to Step 13, i.e., if $\|\xi - W_{s_1}\| > T_{s_1}$ or $\|\xi - W_{s_2}\| > T_{s_2}$, then $A = A \cup r$ and $W_r = \xi$. Similarity threshold T is calculated using Eq. (1) or (2).

5. If a connection does not exist between s_1 and s_2, create it.

$$C = C \cup (s_1, s_2) \quad \ldots\ldots\ldots (10)$$

   Set the age of the connection between s_1 and s_2 to zero.

$$age_{(s_1,s_2)} = 0 \quad \ldots\ldots\ldots\ldots (11)$$

6. Increment the age of all edges linking to s_1 by 1.

$$age_{(s_1,i)} = age_{(s_1,i)} + 1 \ \ (\forall i \in N_{s_1}) \quad \ldots (12)$$

7. Adapt the weight vectors of the *winner* and its direct topological neighbors by fraction ε_1 and ε_2 of the total distance to the input signal. Thus:

$$\Delta \mathbf{W}_{s_1} = \varepsilon_1 (\xi - \mathbf{W}_{s_1}) \quad \ldots\ldots\ldots (13)$$
$$\Delta \mathbf{W}_i = \varepsilon_2 (\xi - \mathbf{W}_i) \quad \ldots\ldots\ldots (14)$$

   for all nodes i in N_{s_1}, which is a direct neighbor set of s_1.

8. Remove edges with ages exceeding a predefined threshold a_d, i.e., if $(i, j) \in C$, and $age_{(i,j)} > a_d$ $(\forall i, j \in A)$, then $C = C \setminus \{(i, j)\}$. If this removal creates nodes having no more linked edges, remove such nodes.

9. If the number of input signals generated so far is an integer multiple of parameter λ, delete some nodes as follows: for all nodes in A, search for nodes having only one or no neighbor, then remove them.

10. Classify nodes in different classes using *Algorithm 2.1*

11. If input pattern ξ is unlabeled, go to Step 13.

12. If input pattern ξ is labeled, add label L_ξ to *winner* s_1: $L_{s_1} = L_\xi$. Labeled nodes are called "teacher nodes."

    Labeling learned nodes

13. Use Algorithm 2.2 to label learned nodes:

14. If different labels exist in teacher nodes of one cluster, divide the cluster into different classes. If all teachers within one cluster have the same labels, label all nodes of this cluster with the teacher node label.

15. If learning ends, output labeled prototypes as learning results, otherwise go to Step 2 to continue online semi-supervised learning.

Algorithm 2.3 consists of (1) topology representation and (2) labeling learned nodes. Labeled and unlabeled data is first mixed to learn the input data topology structure, then labeled data is used to find teacher nodes that are then used to label all learned nodes, even if overlap exists between classes.

In Algorithm 2.3, we must determine parameters λ, a_d, ε_1, and ε_2. Parameters λ and a_d influence the frequency of connection deletion between nodes and nodes located in sparse areas. To retain previously learned knowledge, a large value is chosen for these two parameters and many nodes obtained to realize a detailed topology representation. We set the smallest possible values for these two parameters to remove nodes and edges frequently if we

desire fewer nodes to save memory. The two parameters thus depend on the real task condition and we use these two parameters to control our proposal's performance.

ε_1 and ε_2 are learning rates used to modify the node weight vector. We adapt the learning rate over time using $\varepsilon_1 = 1/M_{s_1}$ and $\varepsilon_2 = 1/100M_i$. M_c is the number of times node c is selected as the *winner*. We determined these values from experience, although appropriate values for various tasks may differ.

Most conventional semi-supervised algorithms act in a *transductive setting* where labels of unlabeled examples are estimated by learning a function defined only over point cloud data [19]. Our proposal, however, acts in a *semi-supervised setting* [19] and can predict previously unencountered input patterns. In other words, at any point in learning, we can construct a prototype-based classifier using our proposal's output node weight as prototypes. We use the One-Nearest Neighbor (1-NN) rule [20] to classify unknown patterns. Based Algorithm 2.2, we set multiple predicted class labels on boundary nodes or do not label then. Using only nodes with one predicted class label as prototypes, the classifier appears to operate based on the k-Nearest Neighbor rule [20].

3. Experiments

In addition to experiments using artificially generated datasets and real datasets, we test and explain our proposal using artificial datasets, and using real datasets, we compare the proposed classifier to semi-supervised classifiers and neural-based classifiers.

3.1. Learning Property Verification

We first create an artificial dataset and verify the property of our proposal using the artificial 2-dimensional dataset in **Fig. 6** to take advantage of its intuitive manipulation, visualization, and insight into system behavior. The dataset is generated by sampling from a bivariate mixture of isotropic Gaussian distributions, i.e., from $N(\mu_i, \mathbf{I})$ with equal priors consisting of four components. Means (centers) of these distributions were positioned so that symmetry for horizontal and vertical axes was four-fold. The distance between each two centers, except for diagonal pairs, is 2.8. Two diagonal components correspond to the same class distribution, i.e., this dataset includes two class distribution. We added 10% random noise to each distribution, generated 10,000 patterns for each distribution, and set 10 labeled samples from all 40,000 generated samples. We selected at least one sample from each distribution when choosing labeled samples, so the dataset consists of 10 labeled and 39,990 unlabeled patterns. The labeling selection rate is very low at 0.025%.

