ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 26-28 April 2017, i6doc.com publ., ISBN 978-287587039-1.
Available from http://www.i6doc.com/en/.

Mutual information for improving the efficiency of the SCH algorithm

D. Fernandez-Francos[1], O. Fontenla-Romero[1], A. Alonso-Betanzos[1] and G. Brown[2] *

1- Department of Computer Science, University of A Coruna, Spain

2- MLO Group, University of Manchester, United Kingdon

Abstract.

A new approach for improving the efficiency of a one-class classification algorithm making it more suitable for big datasets is presented in this work. The original algorithm, called SCH algorithm, approximates a D-dimensional convex hull decision by means of random projections and an ensemble of 2-dimensional decisions. With this new approach we try to get rid of the less relevant projections that lead to bad classification models in the low dimensional space. For that, after the training phase, a new stage based on mutual information is added to the original algorithm in order to remove the unnecessary projections, maintaining the information contained in the model (and thus the accuracy) while making it lightweight. This reduces remarkably the computational time of the testing phase. Finally, some experimental results demonstrate the effectiveness and efficiency of these approach.

1 Introduction

One-class classification is [1] an unsupervised classification task for which only information of one class (target class) is available for learning. Nowadays the continuous collection of data from many different sources leads to Big Data problems that are difficult to handle by the majority of one-class classification algorithms. In this work a new method for adapting the SCH (Scaled Convex Hull) classification algorithm [2] to these kind of problems is presented. This SCH algorithm is an improved version of the one-class method proposed by Casale et al. [3]. It is characterized by the use of the convex hull (CH) [4] to define the boundary of the target class in a one-class classification problem. However, calculating the CH and checking whether a point lies inside it in high dimensional spaces is computationally hard. So, in order to solve this problem, the decision made by the CH in the original D-dimensional space is approximated by an ensemble of τ randomly projected decisions made on 2-dimensional spaces. The number of Random Projections is difficult to establish and it must be high if we want to be sure that the original model is well approximated by the ensemble of 2 dimensional decisions. Besides, this algorithm uses the concept of scaled convex hull presented by Liu et al. [5] to calculate reduced/enlarged versions of the original CH in the low dimensional space, for the sake of avoiding over-fitting and finding

*This research has been financially supported in part by the Spanish Ministerio de Economía y Competitividad (TIN2015-65069-C2-1-R), by European Union FEDER funds and by the Consellería de Industria of the Xunta de Galicia (GRC2014/035).

the optimal operating point of the classifier. Vertices are defined with respect to the expansion parameter $\lambda \in [0, +\infty)$ as in $\overline{v}^{\lambda} : \{\lambda \overline{v_i} + (1-\lambda)\overline{c}\}, i = 1 \ldots N$, where $\overline{c} = Pc$ represents the projection of the center c given a random projection matrix P, $\overline{v_i}$ is the set of CH vertices of the projected data. In addition, using this formula, three different definitions of CH "center" can be used: 1) Average of all the points in the projected space $\overline{c} = \frac{1}{N}\sum_i \overline{x_i}, \forall \overline{x_i} \in \overline{S_t}$, 2) Average of the CH vertices in the projected space $\overline{c} = \frac{1}{N}\sum_i \overline{v_i}, \forall \overline{v_i} \in Conv(\overline{S_t})$,and 3) Centroid. This provides flexibility to the algorithm as each kind of center leads to different decision regions. The learning and testing procedures of the SCH method can be seen in Algorithms 1 and 2, respectively. In Algorithm 2 the test point is projected into the low dimensional space spanned by the t-th projection matrix.

Algorithm 1 Learning algorithm

```
Input: Training set $S \subset R^d$; Number of projections $\tau$; Expansion parameter $\lambda$; Type of center $tc$.
Output: The ensemble model $E_\tau$ composed of $\tau$ projection matrices and their expanded CH vertices.
$E_0 = \phi$;
for $t = 1..\tau$ do
    $P_t \sim N(0,1)$ % Create a random projection matrix $[2 \times d]$;
    $\overline{S_t} : \{P_t x | x \in S\}$ % Project original data;
    $\{v_i\}_t = \mathrm{CH}(\overline{S_t})$ % Return the vertices of the CH;
    $\overline{c} = getCenter(tc, S)$ % Return the selected center in the low dimensional space.
    $v_t^{\lambda} : \{\lambda v_i + (1-\lambda)\overline{c} | \, v_i \in \{v_i\}_t\}$ % Expanded CH in the low dimensional space;
    $E_t = E_{t-1} \cup (P_t, v_t^{\lambda})$ % Store the vertices of the projected CH and the projection matrix;
end for
```

