An Incremental One Class Learning Framework for Large Scale Data

Qilin Deng[1], Yi Yang[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
qldeng@qq.com, yangyi868@gmail.com, {frshen,jxzhao}@nju.edu.cn

[2] Department of Electrical and Computer Engineering, University of Detroit Mercy, Detroit, MI, USA
chaominluo@yahoo.com

Abstract. In this paper, we propose a novel one class learning method for the large scale data. In the context of one class learning, the proposed method could automatically learn the appropriate number of prototypes needed to represent the original target examples, and acquire the essential topology structure of target distribution. Then based on the learned topology structure, a neighbors analysis technique is utilized to separate the target examples from outlier examples. Experimental results show that our method can accommodate the large scale data environment, and achieve comparable or preferable performance than other contemporary methods on both artificial and real word data sets.

Keywords: One class learning · Incremental learning · Topology learning · Neighbors analysis

1 Introduction

One class classification (OCC) [1] addresses such learning problems where only specific examples are available for training the classifier. In this case, the available examples are called the target class, and examples not in this target class the outlier class. The objective of OCC is to separate target examples from examples not belonging to the target class. For the OCC problems, existing methods may have different characteristics concerning the description ability, the robustness against noises, the space and time complexity, etc. However, most of them including the state-of-the-art SVDD [2] or OCSVM [3] could not directly solve large scale learning problems very well. e.g., Nearest Neighbor(s) (NN) and Kernel Density Estimation (KDE) have to maintain the whole training data for testing, while SVDD or OCSVM need to solve a quadratic programming problem which will suffer from the large training data. Consequently, it is practically infeasible or very costly to build description models for OCC problems on large scale

A. Hirose et al. (Eds.): ICONIP 2016, Part II, LNCS 9948, pp. 411–418, 2016.
DOI: 10.1007/978-3-319-46672-9_46

data by traditional methods. Meanwhile, large scale data may have very complex manifold structure, arbitrarily shaped distribution, multiple distributions with different shapes and different density levels. Some OCC methods make a Gaussian or spherical distribution assumption [1], thus these methods may result in a high false positive rate if the assumption is inconsistent with the fact. Moreover, It is common in practice that the training data may be corrupted by noises, and some traditional methods like nearest neighbor are very sensitive to these noisy data. Therefore, an important characteristic of the one class classifiers is their robustness against a few noises or outliers in the target training data.

In this paper, we address the problem of one class learning on large scale data, and propose an incremental one class learning framework. In comparison with other OCC methods, the main contribution of our work can be summarized as follows: (1) It can realize fast incremental one class learning in the situation of large scale data; (2) It could remove the noises and outliers automatically during learning process, thus is more robust to noises or outliers in the target examples; (3) It could detect complex manifold structure or multiple distributions with arbitrarily shapes, and recognize targets and outliers effectively and efficiently.

2 Proposed Method

To deal with those issues stated above, we propose an incremental learning framework named Topology Learning and Neighbors Analysis (TLNA). TLNA first performs incremental topology learning on the target training data to build a graph topological structure which captures the essential knowledge about the target class. Then based on the new graph representation, an effective nearest neighbors analysis technique is utilized to construct the one class classifier.

2.1 Incremental Topology Learning

In order to acquire a concise internal representation for the large scale training data, we adopt topology learning to learn the essential topological relations underlying the training data. The competitive Hebbian learning rule [5], which is an combination of competitive learning and Hebbian theory, is an effective method to construct such topological structure of input data. The principle of this method can be stated as: given some nodes in the input space, for each input pattern x, update the closest node and connect the two closest nodes by an edge. It is proved that the resulting graph is a subgraph of the Delaunay triangulation corresponding to the given set of nodes, and that the resulting graph structure is optimally topology preserving in a very general sense.

In the circumstances of large scale training data, we want to realize incremental topology learning, i.e., successively insert topological nodes and edges connecting these nodes to construct the graph structure $G = (V, E)$, where V is the learned node set, and E is the edge set. Some incremental models such as Growing Neural Gas (GNG) [6] adopt a periodic splitting scheme to insert new nodes nearby the node with maximum accumulated error. However, we employ

an adaptive threshold scheme, i.e., associate a similarity threshold T_i with each node i. If node i has topological neighbors, its similarity threshold is defined by the maximum Euclidean distance between node i and all of its topological neighbors $N(i)$. Thus, we have:

$$T_i = \max_{j \in N(i)} \|w_i - w_j\| \tag{1}$$

Otherwise, its similarity threshold is defined as positive infinity [7]. For an input x, the two nearest nodes of x is denoted by n_1 and n_2, respectively. We have:

$$n_1 = \arg\min_{i \in V} \|x - w_i\|, \; n_2 = \arg\min_{i \in V \setminus \{n_1\}} \|x - w_i\| \tag{2}$$