Using this dataset, we tested our proposal in two environments – a *stationary* and *non-stationary incremental*. In the stationary environment, each input pattern is randomly chosen from all four distributions. In the non-stationary incremental environment, patterns of the respective distributions become available for learning sequentially.

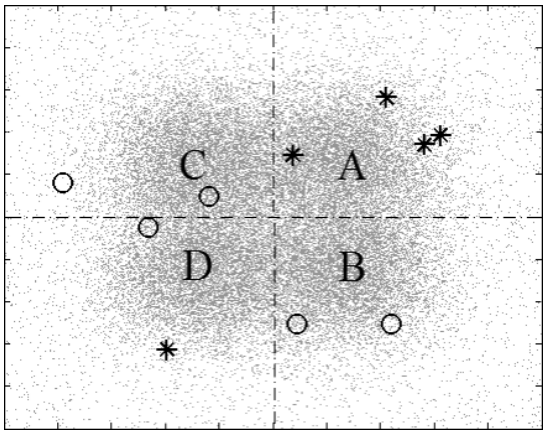


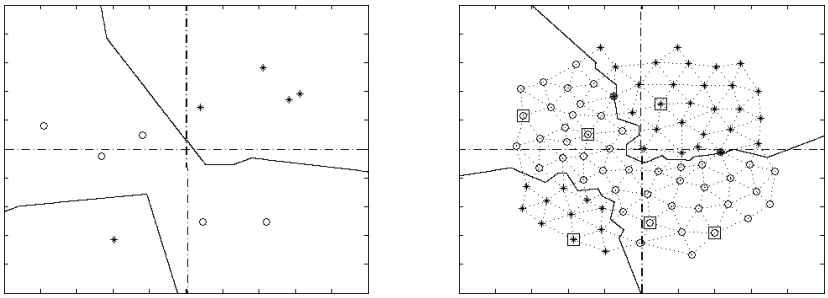
Fig. 6. Artificial dataset: generated from four Gaussian distributions including overlap and 10% random noise (two dimensions, two classes). Areas A and D belong to the same class as area B and C do. Black $*$ and $\circ$ are 10 labeled samples. Gray dots are 39,990 unlabeled samples.

To evaluate learning in the stationary environment, we first randomly chose each pattern from among all 40,000 samples to train the proposed network using all distributions and classes online simultaneously. No new distribution or class occurred during training. **Fig. 7(b)** shows training results for our proposal after 40,000 inputs.

Figure 7(b) shows that our proposal eliminates noise and learns the topological structure well. Because, all nodes are connected with edges, however, topology representation cannot discriminate among class distributions. Labeling learning nodes engender division using information for a small number of labeled patterns. The predicted class labels of each node, as shown in **Fig. 7(b)**, suggest that our proposal divides overlapped classes, so when input data distribution contains overlaps, our proposal realizes online learning to represent the topology of the input distribution and divide overlapped classes, making a more effective decision boundary than that using 1-NN with only labeled patterns (**Fig. 7(a)**). The degree of coincidence between ideal and resulting boundaries is 85.68% for our proposal and 80.11% for 1-NN.

In evaluating learning in the non-stationary incremental environment, we simulate online incremental learning using the following paradigm: from step 1 to step 10,000, area A patterns are chosen randomly. At step 10,001, the environment changes and area B patterns are chosen. At step 20,001, the environment changes again, ad infinitum.

At right in **Fig. 8** are results for our proposal, which, after learning in the first environment, learned the topological structure in a region of area A without noise. Probability then changes to zero in the region of area A. Remaining nodes, often called "dead nodes," play a major role in online incremental learning, preserving knowledge from previous situations for future decisions. As in the stationary environment, our proposal realizes online incre-



(a) Decision boundary using 1-NN

(b) Learning result of the proposed method and its decision boundary ($\lambda = 100$, $a_d = 100$, $\varepsilon_1 = 1/M_{s_1}$, and $\varepsilon_2 = 1/100M_i$ $(\forall i \in N_{s_1})$)

Fig. 7. Online semi-supervised learning results in the stationary environment: the observed decision boundary is represented as a solid line; the ideal boundary is shown as a chain line. Other marks represent the following: (a) $*$ and $\circ$ indicate prototypes of each class; (b) labeled nodes represented as $\square$, $*$ and $\circ$ indicate the predicted class labels of each node. Network edges are shown as dotted lines.

