19th International Computer Science and Engineering Conference (ICSEC)
Chiang Mai, Thailand, 23-26 November, 2015

Training Set Size Reduction in Large Dataset Problems

Varin Chouvatut*, Wattana Jindaluang and Ekkarat Boonchieng
The Theoretical and Empirical Research Group
Center of Excellence in Community Health Informatics
Department of Computer Science
Faculty of Science, Chiang Mai University
Chiang Mai, Thailand
varinchouv@gmail.com (*corresponding author), wjindaluang@gmail.com, ekkarat@boonchieng.net

***Abstract*—Classifiers have known to be used in various fields of applications. However, the main problem usually found recently is about applying a classifier to large datasets. Thus, the process of reducing size of the training set becomes necessary especially to accelerate the processing time of the classifier. Concerning the problem, this paper proposes a new method which can reduce size of the training set in a large dataset. Our proposed method is improved from a famous graph-based algorithm named Optimum-Path Forest (OPF). Our principal concept of reducing the training set's size is to utilize the Segmented Least Square Algorithm (SLSA) in estimating the tree's shape. From the experimental results, our proposed method could reduce size of the training set by about 7 to 21 percent comparing with the original OPF algorithm while the classification's accuracy decreased insignificantly by only about 0.2 to 0.5 percent. In addition, for some datasets, our method provided even as same degree of accuracy as of the original OPF algorithm.**

Keywords—Optimum-Path Forest ; Training Set Size Reduction; Graph-based Classification Algorothm; Supervised Learning

I. INTRODUCTION

Recently, information of interest to be analyzed are often of a large size, for example, there are usually millions of pixels involved in the field of image processing, there can be millions of users or clients connecting to a computer network, a hospital always has hundreds of thousands of patients each of which can have hundreds of attributes to be recorded, etc. What we need to consider in data analysis is not only about the accuracy obtained from a classifier, but also the speed or processing time of the classifier, especially, in a real-time application where processing time of algorithm used is more vital than its accuracy.

Graph-based classification algorithm generally works by representing the interesting data with vertices and the relationships to be compared among the data with edges of a graph in the sample feature space. With this representation, every pair of vertices in the graph is connected by a sole edge whose weight is defined using the Euclidian distance between them. In this way, the generated graph is thus called a complete graph.

A lot of researches used graph-based classification as in [1] where the researchers proposed a classifier with Supervised Lazy Random Walk. Their results showed that their method gave high degree of accuracy and still worked well with noise-added data. Another famous research of graph-based classifier is OPF which is demonstrated by researchers of [2] that the OPF algorithm has several advantages which include (i) the algorithm is free from parameters, (ii) the process of training phase has no classification errors, (iii) the algorithm does not pose the over-fitting problem, and (iv) the algorithm can be used in problems of multi-class data (additional concepts of OPF are explained in Section II). Afterwards, there were several researches based on OPF such as [3-5]. Researchers in [3] studied a robust set of pattern classifiers based on OPF. From their experiments, their method could tolerate noises and also could cope with overlapping classes. Additionally, they claimed that their method worked better than a well-known classifier, Support Vector Machines (SVMs). In [4], researchers revised the original OPF by altering (i) the cost function in paths, for example, f_{max} or f_{min}, (ii) methodology of adjacency relation such as complete graph or k-nn, and (iii) methodology of prototype estimation such as Minimum Spanning Tree (MST) or maxima regions of the data feature space. The researchers compared their method with SVM and Artificial Neural Networks using Multilayer Perceptrons (ANN-MLP). Comparing results showed that their method suited a huge dataset while its accuracy might be higher or lower than SVM, depending on the data in question. However, their method always gave higher accuracy than ANN-MLP. Also, they claimed that their method processed rapidly, had straightforward implementation, could cope with multi-class datasets, and required no assumption of class shapes. Later, researchers of [5] extended the OPF algorithm to a k-Optimum Path Forest (k-OPF) algorithm and compared the new algorithm with a number of popular classification algorithms including k-Nearest Neighbors (k-NN), SVM, and Decision Tree (DT). Their experiments showed that the k-OPF algorithm provided as same degrees of accuracy and decision boundaries as k-NN algorithm when using all samples as Prototypes but the k-OPF algorithm gave results differed from SVM and DT. The researchers found that their k-OPF processed more rapidly than the k-NN and they also proved that k-OPF and k-NN algorithms have the same error bounds.

