An Incremental Network with Local Experts Ensemble

Shaofeng Shen[1], Qiang Gan[1], Furao Shen[1(✉)], Chaomin Luo[2], and Jinxi Zhao[1]

[1] National Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology,
Nanjing University, Nanjing, China
{shaofeng2014,njucsgq}@gmail.com, {frshen,jxzhao}@nju.edu.cn
[2] Department of Electrical and Computer Engineering,
University of Detroit Mercy, Detroit, USA
luoch@udmercy.edu

Abstract. Ensemble learning algorithms aim to train a group of classifiers to enhance the generalization ability. However, vast of those algorithms are learning in batches and the base classifiers (e.g. number, type) must be predetermined. In this paper, we propose an ensemble algorithm called INLEX (**I**ncremental **N**etwork with **L**ocal **EX**perts ensemble) to learn suitable number of linear classifiers in an online incremental mode. Specifically, it incrementally learns the representational nodes of the input space. In the incremental process, INLEX finds nodes in the decision boundary area (boundary nodes) based on the theory of entropy: boundary nodes are considered to be disordered. In this paper, boundary nodes are activated as experts, each of which is a local linear classifier. Combination of these linear experts with dynamical weights will constitute a decision boundary to solve nonlinear classification tasks. Experimental results show that INLEX obtains promising performance on real-world classification benchmarks.

1 Introduction

Ensemble learning algorithms have attracted much attention for decades. Bagging [1] and AdaBoost [2] are well-known ensemble learning algorithms. They enhance the generalizing ability by training a group of classifiers with different distribution. The combinational strategies are voting (e.g. Bagging) or weighted voting (e.g. AdaBoost). Another well-known ensemble model is Mixture of experts (ME) [3] in which the divide-and-conquer strategy is used. In ME, several local experts are trained to partition the input space, the weights of experts are computed dynamically based on the input.

Online incremental learning is another important topic in machine learning. Online learning algorithms process the data one-by-one and are suitable to the applications with continuous online data or insufficient memory space (e.g. visual tracking, embedded system). Incremental learning addresses the ability

S. Arik et al. (Eds.): ICONIP 2015, Part III, LNCS 9491, pp. 515–522, 2015.
DOI: 10.1007/978-3-319-26555-1_58

of repeatedly training a system using new data without destroying the old prototype patterns [4]. Online incremental learning algorithms often adapt their models to the input well.

In the ensemble learning literature, some online ensemble algorithms have been proposed [5–7]. However, in those methods, the base classifier or expert must be predetermined. That means we must determine both suitable number and type of base classifiers before applying those algorithms. In an online learning system, each data arrives one-by-one. Therefore, some artificial methods (e.g. brute searching, cross-validation) cannot be used. As a result, we can not obtain some prior knowledge such as suitable number and type of the base classifier. We need to get those knowledge in the learning process. That inspires the study of online incremental ensemble learning. Specifically, this issue can be that: (1) the base classifiers are generated in the online incremental process, (2) the base classifiers are efficient and simplify. In another aspect, as far as Minsky concerns that, the human solve the classification problem just by perceptrons. Minsky thinks that the brain is not a unified entity but a society of elements that both complement and complete with one another. For efficiency and simplicity, linear classifier is mostly suitable to be the base classifier in an online ensemble system.

In this paper, we propose an ensemble algorithm called INLEX (**I**ncremental **N**etwork with **L**ocal **EX**perts ensemble) to learn suitable number of linear classifiers in an online incremental mode. INLEX incrementally learns the representational nodes of the training data based on competitive Hebbian rule [8]. In the incremental process, it finds nodes in the decision boundary area (boundary nodes) based on the theory of entropy: boundary nodes are considered to be disordered. In addition, boundary nodes will be activated as experts, each of which is a local linear classifier. In total, INLEX aims to find and train the experts in the decision boundary area. As a result, dynamical combination of them can constitute a decision boundary to solve classification tasks. Experimental results show that INLEX obtains promising performance on real-world classification benchmarks.