Then, given the set of expanded vertices, it is possible to check whether the point lies inside the projected polytope. In this scenario, the *decision rule* is the following: *a point does not belong to the modeled class if and only if there exists at least one projection in which the point lies outside the projected convex polytope.*

Algorithm 2 Testing algorithm

```
Input: Test point $x \in R^d$; Ensemble model $E_\tau$.
Output: $Result \in \{INSIDE, OUTSIDE\}$
Result = INSIDE;
for $t = 1..\tau$ do
    $\overline{x_t} : \{P_t x\}$ % Project test point;
    if $\overline{x_t} \notin \mathrm{CH}(v_t^{\lambda})$ then
        Result = OUTSIDE;
        Break;
    end if
end for
```

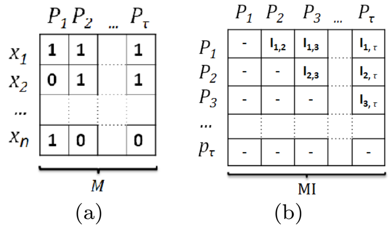
2 Projection pruning phase

In this work we propose to add a new phase to the SCH algorithm, called *projection pruning phase.* Its purpose is to reduce the number of random projections used to classify new data removing the less relevant. In order to do so we used the mutual information criterion to evaluate the set of τ projections and sort them in order of relevance. For two random variables X and Y the mutual information [6] is defined as: $I(X;Y) = H(X) - H(X|Y)$ where $H(X)$ is the entropy of X and

$H(X|Y)$ is the conditional entropy of X knowing Y. This measure is symmetric in X and Y, nonnegative and is equal to zero if and only if the variables are independent. The mutual information measures arbitrary dependencies between random variables and it is suitable for estimating their ”information content” in complex classification tasks. Algorithm 3 shows the steps followed during this phase. Firstly, τ random variables containing information about each one of the projections are needed. After the training phase (see Alg. 1) an ensemble E_τ of τ classification models in 2-dimensions is obtained. A set of n non-seen points, called validation set $V \subset R^d$, is evaluated against each one of the projected models P_i. The result is a matrix M (see Figure 1.a). Each cell shows the result of checking whether the point $x_i \in V, i = 1 \ldots N$ is inside (1) or outside (0) the projected CH created by the projection $P_i, i = 1 \ldots \tau$. Thereafter, we considered each column i of the matrix M as a random variable containing information about the projection P_i. Thus, the mutual information against each pair of variables can be calculated and the upper triangular matrix MI is the result. Notice that the class of each validation sample is not needed, just the result of the classification. As can be seen in Figure 1.b, the diagonal and the

Algorithm 3 Projection pruning phase

```
Input: Model $E_\tau$ provided as output of Alg. 1; Validation set $V \subset R^d$.
Output: A vector $L$ containing the position of $\tau$ projection matrices from less to more relevant.
$L = \phi$; $L_{max} = \phi$;
$M = \phi$; % Classification results matrix $[n \times \tau]$
for $t = 1..\tau$ do
    $\overline{V_t} : \{P_t x | x \in V\}$ % Project original validation data;
    for $i = 1..n$ do
        $M(i,t) = 0$;
        if $\overline{x_i} \in$ CH($v_t^\lambda$) then
            $M(i,t) = 1$;
        end if
    end for
end for
$MI = \phi$; % Mutual information matrix $[\tau \times \tau]$
for $i = 1..\tau$ do
    for $j = i + 1..\tau$ do
        MI(i,j) = I(M(:,i);M(:,j)); % Mutual information between projection $i$ and projection $j$.
    end for
end for
for $i = 1..\tau$ do
    $L_{max}(i) = MAX(MI(i,:))$; % Vector containing the maximum value of each row of $MI$
end for
$[L_{max}, L] = SORT(L_{max}, \text{“}ascend\text{”})$; % Sorted list $L_{max}$ and index $L$ of the new positions.
```