978-1-4673-7825-3/15/$31.00 ©2015 IEEE

Apart from that, the OPF algorithm was applied in many kinds of applications such as in [6] and [7]. In [6], the algorithm was applied with an IRIS database. From their experiments, processing time and accuracy were compared amongst using OPF algorithm, Hamming classifier, and Bayesian classifier. The researchers found that Hamming classifier gave the highest accuracy but also consumed the longest running time whereas OPF algorithm and Bayesian classifier gave similar accuracy but OPF processed faster than Bayesian for about 385 times. As well, the OPF algorithm was applied in analysis of electrocardiogram (ECG) signals [7]. The experimental results demonstrated the robust performance of the OPF algorithm and also showed that OPF gave better running time and higher accuracy comparing to SVM and MLP.

In this paper, we proposes a new supervised learning approach for accelerating a classifier by reducing size of the training set. The proposed classification method is an improvement on OPF algorithm. The improved classification approach can speed up classifiers with insignificant degree of a drop in accuracy from the original classification.

Contents are separately explained in sections. An overview of OPF algorithm is explained in Section II. Our proposed approach is described in Section III. Experimental setup and results are shown in Section IV. Lastly, our conclusion is presented in Section V.

II. Overview of OPF Algorithm

OPF algorithm is a graph-based classifier whose process may be divided into 2 phases, training and testing. The training phase is to select Prototypes out of a training set where selecting a Prototype can be done by finding a MST. Then, Prototypes and the training set will be combined to create a forest using the OPF algorithm. Next, the testing phase will use the forest created from the training phase to predict class of the new coming data. Let $Z = \{x_1, x_2, \ldots, x_n\}$ be a training set with the size of n and $Y = \{y_1, y_2, \ldots, y_n\}$ be class labels where y_i is the class label for data x_i, an x_i will have m features, saying $x_i = \{f_{i1}, f_{i2}, \ldots, f_{im}\}$. The distance between any pair of different points is computed using the Euclidean distance. The two phases of the OPF algorithm are explained more in details as follows.

A. Training Phase

It is well-known that the training phase starts from selecting Prototypes. At this phase, a complete graph will be generated from all data items of a training set. Each edge in the graph will be weighed by the distance between its pair of points. Then, a MST for the graph can be created by either Kruskal or Prim method. The Prototypes will be selected from vertices which are end points of edges in MST connecting data from different classes. Next step is to combine the selected Prototypes with the rest data of the training set in order to be trained with the OPF algorithm.

In [3], researchers used f_{max} to be the path-cost function defined in Eq. (1).

$$f_{max}(v) = \begin{cases} 0 & \text{if } v \in S, \\ \max_{e \in P}\{\text{weight}(e)\} & \text{where P is a path} \\ & \text{from } s \in S \text{ to } v. \end{cases} \quad (1)$$

B. Testing Phase

Testing phase is to predict class of the new coming data items which can be done by creating edges from the new coming data items to all existing data items in the training set in a forest. And each newly created edge's weight will be set by the distance between the new data item and the existing data item being connected from the training set.

Given $x_{new} = (f_1, f_2, \ldots, f_m)$ be a new coming data item, the OPF algorithm can compute the cost from x_{new} to all Prototypes using the function f_{max} as displayed in Eq. (2).

$$C(x_{new}) = \min\{\max\{C(s), d(s, x_{new})\}\} \text{ for all } s \in Z. \quad (2)$$

Given $s^* \in Z$ be the data item given minimum value of $C(x_{new})$, the OPF algorithm will predict the class of x_{new} as same as of s^*.

Additional details of this OPF algorithm such as its Pseudocodes or examples may be found in [2-3] and [5].

III. Proposed Approach

As explained in Section II, in the testing phase, we need to compute the distance from the new data item to all existing data items in the training set; so if we want to accelerate the classifiers, we need to reduce size of the training set. In this paper, we thus proposed a new method which can be used to reduce the set's size. Our method can be considered as an improved OPF algorithm, that is, we used the forest generated from the traditional OPF algorithm then reduced its size before sending it as input to the testing phase. Consequently, the output from our method is a training set with smaller size than usual. After that, the smaller training set will be sent further to the testing phase to predict class of a new data item as usual.