2 Proposed Method

In Fig. 1, we show the motivation of INLEX. Illustrated in Fig. 1, we define the area between class A and B as decision boundary area. In step 1, suitable number of nodes are learned. These nodes are able to represent the distribution of the train data. In step 2, we find the nodes in the decision boundary area (boundary nodes) and activate them as experts. Each expert is a local linear classifier. In step 3, the experts are trained in a supervised way. As a result, these local experts can solve the classification task competitively and complementally. Namely, combination of them can constitute a decision boundary to solve this classification task.

In this paper, we implement that motivation in an online incremental mode in INLEX which can also solve multi-classification tasks. Nodes are generated

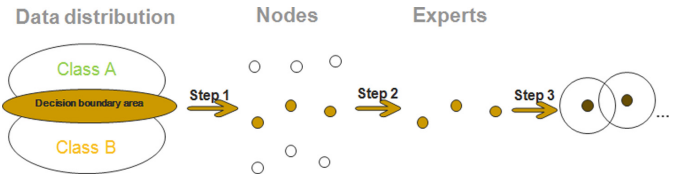

Fig. 1. Motivation of INLEX

incrementally in a self-organized manner. We do not need some prior knowledge such as number of nodes. In the learning process, new experts will be activated if new boundary nodes have generated. Meanwhile, experts in INLEX online learns each labeled data.

To realize these targets, we emphasize the key aspects of nodes growing procedure, boundary nodes detection, experts activation and training.

2.1 Nodes Growing Procedure

To learn the representational nodes of the training data incrementally, competitive Hebbian rule and similarity threshold criterion are used in INLEX. The competitive Hebbian rule can be described as: for each input signal, find its two closest nodes (measured by Euclidean distance) and connect with an edge between them. The similarity threshold criterion is defined in Definition 1. The input signal is a new node if the distance between the signal and the nearest node (or second nearest node) is greater than a threshold T.

Definition 1. ***Similarity Threshold Criterion:*** *If node i has neighbors (there are some nodes connected to i), its similarity threshold is defined by the largest Euclidean distance between i and its neighbors. Otherwise, its similarity threshold is defined by the smallest Euclidean distance between i and other nodes.*

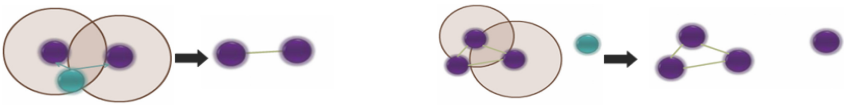

Fig. 2. Two situations of nodes growing procedure. The green nodes are the input signals and other nodes are nodes in INLEX. In Fig. 2(a), an input lies within the local region of its winners, the connection between two winners will be established and the nearest one will be pulled to the input sample. In Fig. 2(b), the input is far from its two winners and lies in a new region. Therefore, a new node will be inserted to represents the new region (Color figure online).

Shown as Fig. 2, based on the competitive Hebbian rule, INLEX learns the topology of the input space. Based on the similarity threshold, new node will

be inserted adaptively. In total, the nodes growing procedure continues in a self-organized manner.

2.2 Boundary Nodes Detection

To solve a classification task, the key is to find the decision boundary. Therefore, in INLEX, only boundary nodes are activated. Herein, we will propose the online boundary nodes detection method.

For an input signal (x, d), where x is the input features vector and d is the label, we can define the similarity between x and node i as:

$$s_i = e^{-||w_{s_i} - x||} \tag{1}$$

where w_{s_i} represents the position of node i. Then, for each input signal, every node i learns its similarity to class d with s_i. Namely, in the learning process, each node i records its accumulated similarity to every class. For T-classification task, the accumulated similarity vector is defined as:

$$sv_i = (p_{1_i}, p_{2_i}, ..., p_{T_i}) \tag{2}$$

p_{t_i} is the accumulated similarity between node i and class t. Then, entropy of sv_i can be used to judge the position of node i. The nodes with the largest values of entropy are considered in the decision boundary area.