	P_1	P_2	...	P_τ
x_1	1	1		1
x_2	0	1		1
...				
x_n	1	0		0

M

(a)

	P_1	P_2	P_3	...	P_τ
P_1	-	$I_{1,2}$	$I_{1,3}$		$I_{1,\tau}$
P_2	-	-	$I_{2,3}$		$I_{2,\tau}$
P_3	-	-	-		$I_{3,\tau}$
...					
p_τ	-	-	-		-

MI

(b)

Fig. 1: (a) Validation results matrix $[n \times \tau]$ (b) Mutual information matrix $[\tau \times \tau]$

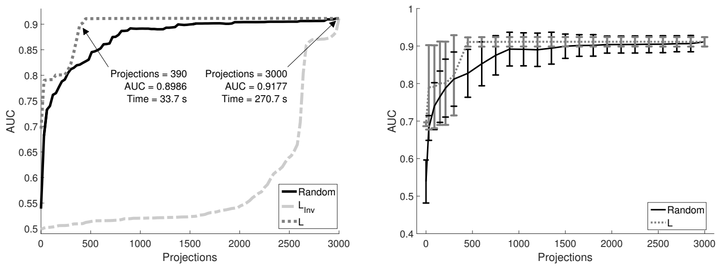
lower triangular part were not calculated because the mutual information values of one projection against itself are not useful here and the lower triangular part is not necessary because of the mutual information symmetry property. Then, the maximum value of mutual information in each row of MI is stored in a vector L_{max}, where the first value contains the maximum mutual information for P_1, the second one the maximum for P_2 and so on. Finally, the vector L_{max} is sorted in ascending order and the index containing the new positions of the projections is stored in L. This index is a rank of the relevant projections, from less to more relevant. Following L, the less relevant projections can be removed from the model making the ensemble model E_τ lightweight and reducing the computational time of Algorithm 2.

3 Experimental results

In this work, 3 high dimensional datasets were employed in order to assess the savings in computational complexity and storage space obtained with the proposed method. The first one is a reduced version of the KDD Cup 99 dataset [7] that contains over 800000 samples split in two subsets, one for training and the other for testing. This small version has only 6 features, as in [8] those were found to be the most representative. The second dataset is the MiniBooNE dataset [9]. It consist of 130064 samples divided in two classes (93565 and 36499 respectively) and 50 features. The last one is the Forest Covertype dataset [9]. It contains 581012 input patterns, each one of them composed of 54 features. The original dataset has 7 classes, but we transformed it into a one-class problem by considering all the five pine species as one class (568772 samples) and the other two species as the other (12240 samples). For the three datasets, we have considered the class with the higher number of samples as target and the other one as outlier. The number of samples and the optimal parameters of the algorithm for each problem are listed in Table 1. These parameters were previously calculated using a 10-fold cross validation on 10 different permutations of the data. Each experiment was repeated 30 times varying the training, validation and testing sets. After the training and pruning phases we have an ensemble model E_τ and a ranking L of the relevant projections. The idea is to remove the less relevant projections from E_τ maintaining the accuracy of the original method. To assess the adequacy of the pruning phase, we calculated the testing time and the Area Under the ROC Curve (AUC) for all the different ensemble models obtained removing projections in the order exhibited by L; starting with the original model (τ projections) and finishing with a model of just one projection (the last one on L). Besides, we compared this results with the ones obtained removing projections randomly and following the ranking L, but in the opposite way (L_{inv}), from more to less relevant. A pairwise t-test

Dataset	Train set	Validation set	Test set	λ	τ	Center type
KDD	250000	2000	309029	1.01	3000	Avg. v_i
MiniBooNE	60000	2000	68064	0.74	5000	Centroid
Forest Covertype	500000	2000	79013	0.88	1000	Avg. v_i

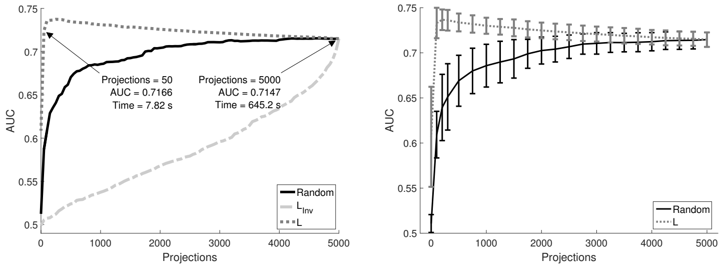
Table 1: Number of samples and parameters used for each dataset.