For the purpose of size reduction, we cannot just remove a subset of data items from the training set directly. Doing so, shape of the tree may be distorted resulting in classification's accuracy drop. Our purpose of reducing size of the training set is thus to reduce the size whereas the classification's accuracy drops only slightly. To achieve the objective, we thus use Segmented Least Square Algorithm (SLSA) to estimate shape of the tree obtained from the training phase of the OPF algorithm. Without loss of generality, we assumed that the forest resulted from the OPF algorithm contains only one tree. If the algorithm returned a forest of more than one tree, each tree will be manipulated one by one.

The Segmented Least Square (SLS) problem is a minimization problem in which a set of points is given and then lines best fit for all of the points need to be searched. One method provided a trivial solution is to create a line connecting each pair of contiguous points together, this way all points will be fitted perfectly. It is unfortunate that in the real-world application we cannot doing so since cost penalty must be paid

for each line addition. Definition of the SLS problem was explained in [8] that given a set of n points, $P = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)\}$ such that $x_1 < x_2 < \ldots < x_n$, we need to divide set P into segments each of which represents an interval of the points, that is, $S = \{p_i, p_{i+1}, \ldots, p_j\}$ for some $i \le j$. For each segment S, we then compute a best-fit line whose efficiency is measured using Eq. (3).

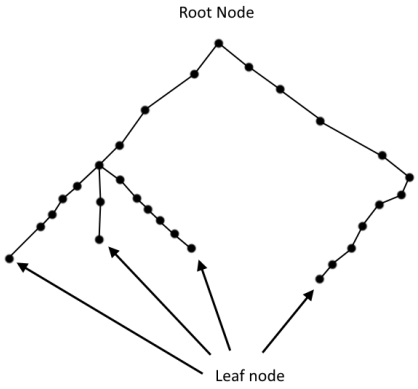
$$e_{ij} = \sum_{k=i \text{ to } j} (f(x_k) - y_k)^2 . \quad (3)$$

Finding a solution for the SLS problem may be done by dynamic programming technique whose key concept for this problem is a recurrence in Eq. (4).

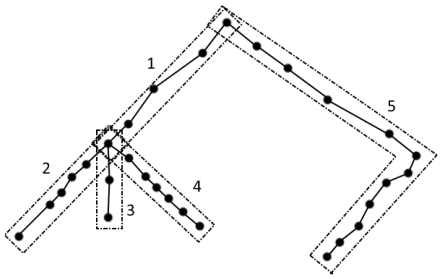
$$OPT(j) = \begin{cases} \min_{1 \le i \le j} \{e_{ij} + C + OPT(i-1)\} & ; j > 1, \\ 1 & ; j = 1. \end{cases} \quad (4)$$

where OPT(j) is the optimal solution for the point interval from p_1 to p_j, e_{ij} is the minimum error in line fitting of the segment $S = \{p_i, p_{i+1}, \ldots, p_j\}$, and C is the (user-defined) penalty paid for each line addition. So finding a solution for the SLS problem is to calculate the OPT(n) where the base case, OPT(0), is set to 0.

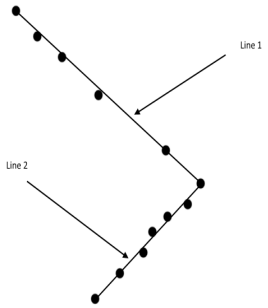
Here, we used SLSA to estimate the tree's shape. Firstly, the tree obtained from the OPF algorithm would be partitioned into branches. Next, shape estimation for each branch in the tree would be done by SLSA. From line estimation fitted by the SLSA, a new set of points along each fitting line together with their new cost would then be redefined. Finally, all newly estimated branches would be assembled back to reform the tree. Fig.1 shows an example resulted from the explained method for estimating a new tree using SLSA.



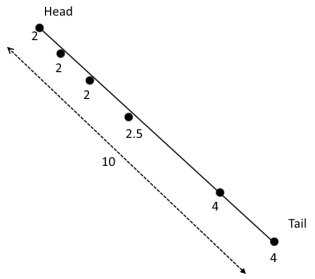
(a) The tree output from OPF algorithm.