In thermodynamics, entropy is a measure of the number of specific ways in which a thermodynamic system may be arranged, commonly understood as a measure of disorder. In sv_i, p_{t_i} is the similarity between node i and class t. Consider that in a binary classification task. If node i is in the boundary between class A and class B, the values of its accumulated similarity points p_{A_i} and p_{B_i} are approaching. Namely, it is similar to both class A and class B. Incorporated with the theory of entropy, we can interpret that as: the similarity vectors of the boundary nodes are usually disordered. Their values of entropy are universally larger than other nodes. For multi-classification tasks, the theory is same. Therefore, the nodes with the largest values of entropy are able to be considered in the decision boundary area.

Because INLEX learns incrementally, it finds boundary nodes based on accumulated entropy. In INLEX, each node records the accumulated similarity to each class based on latest learned L_2 (predefined parameter) input signals. When the learning times are multiple of L_2, each node i calculates the entropy E_i of the similarity vector and adds E_i to its accumulated entropy. The nodes which are not experts and with the largest values of accumulated entropy are activated.

2.3 Experts Activation and Training

Each expert is a linear classifier. Therefore, while node i is activated as an expert, two extra parameters will be assigned to it. The first one in expert i is

$\theta_i = \{w_{i_1}, ..., w_{T_1}\}$. To solve T-classification task, each expert i gives its output for class t $(t = 1, ..., T)$ based on w_{t_i}, namely:

$$O_{t_i} = \frac{1}{1 + e^{-w_{t_i}^T x}} \tag{3}$$

The second extra parameter in expert i is w_{g_i} which is used to assign weight. w_{g_i} is initialized by w_{s_i}. For each input data, the weight g_i of expert i is defined by:

$$g_i = \frac{e^{-||w_{g_i} - x||}}{\sum\limits_{i \in E_s} e^{-||w_{g_i} - x||}} \tag{4}$$

where E_s is the experts set. The training for experts aims to update parameter θ_i and slightly adjust the position of expert i. As a result, these experts solve the classification task both competitively and complementally. Therefore, to preserve the topology of nodes, we assign each expert i an extra parameter w_{g_i} which is used in the experts training process.

Next, INLEX will combine the output values of these experts to make prediction. For each data (x, d), the ensemble output for class t and predictive label are:

$$O_t = \sum_{i \in E_s} g_i O_{t_i} \quad y = arg \max_t O_t \tag{5}$$

We define the loss function as:

$$L_e = -log \sum_{i \in E_s} g_i e^{-\frac{1}{2}(1 - O_{d_i})^2} \tag{6}$$

Computing the derivatives of the loss function with respect to w_{d_i} and w_{g_i} of expert i will yield the update strategies for the parameters under the method of gradient descent. Namely,

$$\Delta w_{g_i} = -\eta(h_i - g_i) \frac{x - w_{g_i}}{||x - w_{g_i}||} O_{t_i}(1 - O_{t_i}) \tag{7}$$

$$\Delta w_{d_i} = \eta h_i (1 - O_{d_i}) O_{d_i} (1 - O_{d_i}) \tag{8}$$

$$h_i = -\frac{g_i e^{-\frac{1}{2}(1 - O_{d_i})^2}}{\sum\limits_{k \in E_s} g_k e^{-\frac{1}{2}(1 - O_{d_k})^2}} \tag{9}$$

In addition, a penalty to each parameter w_{t_i} $(t \neq d, i \in E_s)$ is provide so that the output value for class t decreases. Namely,

$$\Delta w_{t_i} = \eta h_i (0 - O_{t_i}) O_{t_i} (1 - O_{t_i}) \tag{10}$$

In Eqs. (7), (8) and (10), η is the learning rate and calculated by: $\eta = 1/M_j$. M_j is the winning time of node j.