Depending on the similarity (measured by Euclidean distance) between input x and these two nodes, i.e., $d_1 = \|x - w_{n_1}\|$ and $d_2 = \|x - w_{n_1}\|$, we take corresponding update actions as follows:

- If $d_1 > T_1$ (Fig. 1(a)), insert x as a new node, $V = V \cup \{x\}$.
- If $\alpha \cdot T_1 < d_1 \leq T_1$ (Fig. 1(b)), also insert x as a new node, $V = V \cup \{x\}$. Here, parameter α is a real number between 0 and 1.
- If $d_1 \leq \alpha \cdot T_1$ (Fig. 1(c)), update the weight of winner node. Here, η is the learning rate.

$$w_{n_1} = w_{n_1} + \eta \cdot (x - w_{n_1}) \tag{3}$$

- Meanwhile, if $d_1 \leq T_1$ and $d_2 \leq T_2$ (Fig. 1(d)), add an edge between the two winner nodes n_1 and n_2, $E = E \cup \{(n_1, n_2)\}$.

The graphical illustration is shown in Fig. 1. However, as the learning process proceeds, connected nodes might not be neighbors any more because of the motion of nodes. Thus we apply the similar edge aging scheme used by GNG model to deal with these obsolete edges. In this case, we associate each edge with an age value. When the age of some edge exceeds a predefined maximal age parameter τ, then remove that edge. When the edge is activated then reset its age. Let t_i denote the winning times of node i, i.e. the times that node i has become the most nearest node so far during competitive learning process, and we set the learning rate η to $1/t_i$, then the weight update Eq. (3) could also be considered as the mean value of input patterns around node i. Furthermore, the winning times of topological nodes also captures the local density information around these nodes to some extent.

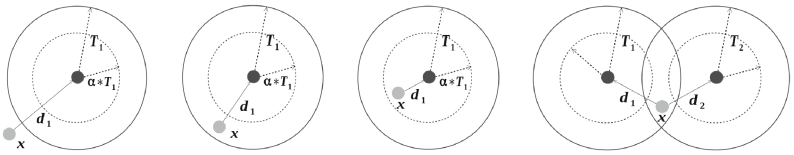


Fig. 1. (a) $d_1 > T_1$ (b) $\alpha \cdot T_1 < d_1 \leq T_1$ (c) $d_1 \leq \alpha \cdot T_1$ (d) $d_1 \leq T_1$ and $d_2 \leq T_2$

2.2 Periodic Refinement

During the incremental topology learning, we adopt the refinement process to eliminate the superfluous nodes and noisy nodes to maintain the main topology. With the assumption that noisy or outlier data is mostly distributed over the regions with lower probability density, and is sparse enough so that the main structure of target density distribution could still be recovered, most nodes created for them will have no or very few neighbors around. Let $N_k(i)$ denote the k nearest neighbors of node i, where k is determined by the maximal neighbors for all the nodes in topology, then the average winning times of its neighbors is defined as:

$$t(N_k(i)) = \frac{1}{|N_k(i)|} \sum_{j \in N_k(i)} t_j \tag{4}$$

Every λ learning iterations, i.e. λ input examples, we find the nodes which have no neighbor or only one neighbor. Based on the presumption that such nodes lie in the low-density area, we eliminate those nodes whose winning times is lower than the averaged winning times of its neighbors to a certain degree $\beta \cdot t(N_k(i))$. Here, λ is a positive integer, and β is a real number between 0 and 1. They are both user defined parameters. By this periodic refinement, we make the learned topology more concise and more robust. As a summary, we give the complete incremental topology learning algorithm in Algorithm 1.

Algorithm 1. Incremental Topology Learning

Input: density factor α, maximal age τ, refinement parameters λ and β.
Output: graph topological structure $G = (V, E)$, where V is the learned node set and E is the edge set between nodes.

1: Initialize node set $V = \phi$, edge set $E = \phi$.
2: Input two patterns, add them to the node set V.
3: Input a new pattern x.
4: Find the nearest nodes n_1 and n_2 of x according to Eq. (2).
5: Calculate the distance between x and n_1, n_2, take corresponding actions described in Sect. 2.1, and update weight of n_1 according to Eq. (3) when necessary.
6: If n_1 and n_2 are connected by an edge, set the age of this edge to zero. If such an edge does not exist, create it and add it to edge set E when necessary.
7: Increase the age of all edges emanating from n_1, remove edges with an age larger than τ. If this results in nodes having no connecting edges, remove them as well.
8: Update the similarity threshold of node n_1 according to Eq. (1).
9: If the number of patterns inputed so far is an integer multiple of parameter λ, refine current topology as described in Sect. 2.2 according to Eq. (4).
10: Go to step 3 to continue the incremental learning process.