(a) (b)

Fig. 2: KDD results (a) AUC vs Projections (b) Avg. AUC and standard deviation (SD) for the proposed method L and random elimination.

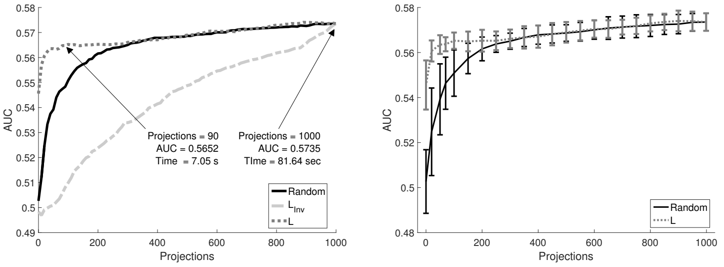
between the result obtained by the original model E_{τ} and the result obtained by each lightweight model $E_{\tau-i}$ was made to evaluate the statistical difference at 95% significance level. Figure 2.a displays the results of the KDD dataset. As can be seen, removing projections in the order indicated by L produces the best results, making it possible to obtain a model with 390 projections that there is no evidence that it is statistically significantly different as the original. This lightweight model remarkably reduces the time employed for testing, from 270 to 33 seconds. It can also be seen that eliminating projections in the opposite way (L_{inv}) produces the worst result. In Figure 2.b the mean AUC and the standard deviation for the lightweight models obtained by removing projections randomly and following the ranking L is showed. As can be seen, the results during the 30 repetitions of the experiment are much more stable when we get rid of the projections using L. Figures 3.a and 3.b display the results of the MiniBooNE dataset. Again, removing projections in the order indicated by L produces the best results. In this case an even better model than the original with just 50 projections can be obtained, reducing drastically the testing time, from 645 to 7.8 seconds. Experimental results of the Covertype dataset are showed in Figures 3.a and 3.b. In this case, the difference between the proposed method and the random elimination is less clear until the end, where a model with around



(a) (b)

Fig. 3: MiniBooNE results (a) AUC vs Projections (b) Avg. AUC and SD.

90 projections but a bit worse than the original in terms of performance can be obtained. Again, especially when the number of projections is smaller, the results are more stable when we use the proposed method L.



(a) (b)

Fig. 4: Covertype results (a) AUC vs Projections (b) Avg. AUC and SD.

4 Conclusions

In this work, a new phase, called projection pruning phase, is added to the SCH one-class classification algorithm in order to improve its efficiency when dealing with Big Data. During this stage, a rank of the relevant projections is obtained by means of mutual information. Besides, the validation dataset used in this phase does not need to be labeled, which is important in one-class problems where outlier data can be difficult to obtain. Experimental results demonstrated that the proposed approach maintains the performance of the original method with a minimum number of projection models. Besides, the savings obtained in computational time make this method more suitable for Big Data problems.

References

[1] S. S Khan and M. G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(03):345–374, 2014.

[2] D. Fernandez-Francos, O. Fontenla-Romero, and A. Alonso-Betanzos. One-class classification algorithm based on convex hull. In *ESANN 2016*, pages 477–482.

[3] P. Casale, O. Pujol, and P. Radeva. Approximate polytope ensemble for one-class classification. *Pattern Recognition*, 47(2):854–864, 2014.

[4] F. P. Preparata and M. Shamos. *Computational geometry: an introduction.* Springer Science & Business Media, 2012.

[5] Z. Liu, JG Liu, C. Pan, and G. Wang. A novel geometric approach to binary classification based on scaled convex hulls. *Neural Networks, IEEE Transactions on*, 20(7):1215–1220, 2009.

[6] T M Cover and J A Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

[7] Kdd cup 99 dataset. *http://kdd.ics.uci.edu/databases/kddcup99*, Accesed 28.02.15.

[8] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos. Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications*, 38(5):5947–5957, 2011.

[9] M Lichman. Uci machine learning repository (2013). *http://archive. ics. uci. edu/ml.*