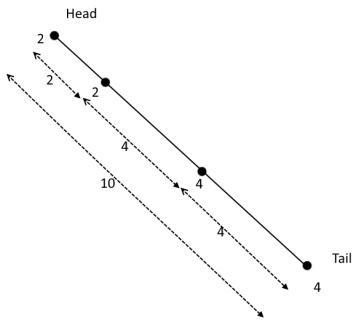
(b) The tree partitioned into five branches.



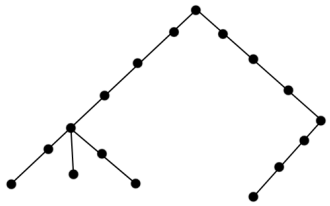
(c) Two line segments estimated by SLSA manipulating on the 5th branch of (b).



(d) The 1st line segment with points from (c) whose distance from end (Head) to end (Tail) is assumed to be 10.



(e) The line segment from (d) with newly defined points and their cost.



(f) The newly estimated tree to be used as the training set with smaller size.

Fig. 1. An example result of newly estimated tree from the proposed method.

From Fig. 1(f), the number of points in the newly estimated tree to be used as the training set is obviously less than the original tree, Fig. 1(a), obtained from the OPF algorithm whereas the tree's shape remains similarly. This demonstrated that our proposed method can reduce size of the training set while the classification's accuray drops only slightly.

IV. Experimental Settings and Results

We compared our proposed method with the traditional OPF algorithm on five datasets retrieved from UCI [9] and Machine Learning Data Set Repository [10] databases, in terms of size of the training set and accuracy from classification. Characteristics of the datasets are given in Table I.

TABLE I. DATASET CHARACTERISTICS

Dataset Name	Dataset Size	The Number of Features	The Number of Classes
Banana	5,300	2	2
E. coli	336	7	8
Ionosphere	351	34	2
Letter Recognition (LR)	20,000	16	26
Sonar	208	60	2

As we noted that the penalty paid for each line addition, C, in Eq. (4) is a user-defined parameter, thus, we experimented with three value settings of C for 0.125, 0.25, and 0.5, respectively. For the minimum error, e_{ij}, calculated from Eq. (3), each feature of the data was normalized to be between 0 and 1. To evaluate the performance of the two methods (the traditional OPF and the proposed method), five-fold cross validation has been used. Comparisons with respect to different C's (the penalty) of the classification's accuracy and the training set's size between our proposed method and the traditional OPF algorithm are as Tables II - IV. Note that, due to the five-fold cross validation, size of a training set is its average size.

For C equal to 0.125 in Table II, the classification's accuracy obtained from our proposed method decreases insignificantly, i.e. for 0.51, 0.37, and 0.24 percent in the datasets named Banana, E. coli, and Letter Recognition (LR), respectively. For the datasets Ionosphere and Sonar, the accuracy obtained from both our method and the traditional OPF algorithm is the same. In addition, sizes of the training sets are reduced by 21.71, 7.44, 6.98, 12.89, and 9.13 percent in datasets Banana, E. coli, Ionosphere, LR, and Sonar, respectively using our method.

TABLE II. ACCURACY OF CLASSIFICATION AND TRAINING SET SIZE FOR C EQUAL TO 0.125

Dataset Name	Traditional OPF		Proposed Method	
	Accuracy	*Training set size*	*Accuracy*	*Training set size*
Banana	70.08	4,240.0	69.72	3,316.4
E. coli	76.69	268.8	76.40	248.8
Ionosphere	84.33	280.8	84.33	261.2
LR	95.09	16,000.0	94.86	13,937.4
Sonar	61.48	166.4	61.48	151.2

TABLE III. ACCURACY OF CLASSIFICATION AND TRAINING SET SIZE FOR C EQUAL TO 0.25

Dataset Name	Traditional OPF		Proposed Method	
	Accuracy	*Training set size*	*Accuracy*	*Training set size*
Banana	70.08	4,240.0	69.72	3,316.2
E. coli	76.69	268.8	76.40	248.0
Ionosphere	84.33	280.8	84.33	261.0
LR	95.09	16,000.0	94.88	13,936.2
Sonar	61.48	166.4	61.48	150.8

As shown in Table III with C equal to 0.25, our method reduces accuracy of the classification by only 0.51, 0.37, and 0.22 percent in datasets Banana, E. coli, and LR, respectively whereas the accuracy remains the same as of the traditional OPF algorithm for the datasets Ionosphere and Sonar. Accuracy values obtained from setting C to 0.25 shown in Table III are very similar to those obtained from setting C to 0.125 shown in Table II. With C equal to 0.25, sizes of the training sets are reduced by 21.79, 7.44, 7.05, 12.89, and 9.13 percent in datasets Banana, E. coli, Ionosphere, LR, and Sonar, respectively.