Algorithm 1. Complete algorithm of INLEX

1. Initially, two nodes are initialized randomly. One of them is activated as an expert.
2. Input a labeled data point (x, d).
3. Find the nearest node n_1 and second nearest node n_2 of x.
4. Get the threshold T_{n_1}/T_{n_2} of n_1/n_2 based on the similarity threshold criterion.
5. If $||w_{s_{n_1}} - x|| > T_{n_1}$ or $||w_{s_{n_2}} - x|| > T_{n_2}$, x is a new node. Otherwise, update the positions of n_1 by:
$$\Delta w_{s_{n_1}} = \frac{1}{M_{n_1}}(x - w_{s_{n_1}}) \tag{11}$$
6. If there is no edge connecting n_1 and n_2, connect n_1 and n_2 by an edge. Set the age of this edge to 0. Otherwise, only refresh this age: set its age to 0.
7. Find all edges connected to n_1. Add the ages of those edges by 1. Delete those edges whose ages are larger than a_{max} (predefined parameter).
8. For each node i, calculate the similarity s_i between i and x by Eq. (1), add its accumulated similarity point p_{d_i} by s_i.
9. Each expert updates its parameters according to Eqs. (7), (8) and (10).
10. If the learning times are multiple of L_1 (predefined parameter), go to step 11. Otherwise, go to step 12.
11. Denoising: delete the nodes which are not experts and have no neighboring node.
12. If the learning times are multiple of L_2, go to step 13. Otherwise, goto step 2.
13. Find and activate boundary nodes:
    - Calculated the entropy of each node i in this period:
$$E_i = -\sum_{t=1}^{T} \frac{p_{t_i}}{\sum_{t=1}^{T} p_{t_i}} log(\frac{p_{t_i}}{\sum_{t=1}^{T} p_{t_i}}) \tag{12}$$
    - For each node i, add its accumulated entropy ${E_i}^A$ by E_i. Calculate the maximum/mean accumulated entropy ${E_{max}}^A/{E_{mean}}^A$. The threshold T_e is defined by $({E_{max}}^A + {E_{mean}}^A)/2$.
    - For each node i, if ${E_i}^A$ is larger than T_e, it is a boundary node. And if boundary node i is not an expert, activate node i as expert.
    - Clear the similarity vector.
14. Goto step 2.

2.4 The Complete Algorithm of INLEX

As a summary, we provide the complete algorithm of INLEX in Algorithm 1. In Algorithm 1, while a new node i is inserted, it will be added into nodes set N_s ($N_s = i \cap N_s$). Its accumulated similarity vector is 0. Its accumulated entropy ${E_i}^A$ is set to 0. Its times as winners (M_i) are set to 1. While node i is activated as expert i, it will be added into experts set E_s ($E_s = i \cap E_s$). When the following input signals arrive, the new expert will be trained.

3 Experiments

This section evaluates INLEX's performance on classification benchmark data sets. The details of the data sets are shown in Table 1. They are all from

UCI machine learning repository [9]. In experiments, each data set is divided into ten parts. We randomly select one part as testing set and the remaining parts as training set. Besides, we compare INLEX with leading online Boosting (OnBoost) algorithm proposed in [6] to show its generalizing performance.

Experiment Setup: Before applying OnBoost on classification tasks, suitable number and type of base classifiers must be predetermined. In the experiment, we set the number of base classifiers of OnBoost to 100, which is same as that in [6]. Then, we test the performance of OnBoost with different type of base classifier. The base classifiers of OnBoost are Perceptron, Naive Bayes and Multi-Layer Perceptrons (MLP), respectively. We notate OnBoost with Perceptron, Naive Bayes and Multi-Layer Perceptrons as OB_P, OB_NB and OB_MLP, respectively.