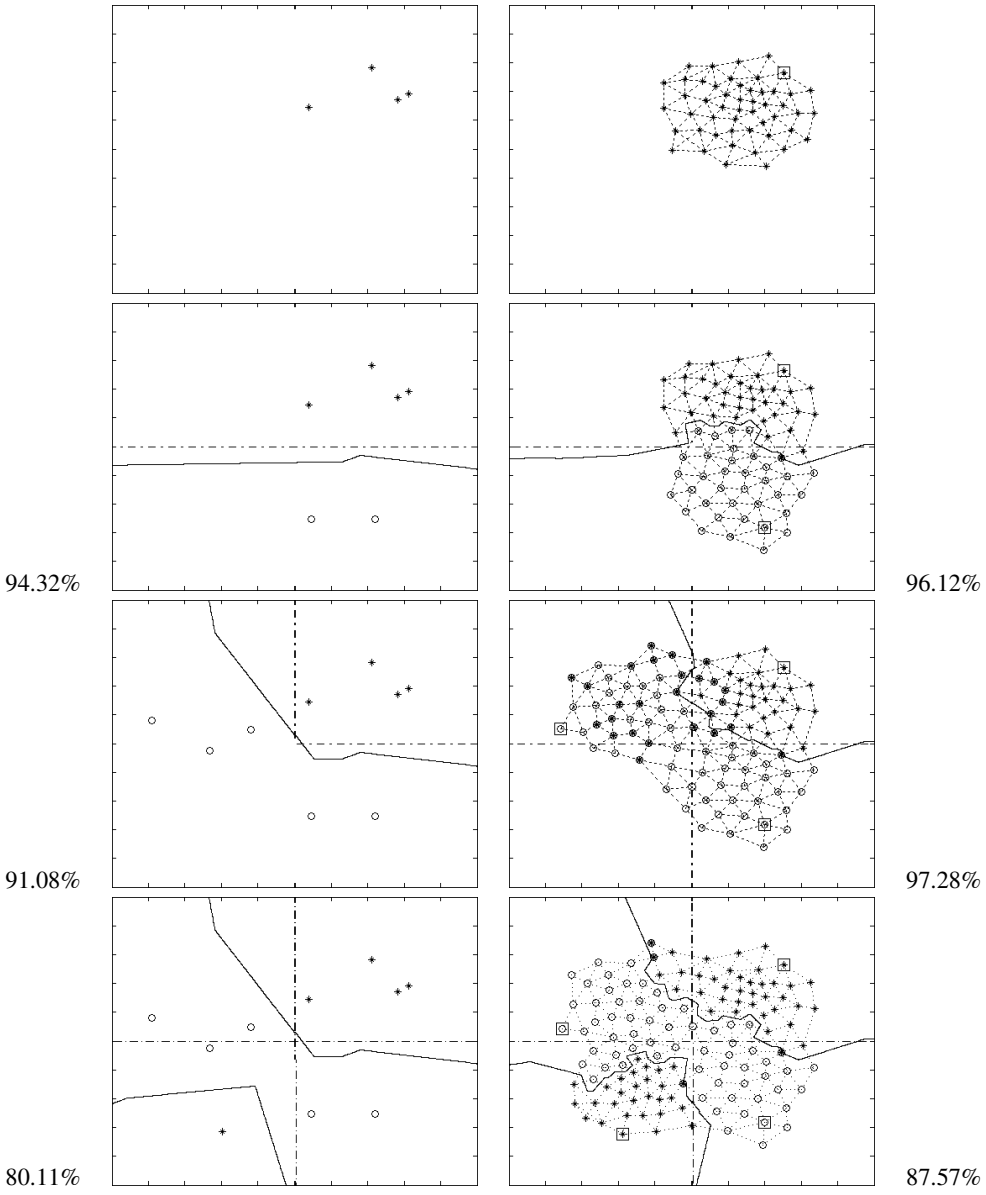


Fig. 8. Online incremental semi-supervised learning results in the non-stationary environment: decision boundary using 1-NN (left) and learning result and decision boundary using our proposal with $\lambda = 100$, $a_d = 100$, $\varepsilon_1 = 1/M_{s_1}$, and $\varepsilon_2 = 1/100M_i$ $(\forall i \in N_{s_1})$ (right). Figures on both sides are the degree of coincidence between ideal and resulting boundaries.

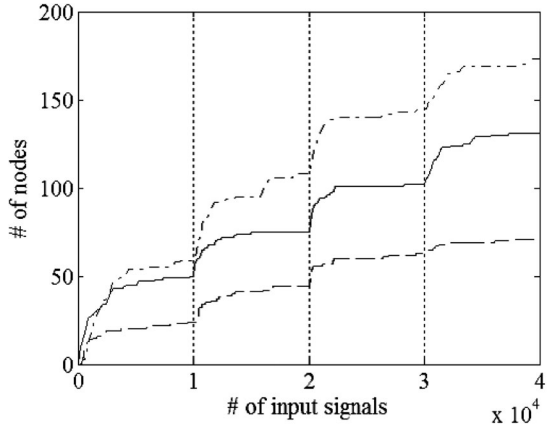


Fig. 9. Change in the number of nodes in the non-stationary environment with different parameter sets $\{\lambda, a_d\}$, {50, 50} (dotted line), {100, 100} (solid line), and {200, 200} (chain line). Changes in the number of nodes throughout learning indicate a similar trend with different parameters values.

mental semi-supervised learning and discriminates among classes in the non-stationary environment, also producing a more effective decision boundary than that of 1-NN. The degree of coincidence between ideal and resulting boundary is 96.12% – 97.28% – 87.57% for our proposal method and 94.32% – 91.08% – 80.11% for 1-NN.

Figure 9 shows how the number of nodes changes during learning, increasing when the input signal originates from a new area. After some learning steps in the same environment, the node increase stops and the number of nodes converges to a constant. Results obtained using three different parameter sets shows that the number of nodes for each distribution is related directly to λ and a_d. Changes in the number of nodes occurring during learning indicate a trend similar to that obtained with different parameter values, showing that our proposal avoids permanently increasing the number of nodes. Our proposal thus works well in online semi-supervised learning tasks even in noisy and non-stationary incremental environments.

3.2. Real-World Dataset

Using the optical recognition of handwritten digits database (optdigits) [21], we evaluated our proposal's performance using different parameter sets. This database obtains, 10 classes of handwritten digits from 43 people – 30 contributing to the training set and 13 others contributing to the test set. Numbers of training and test set samples are listed in **Table 1**. The training set has 3823 samples and the test set 1797. Sample dimensionality is 64.

We consider a classification task in which labeled samples are difficult to obtain. Setting only one labeled training sample per class makes available 10 labeled and 3813 unlabeled samples for training, so classifier learning results depend on labeled sample selection from training samples. We set 10 subsets for labeled samples and tested classifiers with each subset to reduce data selection bias. With traditional 1-NN and using only labeled samples as prototypes, we classify test samples into different classes using Euclidean distance as the metric. The average recognition ratio of 1-NN is $65.45 \pm 7.64\%$.

Table 1. Number of samples in optdigits database.

Class	Samples	
	Training	Test
0	376	178
1	389	182
2	380	177
3	389	183
4	387	181
5	376	182
6	377	181
7	387	179
8	380	174
9	382	180
Total	3823	1797

We tested our proposal in a non-stationary incremental environment. We began by randomly selecting samples from the class 0 training set to train the system. No samples from other classes were chosen. After 4,000 training iterations, samples input are selected randomly from the class 1 training set and trained 4,000 times. Classes 2, 3, ..., 9 are then trained in the same way, after which we obtain nodes of the generated self-organizing network and label all nodes using labeled samples. We then use the weights of such nodes as prototypes to classify test samples for the test dataset and report the correct recognition ratio. As stated, the learning result for our proposal depends on labeled input pattern selection. We tested our proposal on 10 different orders of input pattern, creating 100 classifiers – 10 kinds of labeled subsets $\times$ 10 input sequences – and report the average result for these 100 classifiers.

During training, we also tested three different parameter sets for $\{\lambda, a_d\}$ – (1) {100, 100}, (2) {500, 500}, and (3) {1000, 1000}, obtaining different results as listed in **Table 2**. Using weights of nodes as prototype vectors, we classify test samples into different class and report recognition results for different parameter sets in **Table 3**. For different classes, **Table 2** shows that the number of nodes needed to represent classes differ. **Table 3** shows four facts: (i) compared to 1-NN, our proposal improves the correct recognition ratio from 65.45% to 72.64% for parameter set (1) using unlabeled training patterns. (ii) Both the number of nodes and the correct recognition ratio change with parameters, e.g., parameter sets $\{\lambda, a_d\}$ are useful in controlling classification. (iii) With large numbers of training samples, our proposal generated only a few prototypes, e.g., with parameter set (1), only 56 ± 11 prototypes are generated from 3823 training samples, meaning that online incremental semi-supervised learning deals with large-scale datasets with large amounts of unlabeled data. (iv) Even if larger $\{\lambda, a_d\}$ generates more nodes and represents training data better, the limited number of labeled samples limits the

Table 2. Average number of nodes for different classes of the optdigits database with different parameter sets $\{\lambda, a_d\}$.

Class	# of nodes for different sets of $\{\lambda, a_d\}$		
	(1)	(2)	(3)
0	2 ± 1	9 ± 3	14 ± 6
1	4 ± 2	9 ± 5	12 ± 5
2	5 ± 2	12 ± 4	19 ± 7
3	4 ± 4	13 ± 5	21 ± 9
4	4 ± 2	10 ± 6	17 ± 9
5	3 ± 2	10 ± 6	16 ± 6
6	5 ± 2	15 ± 3	23 ± 7
7	4 ± 2	12 ± 4	20 ± 8
8	5 ± 3	13 ± 5	20 ± 11
9	3 ± 2	8 ± 5	10 ± 8
Total number	56 ± 11	143 ± 17	229 ± 17

Table 3. Comparison of classification results of the proposed method for the optdigits database with different parameter sets $\{\lambda, a_d\}$. % means recognition ratio; # means number of prototypes.

	set of $\{\lambda, a_d\}$		
	(1)	(2)	(3)
%	72.64 ± 10.23	74.25 ± 7.66	74.33 ± 7.19
#	56 ± 11	143 ± 17	229 ± 17

increase in classification. Compared to parameter set (2) (143 ± 17 prototypes), parameter set (3) generated more prototypes (229 ± 17), but classification is not noticeably improved for parameter set (3).

3.3. Comparison with Other Methods

As stated in Section 2, our proposal makes direct predictions for unknown patterns, making it useful as a classifier. Preceding experiments using artificial and real-world datasets show that our proposal produce a more effective decision boundary than 1-NN. We compare our proposal to state-of-the-art classifiers in supervised and semi-supervised learning using typical real-world datasets.

Of the five real-world datasets we used, two were from the UCI Machine Learning repository [21] and three from the experiment in [4], Chapter 21. We used only two-class classification tasks to compare classifiers strictly, observing the area under the ROC curve (AUC) [22] for each method. **Table 4** shows dataset properties. For details see [4, 21].

3.3.1. Experimental Setup

The experimental setup was semi-supervised and the test set used "out-of-sample" patterns, separate from the unlabeled training pattern set. Labeled patterns numbered either 10 or 100 for all datasets. We devised 10-12 subsets (splits) for labeled patterns to produce accuracy estimates and coincidental properties of the labeled patterns chosen. For Pima and Cancer datasets, we chose random labeled patterns, producing 10 subsets. For other datasets, we use 12 subsets provided at the following website:
http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html

Table 4. Properties of benchmark datasets.

Datasets	Classes	Dimension	Samples
Pima	2	8	768
Cancer	2	9	699
BCI	2	117	400
USPS	2	241	1500
COIL	2	241	1500

We were careful to pick at least one pattern from each class when choosing subsets of labeled patterns.

We selected the following semi-supervised learning for comparison: Transductive SVM (TSVM) [23], Laplacian RLS (LapRLS), and Laplacian SVM (LapSVM) [19]. We also compared our proposal to supervised learning: 1-NN, SVM, and ssEAM. While these cannot conduct online incremental learning, as far as we know, but no efficient incremental semi-supervised learning has been reported. Please note that we compare only learning performance between our proposal and typical semi-supervised methods, even if our proposal targets to realize online incremental learning.

We combine numerous user-determined parameter values and use 10-fold Cross-Validation (CV) to tune parameters and avoid data and parameter selection bias.

In addition to recognition accuracy, we calculated AUC [22], the area under the Receiver Operating Characteristic (ROC) curve, to measure classifier performance. AUC is actually calculable only in two-class classification tasks, representing a balance between sensitivity and loss in classifier specificity. It is invariant to prior probabilities. With 10-fold CV, we calculated 10 pairs of true positive rate (sensitivity) and false positive rate (loss in specificity) for the test set. These pairs are plotted and are used in trapezoidal integration to calculate AUC for each of the 10 or 12 splits. We averaged AUC across all 10 or 12 splits to measure dataset classifier performance.

For each learning method, we describe production of the classifier and combinations with user-determined parameter values.

1. 1-NN: A classifier is built using only the small set of labeled patterns as prototypes.

2. SVM: As in 1-NN, only labeled patterns are used for learning in SVM. We use SVM^{light} (http://svmlight.joachims.org/). We report the result of nonlinear SVM because nonlinear SVM is more effective than linear one in most datasets. We implement SVM with a Radial Basis Function (RBF) kernel. Its width is chosen as the median of pairwise distances and C is fixed to 100. We set these conditions based on [4].

3. ssEAM: ssEAM is trained on online supervised learning, so each input pattern is selected randomly from among labeled patterns. Training iterations are

Table 5. Average recognition rate (%) and standard deviation with 10 or 100 labeled samples.

	supervised learning			semi-supervised learning			
	1-NN	SVM	ssEAM	**Proposed**	TSVM	LapRLS	LapSVM
Cancer	90.22 ± 5.82	90.03 ± 6.41	92.02 ± 3.30	91.43 ± 3.63	88.66 ± 4.28	91.24 ± 4.15	91.09 ± 3.15
Pima	64.19 ± 2.67	65.98 ± 2.04	65.59 ± 0.74	64.22 ± 2.73	64.81 ± 3.81	67.73 ± 3.54	67.82 ± 2.80
BCI	51.28 ± 2.74	52.75 ± 2.54	52.45 ± 2.13	51.06 ± 1.31	50.69 ± 2.30	54.21 ± 2.62	53.97 ± 3.06
USPS	80.18 ± 3.79	80.43 ± 1.82	80.38 ± 1.20	79.98 ± 5.02	73.16 ± 2.67	80.75 ± 0.73	81.46 ± 1.45
COIL	56.98 ± 3.12	56.09 ± 3.57	60.51 ± 3.53	61.20 ± 3.84	56.72 ± 2.85	58.47 ± 5.93	57.48 ± 5.25

(a) With 10 labeled samples

	supervised learning			semi-supervised learning			
	1-NN	SVM	ssEAM	**Proposed**	TSVM	LapRLS	LapSVM
Cancer	93.16 ± 1.18	92.54 ± 1.83	94.00 ± 1.00	93.04 ± 1.39	91.05 ± 2.22	93.96 ± 0.75	94.21 ± 0.49
Pima	65.46 ± 3.00	72.06 ± 4.07	65.49 ± 1.20	66.28 ± 3.20	72.76 ± 2.13	74.70 ± 1.92	74.23 ± 4.01
BCI	55.20 ± 1.72	66.03 ± 2.68	56.87 ± 2.18	52.86 ± 1.64	65.39 ± 3.28	72.52 ± 2.80	71.47 ± 3.27
USPS	92.36 ± 0.83	91.06 ± 1.67	92.36 ± 0.83	92.18 ± 1.30	90.35 ± 1.37	87.94 ± 0.75	88.35 ± 1.02
COIL	88.16 ± 1.62	87.01 ± 1.90	88.16 ± 1.62	89.89 ± 1.57	86.95 ± 1.81	85.38 ± 1.48	83.77 ± 2.25