2.3 Weighted Neighbors Analysis

When the topology learning finished, we obtain a concise graph representation $G = (V, E)$ of the original training data. We then construct the k-nearest neighbor (KNN) graph based on the learned topology. The choice of k can be made as the maximal number of neighbors of learned nodes in the topology. The original one class nearest neighbor method takes local density into account [1]. It can be described as: the distance from object x to its nearest neighbor in the training set $NN(x)$ is compared with the distance from this nearest neighbor $NN(x)$ to its nearest neighbor $NN(NN(x))$, and the dissimilarity measure between test object x and the target class X is defined as:

$$d(x, X) = \frac{\|x - NN(x)\|}{\|NN(x) - NN(NN(x))\|} \tag{5}$$

Inspired by the nearest neighbor classifier, for each node i in the learned topology, we define the average distance to its topological neighbors $N(i)$ as:

$$d(i, N(i)) = \frac{1}{|N(i)|} \sum_{j \in N(i)} \|w_i - w_j\| \tag{6}$$

Note that we have constructed the KNN graph, then $|N(i)|$ is equal to k for every node i. Given test object x, the local dissimilarity score $d(x, X)$ to the target class is defined as the average distance ratio for all its k-nearest neighbors $N_k(x)$. Here, k is also the maximal number of neighbors of learned nodes in the topology. Since some nodes are near the distribution boundary, while the others are deep within the region with high distribution density. To indicate different contribution of nodes for our neighbors analysis, we incorporate a weight scheme to get a more meaningful score. Thus we have:

$$\alpha_i = \frac{t_i}{\sum_{j \in N_k(x)} t_j}, \ i \in N_k(x) \tag{7}$$

$$d(x, X) = \frac{1}{|N_k(x)|} \sum_{i \in N_k(x)} \alpha_i \cdot \frac{\|x - w_i\|}{d(i, N(i))} \tag{8}$$

where t_i is winning times of node i, and in some sense it captures the local density information around node i as we stated before. The dissimilarity score $d(x, X)$ is local in the sense that only a restricted neighborhood of each object is taken into account.

This neighbors analysis scheme is comparable but not identical to the Local Outlier Factor(LOF) method [9]. Finally, the decision whether x belongs to the target or outlier class is based on the score compared with threshold θ which is adjusted to accept or reject a user-defined fraction of the target class [1]. In summary, to apply our method to the OCC problems, firstly we perform incremental topology learning for training on the large scale target data or the target data in stream-oriented applications. When testing is performed, we constructed the

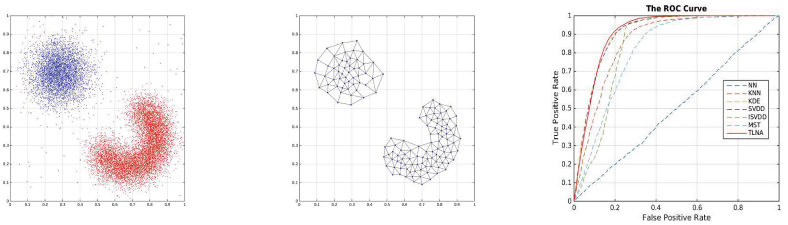
KNN graph based on the learned topology. After that, the weighted neighbors analysis technique is utilized to construct the one class classifier. Although it seems that our method is not a pure incremental learning method, it is effective enough for the real large scale OCC problems.

3 Experiments

In this section, we conduct experiments on an artificial data set and some widely used real world data sets, also present detailed experimental results for TLNA in comparison with contemporary one class classifiers. In our comparative experiments, the value of k for the KNN method and value of width parameter for the KDE method is optimized. The Gaussian kernel is used as kernel function in the SVDD methods. For OCC problems, usually Receiver Operating Characteristic (ROC) curve is used as the performance measurement of classifier. In our experiment results, the Area Under the Curve (AUC) for the ROC is also calculated and compared. The parameters of proposed method in all experiments are set as follows: $\alpha = 0.5$, $\tau = 50$, $\lambda = 50$, $\beta = 0.5$.

3.1 Artificial Data

For the artificial dataset, the training data consists of three different distributions: Gaussian distribution, banana-shaped distribution and random noisy data following the uniform distribution. Both Gauss and banana distributions are considered as target class, and others outlier class. Figure 2 shows the plots of the generated dataset, the learned topology structure and the experiment result. We compare the performance of TLNA with contemporary one class classifiers including NN(Neatest Neighbor), KNN(k-Neatest Neighbor), KDE(Kernel Density Estimation), SVDD(Support Vector Data Description), ISVDD(Incremental SVDD) [4] and MST(Minimum Spanning Tree) [8]. The result shows that TLNA can obtain comparable performance with the state-of-the-art methods SVDD and KDE, and preferable performance than others for this synthetic data set.