TABLE IV. ACCURACY OF CLASSIFICATION AND TRAINING SET SIZE FOR C EQUAL TO 0.5

Dataset Name	Traditional OPF		Proposed Method	
	Accuracy	*Training set size*	*Accuracy*	*Training set size*
Banana	70.08	4240.0	69.72	3316.2
E. coli	76.69	268.8	76.40	248.0
Ionosphere	84.33	280.8	84.33	261.0
LR	95.09	16000.0	94.86	13936.2
Sonar	61.48	166.4	61.48	150.8

Table IV with C equal to 0.5 demonstrates that our method still provides good performance of the classification for datasets Ionosphere and Sonar whereas accuracy values of the classification for the rest datasets decrease for just small amounts, which are 0.51, 0.37, and 0.25 percent for Banana, E. coli, and LR, respectively. The training sets for datasets Banana, E. coli, Ionosphere, LR, and Sonar use smaller sizes by 21.82, 7.74, 7.34, 12.90, and 9.62 percent, respectively than usual.

V. CONCLUSION

Our paper proposes a new method aiming to reduce time consumption of a classifier by reducing size of the training set in a large dataset problem. To reduce the training set's size, SLSA is used to estimate shape of a tree generated from the training phase of OPF algorithm. The training set with reduced size obtained from the shape-estimated tree is thus sent to the testing phase in order to predict class of the new input data. Our experiments showed that sizes of the training sets were reduced by 7 - 21 percent comparing with the traditional OPF algorithm. Also, the classification's accuracy remains the same as of the traditional algorithm or decreases (if any) with an insignificant degree. Lastly, a higher penalty paid, C, gave a smaller size of the training set.

REFERENCES

[1] L. Lu, X. Xu, P. He, Y. Ma, Q. Chen and L. Chen, "Supervised Lazy Random Walk calssifier", 10th Web Information System and Application Conference (WISA), Yanzhou, China, 2013.

[2] J.P. Papa, A.X. Falcao, C.T. Suzuki and N.D. Mascarenhas, "A discrete approach for supervised pattern recognition", IWCIA 2008, LNCS 4958 Springer Heidelberg, pp: 136 – 147.

[3] J.P. Papa, A.X. Falcao, P.A. Miranda, C.T. Suzuki and N.D. Mascarenhas, "Design of robust pattern callsifiers based on Optimum-path Forests", 8th International Symposium on Mathematical Morphology, Rio de Janeiro, Brazil, 2007.

[4] J.P. Papa and A.X. Falcao, "Optimum-Path Forest: a novel and powerful framework for supervised graph-based pattern recognition techniques", Congresso da SociedadeBresileire de Computacao, Bento Goncalves, 2009.

[5] R. Souza, L. Rittner and R. Lotufo, "A comparision between k-Optimum Path Forest and k-Nerest Neighbors supvervise classifiers", Pattern Recognition Lets. 39 (2014):2 – 10.

[6] L.C. Afonso, J.P. Papa, A.N. Marana, A. Poursaberi and S.N. Yanushkevich, "A fast large scale iris database classification with Optimum-Path Forest technique", International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, 2012.

[7] E.J. Luz, T.M. Nunes, V.H. Albuquerque, J.P. Papa and D. Menotti, "ECG arrhythmia classification based on Optimum-Path Forest", Expert Systems with Applications 40(2013):3561 – 3573.

[8] J. Kleinberg and E. Tardos, Algorithm Design. Person Addision Wesley, Boston, 2006.

[9] UCI Database, https://archive.ics.uci.edu/ml/datasets.html (searched in Jul. 2014).

[10] Machine Learning Data Set Repository, http://mldata.org/repository/data/ (searched in Jul. 2014).