(b) With 100 labeled samples

Table 6. Average area under ROC curve (AUC; %) and standard deviation with 10 or 100 labeled samples.

	supervised learning			semi-supervised learning			
	1-NN	SVM	ssEAM	**Proposed**	TSVM	LapRLS	LapSVM
Cancer	88.97 ± 8.77	87.75 ± 10.99	93.77 ± 3.09	92.22 ± 6.29	89.90 ± 3.59	88.52 ± 7.57	86.74 ± 8.03
Pima	55.78 ± 7.60	55.62 ± 7.51	54.92 ± 5.75	56.22 ± 9.70	59.33 ± 5.13	59.14 ± 8.52	60.98 ± 8.65
BCI	48.35 ± 7.52	51.16 ± 5.94	53.82 ± 4.31	51.26 ± 3.58	50.37 ± 2.79	53.36 ± 5.65	52.33 ± 7.16
USPS	64.01 ± 8.46	56.58 ± 6.03	57.52 ± 5.74	78.47 ± 7.50	56.85 ± 8.79	50.47 ± 1.13	55.17 ± 5.15
COIL	58.10 ± 2.98	57.08 ± 5.00	65.69 ± 4.15	65.29 ± 5.82	56.89 ± 2.78	59.51 ± 6.38	58.66 ± 5.53

(a) With 10 labeled samples

	supervised learning			semi-supervised learning			
	1-NN	SVM	ssEAM	**Proposed**	TSVM	LapRLS	LapSVM
Cancer	93.40 ± 2.43	91.93 ± 3.35	93.19 ± 4.59	93.44 ± 4.24	91.95 ± 4.22	93.70 ± 2.37	94.23 ± 2.16
Pima	62.03 ± 4.20	61.20 ± 6.26	57.12 ± 4.86	61.61 ± 3.91	70.35 ± 5.22	66.13 ± 4.57	66.88 ± 6.12
BCI	56.04 ± 3.75	69.06 ± 5.06	59.28 ± 3.47	51.15 ± 4.92	70.03 ± 3.38	75.82 ± 4.53	75.10 ± 4.92
USPS	85.84 ± 3.81	82.40 ± 4.59	85.84 ± 3.81	90.91 ± 3.33	86.25 ± 4.21	74.19 ± 5.36	76.98 ± 3.32
COIL	90.11 ± 1.62	89.20 ± 2.73	90.11 ± 1.62	92.35 ± 3.13	89.96 ± 0.95	88.67 ± 1.86	87.22 ± 1.51

(b) With 100 labeled samples

continued in increments of training patterns until no new categories are inserted after a training pattern set is learned. Four hyperparameters μ, ρ, α, and ε are varied in $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and are chosen through a search of all combinations.

4. Our proposal: Our proposal involves online incremental semi-supervised learning, so input patterns are selected randomly from training patterns regardless of pattern labeling not. The number of training iterations for all datasets is 1000 times greater than the number of all training patterns. After training, an obtained classifier is built using the weight of labeled nodes as prototypes. Hyperparameters λ and a_d are chosen in a search of the following grid: $\lambda \in \{1000, 4000, 7000, 10000\}$; $a_d \in \{1000, 4000, 7000, 10000\}$. The learning rate over time is set to $\varepsilon_1 = 1/M_{s_1}$ and $\varepsilon_2 = 1/100M_i$.

5. TSVM: TSVM is usually used in a transductive setting, but predicts the unknown pattern class if we "freeze" it after training with labeled and unlabeled patterns. We implement TSVM with the RBF kernel using SVM^{light}. Its width and C are the same as for SVM.

6. LapRLS/LapSVM: In LapRLS and LapSVM, experiments were conducted using the semi-supervised kernel used either in an SVM or in Regularized Least Squares (RLS). The kernel taken the following form:

$$\tilde{K}(x,z) = K(x,z) - \mathbf{k}_x{}^\top (I + rL^p G)^{-1} L^p \mathbf{k}_z, \quad (15)$$

$K(x,z)$ is a base kernel, $[\mathbf{k}_x]_i = K(x_i, x)$, G the Gram matrix, L the normalized graph Laplacian, and r the ratio $\frac{\gamma_I}{\gamma_A}$. For details see [4].

3.3.2. Observations

Table 5 presents the average recognition rate and standard deviation for the test set for each classifier and each dataset. **Table 6** presents the average AUC and standard

Table 7. Rank ordered significant subgroups from Tukey's multiple comparison test on average AUC.

	Supervised learning			Semi-supervised learning			
	1-NN	SVM	ssEAM	**Proposed**	TSVM	LapRLS	LapSVM
Cancer	1	1	1	1	1	1	1
Pima	1	1	1	1	1	1	1
BCI	1	1	1	1	1	1	1
USPS	2	2, 3	2, 3	1	2, 3	2, 3	2, 3
COIL	3	3	1	1, 2	3	2, 3	3

(a) With 10 labeled samples

	supervised learning			semi-supervised learning			
	1-NN	SVM	ssEAM	**Proposed**	TSVM	LapRLS	LapSVM
Cancer	1	1	1	1	1	1	1
Pima	2, 3	2, 3	3	2, 3	1	1, 2	1, 2
BCI	4, 5	3	4	5	2, 3	1	1, 2
USPS	1, 2	2	1, 2	1	1, 2	3	3
COIL	1, 2	2	1, 2	1	1, 2	2, 3	3

(b) With 100 labeled samples

Table 8. Comparison of proposal and other methods.

Method	Transductive setting	Semi-supervised setting	Incremental learning	Low dimension	High dimension
TSVM	◯	×	×	◯	△
LapRLS	×	◯	×	◯	△
LapSVM	×	◯	×	◯	△
Proposed	×	◯	◯	◯	◯

◯: best, △: lower performance than ◯, ×: unsuitable for condition

deviation. **Tables 5** and **6** show that, with 10 or 100 labeled samples, our proposal works best or near best for Cancer, USPS, and COIL.

Semi-supervised learning is expected to indicate better performance than supervised learning because this is the purpose of using unlabeled training patterns, but unlabeled data does not always help semi-supervised learning, as observed elsewhere [5].

For Cancer and Pima, which are low-dimensional datasets, 1-NN and our proposal (both are prototype-based) indicate equivalent performance with either 10 or 100 labeled patterns. TSVM does not work as well as supervised SVM. Both LapRLS and LapSVM indicate equivalent or better performance than supervised learning. For BCI, which is a 117-dimensional dataset, results are similar among all seven learning methods, as those for Cancer and Pima.

For high-dimensional USPS and COIL datasets, conventional semi-supervised learning cannot use unsupervised training patterns efficiently with either 10 or 100 labeled patterns. Our proposal mostly attains equivalent or better classification performance than conventional methods, indicating high AUC performance, and indicating our proposal achieves an appropriate balance between sensitivity and loss in classifier specificity.

To compare classification performance for any pair of classifier types for each dataset, we implement Tukey's multiple comparison test, setting the least significant difference to 5% ($\alpha = 0.05$). **Table 7** show the results of multiple comparison. Number "1" is the group of classifiers with the best performance and not significantly different. Number "2" means the second group, etc. **Table 7** shows that, according to Tukey's multiple comparison test on average AUC for 10 labeled samples, for all datasets, our proposal belongs to the group with best performance. For 100 labeled samples, Cancer, USPS, and COIL our proposal belongs to the group with best performance.

Experiments using artificial and real-world data thus verify that our proposal realizes online incremental learning well, and even compared to batch learning semi-supervised, our proposal remains comparable.

4. Conclusions

The online incremental semi-supervised learning algorithm we have proposed learns the input data topology structure and divides clusters as needed through topology learning and labeling learned nodes. Using node weights as prototypes, we design a proposed classifier with 1-NN as the classification rule.

Table 8 compares proposed and conventional methods. Only our proposal realizes incremental learning and works better than other methods for high-dimensional data.

Results for artificial and real-world datasets showed that our proposal's clustering works well for online incremental learning tasks even in noisy and non-stationary

environments. Even compared to conventional supervised and semi-supervised batch learning, our proposal remains comparable and equivalent or better than conventional semi-supervised learning for data exceeding 200 dimensions.

Acknowledgements

This work was supported in part by China NSF grant #60975047, #60723003, #60721002, Jiangsu NSF grant #BK2009080, and 973 Program 2010CB327903. It was also supported in part by NEDO of Japan.

References:

[1] S. Grossberg, "Nonlinear neural networks: principles, mechanisms, and architectures," Neural Networks, Vol.1, pp. 17-61, 1988.
[2] F. H. Hamker, "Life-long Learning Cell Structures – Continuously Learning without Catastrophic Interference," Neural Networks, Vol.14, No.4, pp.551-572, 2001.
[3] F. Shen and O. Hasegawa, "An Incremental Network for On-line Unsupervised Classification and Topology Learning," Neural Networks, Vol.19, No.1, pp. 90-106, 2006.
[4] O. Chapelle, B. Schölkopf, and A. Zien, editors, "Semi-Supervised Learning," MIT Press, 2006.
[5] X. Zhu, "Semi-Supervised Learning Literature Survey," Technical Report 1530, University of Wisconsin, Madison, 2005.
[6] A. Dong and B. Bhanu, "Active Concept Learning in Image Databases," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSıPART B: CYBERNETICS, Vol.35, No.3, pp. 450-466, 2005.
[7] J. Tang, X.-S. Hua, M. Wang, Z. Gu, G.-J. Qi, and X. Wu, "Correlative Linear Neighborhood Propagation for Video Annotation," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSıPART B: CYBERNETICS, Vol.39, No.2, pp. 409-416, 2009.
[8] T. Kohonen, editor, "Self-Organizing Maps," Springer Series in Information Sciences, Springe, Heidelberg, 1997.
[9] H. Bauer and T. Villmann, "Growing a hypercubical output space in a self-organizing feature map," IEEE Trans. on Neural Networks, Vol.8, No.2, pp. 218-226, March 1997.
[10] B. Fritzke, "Growing Cell Structures – A Self-organizing Network for Unsupervised and Supervised Learning," Neural Networks, Vol.7, No.9, pp. 1441-1460, 1994.
[11] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," In Neural Information Processing Systems, Vol.7, pp. 625-632, Denver, USA, 1995, MIT Press.
[12] Y. Prudent and A. Ennaji, "An Incremental Growing Neural Gas Learns Topologies," In Proc. of the IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks (IJCNN '05), pp. 1211-1216, Montreal, Canada, 2005.
[13] S. Grossberg, "Adaptive Pattern Recognition and Universal Encoding II: Feedback, Expectation, Olfaction, and Illusions," Biological Cybernetics, Vol.23, pp. 187-202, 1976.
[14] 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. on Neural Networks, Vol.3, No.5, pp. 698-713, 1992.
[15] G. C. Anagnostopoulos, M. Bharadwaj, M. Georgiopoulos, S. J. Verzi, and G. L. Heileman, "Exemplar-based Pattern Recognition via Semi-Supervised Learning," In Proc. of the IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks (IJCNN '03), Vol.3, pp. 1350-1356, 2003.
[16] F. Shen, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," Neural Networks, Vol.20, pp. 893-903, 2007.
[17] T. M. Martinetz, "Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps," In Proc. of Int. Conf. on Artificial Neural Network (ICANN), pp. 427-434, 1993.
[18] T. M. Martinetz and K. J. Schulten, "Topology Representing Networks," Neural Networks, Vol.7, No.3, pp. 507-522, 1994.
[19] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the Point Cloud: from Transductive to Semi-Supervised Learning," In Proc. of the 22nd Int. Conf. on Machine Learning (ICML 05), pp. 824-831, New York, NY, USA, 2005, ACM Press.
[20] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. on Information Theory, IT–13, No.1, pp. 21-27, 1967.
[21] C. Merz and M. Murphy, "UCI Repository of Machine Learning Databases," Technical report, University of California Department of Information, Irvine, CA.
[22] A. P. Bradley, "The Use of the Area Under the ROC curve in the Evaluation of Machine Learning Algorithms", Pattern Recognition, Vol.30, No.7, pp. 1145-1159, 1997.
[23] V. Vapnik, editor, "Statistical Learning Theory," Wiley, New York, 1998.

Name:
Furao Shen

Affiliation:
Associate Professor, the State Key Laboratory for Novel Software Technology, and Jiangyin Information Technology Research Institute, Nanjing University

Address:
Nanjing 210093, P.R. China

Brief Biographical History:
1991-1998 Department of Mathematics, Nanjing University
2003-2006 Tokyo Institute of Technology
2006- Department of Computer Science and Technology, Nanjing University

Main Works:
• F. Shen and O. Hasegawa, "An Incremental Network for On-line Unsupervised Classification and Topology Learning," Neural Networks, Vol.19, pp. 90-106, 2006.
• F. Shen, T. Ogura, and O. Hasgeawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," Neural Networks, Vol.20, pp. 893-903, 2007.

Membership in Academic Societies:
• IEEE, Member
• China Computer Federation, Member

Name:
Hui Yu

Affiliation:
Assistant, Jiangyin Information Technology Research Institute, Nanjing University

Address:
Nanjing 210093, P.R. China

Brief Biographical History:
2003-2008 Department of Electrical and Electronic Engineering, The University of Tokushima
2009- Jiangyin Information Technology Research Institute, Nanjing University

Main Works:
• F. Shen, H. Yu, W. Kasai, and O. Hasegawa, "An Associative Memory System for Incremental Learning and Temporal Sequence," IJCNN 2010.

Membership in Academic Societies:
• China Computer Federation, Member

Name:
Youki Kamiya

Affiliation:
C&C Innovation Research Laboratories, NEC Corporation, Japan

Brief Biographical History:
2007 Received Dr.Eng. degree from Dept. of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology

Main Works:
• Y. Kamiya, F. Shen, and O. Hasegawa, "An Incremental Neural Network for Online Supervised Learning and Topology Learning," J. of Advanced Computational Intelligence and Intelligent Informatics, 2006.

Name:
Osamu Hasegawa

Affiliation:
Associate Professor, Imaging Science and Engineering Laboratory, Tokyo Institute of Technology

Address:
R2-52, 4259 Nagatsuda, Midori-ku, Yokohama 226-8503, Japan

Brief Biographical History:
1993 Received the Dr.Eng. degree in Electronic Engineering from the University of Tokyo
1993-1999 Research Scientist, the Electrotechnical Laboratory
1999-2000 Visiting Scientist, the Robotics Institute, Carnegie Mellon University
2000-2002 Research Scientist, the National Institute of Advanced Industrial Science and Technology
2002- Faculty Member, the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology
2002- Researcher at PRESTO, Japan Science and Technology Agency

Main Works:
• A. Sudo, A. Sato, and O. Hasegawa, "Associative Memory for Online Learning in Noisy Environments Using Self-organizing Incremental Neural Network," IEEE Trans. on Neural Networks, Vol.20, No.6, pp. 964-672, Jun. 2009.
• T. Toyoda and O. Hasegawa, "Random Field Model for Integration of Local Information and Global Information," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.30, No.8, pp. 1483-1489, Apr. 2008.

Membership in Academic Societies:
• IEEE Computer Society
• Institute of Electronics Information and Communication Engineers (IEICE)
• Information Processing Society of Japan (IPSJ)