(a) (b) (c)

Fig. 2. Artificial dataset and experiment result (a) target training data with noise (b) learned topology structure (c) ROC curve compared with contemporary OCC methods

3.2 Real World Data

For the case of real world data sets, since there is no special data set for one class classifiers, we use the conventional classification data sets. Most of them were originally collected from the UCI machine learning repository [10], and some data sets were collected in the LIBSVM data sets [11]. We divide each data set into two separate class group. One class group is used as target class and the other outlier class. Half of randomly selected data from the target class is used as the training target data, and the rest testing target data. To demonstrate the robustness to noises or outliers of OCC methods, we also add 5 % examples from outlier class to the training target data to construct the noisy target training data. Equal amount of targets and outliers are put together to construct the testing data. To make sure that the achieved results are not coincidental, this scheme is repeated for 20 times to build 20 different training and testing sets. And the final result is achieved by averaging over the best 10 models for each classifier. The result is shown in Table 1.

Table 1. Average AUC of each method for the real world data sets (best method in **underline bold**, second best **bold)**

Data set	TLNA	NN	KNN	KDE	SVDD	ISVDD	MST
Skin	**98.94 ± 0.27**	66.81 ± 0.60	74.70 ± 0.35	95.70 ± 0.36	**98.34 ± 0.28**	81.84 ± 0.21	72.71 ± 0.32
Svmguide1	**96.91 ± 0.57**	58.36 ± 1.07	95.83 ± 0.16	**97.58 ± 0.05**	96.63 ± 0.51	94.11 ± 0.37	93.96 ± 0.23
Cod-rna	**85.95 ± 1.32**	52.94 ± 0.25	79.75 ± 0.14	82.44 ± 0.26	**87.10 ± 0.26**	79.50 ± 0.31	77.25 ± 0.22
Shuttle	**98.79 ± 0.28**	58.75 ± 1.03	68.29 ± 0.30	78.91 ± 0.45	**98.32 ± 0.20**	97.07 ± 0.22	65.29 ± 0.39
Pendigits	**96.08 ± 0.43**	53.23 ± 0.25	94.20 ± 0.47	94.89 ± 0.31	**95.01 ± 0.40**	94.95 ± 0.42	92.23 ± 0.37
Connect4	**77.69 ± 0.29**	58.62 ± 0.11	60.24 ± 0.12	**88.74 ± 0.18**	61.47 ± 0.25	59.64 ± 0.24	66.90 ± 0.20

The experimental results on the real world datasets show that our approach TLNA can obtain comparable or preferable performance compared with the best method in most datasets. Particularly, for the *pendigits* dataset, we use several digits as target class and others as outliers, and TLNA achieves better performance than all the other methods. It verifys that our approach can deal with complex data distribution and is robust to noises or outliers in the target training data.

4 Conclusion

In this paper, we propose an incremental learning framework for OCC problems in large scale data environment. It can automatically acquire a concise graph representation of the target class by incremental topology learning. Based on the graph representation, we build a description model for the target class. As a result of the incremental characteristic, our method need not to store any historical data but the learned graph model. Therefore, the time and space complexity of our method are much smaller than the traditional batch OCC methods. Furthermore, The periodic refinement operation in the topology learning process

makes it robust to noises and outliers. Experimental results on artificial and real world data sets confirmed the validity of proposed learning method. The future works may include extension to other one class learning models based on the learned graph representation of original input data.

Acknowledgement. The authors would like to thank the anonymous reviewers for their time and valuable suggestions. This work is supported in part by the National Science Foundation of China under Grant Nos. (61375064, 61373001) and Jiangsu NSF grant (BK20131279).

References

1. Tax, D.M.J.: One-class classification: concept-learning in the absence of counterexamples. Ph.D. thesis, Delft University of Technology (2001)
2. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Mach. Learn. **54**(1), 45–66 (2004)
3. Scholkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)
4. Tax, D.M.J., Laskov, P.: Online SVM learning: from classification to data description and back. In: Proceedings of the 13th IEEE NNSP Workshop. IEEE Computer Society Press, Los Alamitos (2003)
5. Martinetz, T.M.: Competitive Hebbian learning rule forms perfectly topology preserving maps. In: International Conference on Artificial Neural Networks, pp. 427–434 (1993)
6. Fritzke, B.: A growing neural gas network learns topologies. In: Proceedings of the Advances in Neural Information Processing Systems, pp. 625–632 (1995)
7. Shen, F., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Netw. **19**(1), 90–106 (2006)
8. Juszczak, P., Tax, D.M.J., Duin, R.P.W.: Minimum spanning tree based one-class classifier. Neurocomputing **72**(7), 1859–1869 (2009)
9. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104 (2000)
10. Asuncion, A., Newman, D.: UCI Machine Learning Repository. University of California, Irvine (2007)
11. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 389–396 (2011). http://www.csie.ntu.edu.tw/cjlin/libsvm