There are three predefined parameters in INLEX. They are a_{max}, L_1 and L_2. We set them to 100, 200, 600, respectively, in our following experiment. In experiments, we also find that our method achieves stable results with respect to settings of these parameters. In addition, the learning times of INLEX on iris, balance-scale and breast are set to 3000. In the remaining data sets, the learning times are set to be identical to the instances number of each training set. Namely, INLEX learns those data sets with only one iteration.

Comparing Results: The error rates of INLEX and OnBoost are summarized in Table 1. At first, we analyze the results of OnBoost. The performances of OnBoost with different type of base classifier are different. When the base classifier is linear (perceptron), OnBoost's performance is poor. OnBoost also gets unstable results when its base classifier is nonlinear. For example, OB_MLP performs well in iris and nursery but much poorly in balance-scale and breast. OB_NB gets good results in breast and balance-scale but poor results in waveform and waveform-n. As the results demonstrating that, when apply OnBoost to solve classification tasks, we must select suitable type of base classifier.

Then, we compare INLEX with OnBoost. Summarized in Table 1, INLEX performs over OnBoost in all the data sets other than nursery. INLEX incorporates the merits of incremental learning. It usually adapts to each data set well. In nursery, the performance of INLEX is worse than that of OB_NB and OB_MLP. It is notable that one of the critical weakness of MLP is that its converging speed is so slow. In our experiment, ON_MLP learns each data set 50 iterations. However, INLEX only learns the data set of nursery one iteration. Therefore, OB_MLP performs better than INLEX in nursery. Even so, OB_MLP performs poorly on some data sets as well.

Next, the experts numbers of INLEX on those data sets are 14, 29.6, 30.1, 17.4, 39.1 and 59.2 respectively. Because the experts are activated from nodes, the number of experts in INLEX is related to the distribution of each data set. It has relevance to the instances number of each data set.

Table 1. Error rates for INLEX VS OnBoost

Data set	Instances	Classes	Attributes	INLEX	OB_P	OB_NB	OB_MLP
Iris	150	3	4	**0.0400**	0.0733	0.0600	**0.0400**
Balance-scale	625	3	4	**0.0996**	0.1520	0.1253	0.1680
Breast	683	2	9	**0.0293**	0.0322	0.0427	0.0864
Waveform-n	5000	3	40	**0.1378**	0.2240	0.1920	0.1680
Waveform	5000	3	21	**0.1436**	0.2820	0.1860	0.1700
Nursery	12960	5	9	0.1134	0.1427	0.0908	**0.0555**

4 Conclusion

In this paper, an online incremental ensemble algorithm called INLEX is proposed. INLEX incrementally learns the nodes in the decision boundary area and activates them as experts. Each expert is a local linear classifier. Combination of experts will constitute a decision boundary to solve classification tasks. Experimental results show that INLEX has promising generalizing performance.

The future work includes decreasing the experts number. Some strategy should be used to avoid activating similar nodes. Specifically, in the experts activation step, if a new boundary node is found, we should use some strategies to judge if it is necessary to activate this node as expert.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (Grant No. 61375064, 61373001 and 61321491), Foundation of Jiangsu NSF (Grant No. BK20131279).

References

1. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
2. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
3. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., et al.: Adaptive mixtures of local experts. Neural Comput. **3**(1), 79–87 (1991)
4. Shen, F.R., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Netw. **19**(1), 90–106 (2006)
5. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Comput. **6**(2), 181–214 (1994)
6. Oza, N.C.: Online bagging and boosting. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2340–2345. IEEE (2005)
7. Chen, S.T., Lin, H.T., Lu, C.J.: An online boosting algorithm with theoretical justifications. In: ICML, pp. 1007–1014 (2012)
8. Martinetz, T., Schulten, K.: Topology representing networks. Neural Netw. **7**(3), 507–522 (1994)
9. Bache, K., Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml