ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

RBClust: High quality class-specific clustering using rule-based classification

Michael J. Siers & Md Zahidul Islam

School of Computing and Mathematics - Charles Sturt University
Panorama Avenue, NSW, 2795 - Australia

Abstract. Within a class-labeled dataset, there are typically two or more possible class labels. Class-specific subsets of the dataset have the same class label for each record. Class-specific clusters are the groups of similar records within these subsets. There exists many machine learning techniques which require class-specific clusters. We propose RBClust, a rule based method for finding class-specific clusters. We demonstrate that when compared to traditional clustering methods, the proposed method achieves better cluster quality, and computation time is significantly lower.

1 Introduction

A class-specific cluster (CSC) is a cluster in which each record has the same class label. Some data mining algorithms [1][2][3] require the discovery of CSCs. However, these algorithms use traditional clustering methods which are not designed for finding CSCs. Traditional clustering methods can often be slow due to high numbers of distance calculations. We propose RBClust, a fast rule-based method for finding high quality CSCs. RBClust uses a rule-based method such as C4.5 [4] to find the patterns in the dataset. Then the clusters are extracted from the rules. Then to avoid potential issues such as class imbalance, noise, and overfitting, some clusters are merged. Due to the design of RBClust, the number of distance calculations needed is much lower, so computation time is kept low. By leveraging the information of the class-labels, RBClust finds high quality CSCs. The rest of this study is organised as follows. Section 2 gives a briefing of the basic concepts and related work. In Section 3 we present RBClust. Section 4 provides a discussion of our empirical comparisons between RBClust and the chosen existing methods. Finally, in Section 5, we make our concluding remarks.

Main Contributions of this Study

- We demonstrate empirically that the proposed method can provide clusters with similar or higher quality than the existing methods.
- Computation time of the proposed method is empirically shown to be much lower than some existing methods.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

2 Related Work

Classification is the task of determining the class label L_i of a record x from a predefined set of classes $L = (L_1, L_2, \ldots, L_3)$. For example, in medical diagnosis, x can represent a patient and L could be a set of possible diagnoses. That is, $L = (\text{“}Asthma\text{”}, \text{“}Diabetes\text{”}, \text{“}Cancer\text{”}, \text{“}Healthy\text{”})$.

Classifiers are models which can intelligently assign a L_i to x. Building a classifier requires two things: a dataset D, and a classification method m. One type of classifier model is the rule-based classifier. A rule-based classifier is comprised of a set of rules R. Each $R_j \in R$ is a sequence of conditions and also has an associated L_i. The L_i for x is chosen by checking which $R_j \in R$ is satisfied by x. x is then taken as the class label of R_j. To build a rule-based classifier, D must contain records similar to x which have known class labels, and m must be a rule-based classification method such as C4.5 [4] or CSForest [5][6].

Clustering is the task of finding groups (known as clusters) of similar records within a dataset D. The set of found clusters C is often called the clustering solution. Unlike classification, clustering typically does not require D to have known class labels. Continuing the previous example, if we take all $x \in D$ where $L_i = \text{“}Asthma\text{”}$, we would have a new dataset D' which does not have class labels. A clustering method may find two clusters within D'. One where each patient was diagnosed with asthma since birth and one where the diagnosis occurred after pneumonia. These two clusters would be class-specific to “Asthma”. The centroid of a cluster is the mean of all records within a cluster.

Whereas classifiers may be evaluated by testing the accuracy on a separate testing dataset, a clustering solution is harder to evaluate. There exist many metrics for evaluating clustering solutions. These metrics often require the calculation of distances between records resulting in a high computational cost. For example, the silhouette coefficient [7] metric requires the distance to be calculated between every possible $\{(x_i, x_j) | x_i \in D, x_j \in D\}$. Many clustering methods require potential clustering solutions to be evaluated multiple times. This evaluation step contributes heavily to the computation time of clustering methods. State-of-the-art clustering methods may even take several hours to run [8].

K-means [9] is a clustering method which first randomly chooses k number of records where k is user-defined. These chosen records are called seeds. Each record in D is then assigned to its nearest seed such that k number of clusters are formed. The centroids of these clusters are chosen as the new seeds, and the process is repeated until some termination condition or a maximum number of iterations is reached. The main disadvantage of k-means is that the user needs to guess the number of clusters k in the dataset. If the version of k-means uses a termination condition, then the condition must be checked every iteration. This can become time consuming. Several algorithms such as Affinity Propagation [10] and GenClust [8] do not require the user to estimate the number of clusters in D. However, these algorithms typically have high-complexity (GenClust and AP both have quadratic complexity).

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

3 Our Method: RBClust

In Section 2, we highlighted two main disadvantages with the reviewed existing methods. Long computation time, and the reviewed existing methods are not designed specifically for finding CSC. Therefore, they do not make use of the overall dataset for finding the CSCs. We propose a rule-based clustering method RBClust which aims to overcome these disadvantages. The pseudocode for RBClust is shown in Algorithm 1. The steps of RBClust can be described as follows:

- **Step 1: Rule Discovery** Build a rule based classifier over the whole training dataset D. In Algorithm 1 the classifier choice is the parameter m[1]. The rules need to be extracted from the classifier, that is, the results of applying m to D. Later, in Section 4 we choose the C4.5 [4] algorithm for m. C4.5 uses a pruning step to avoid a classifier that is too specific to the training data (known as overfitting). However, since we are interested in a classifier with high accuracy on D, we turned pruning off.

- **Step 2: Record Sorting** A tri-dimensional array X is used to store the records where X_{ijk} is the k^{th} record which follows the j^{th} rule and belongs to the i^{th} class. The simplest way of performing this step is to classify all records from D using the classifier.

- **Step 3: Cluster Formation** Using each $X_{ij} \in X$, we build the data structure required for a cluster and store it as $C_{ij} \in C$. During this step, we ignore empty sets of X_{ij} (where no i^{th} class records followed the corresponding rule).

- **Step 4: Cluster Merging** There are three problems that could be present after using Steps 1-3. Firstly, D could have an imbalanced ratio of records in each class. For example, there could be 200 records with class label L_1 and 40 records with L_2. This is a well-known problem in classification called class-imbalance. If D is class imbalanced, RBCClust has many more records to find clusters specific to L_1 than clusters specific to L_2. Secondly, the clusters in C could be too specific to D which can cause the clustering solution to have poor generalization on future unseen records. This is known as cluster overfitting. Thirdly, sets of X_{ij} could have a very small number of records. Many other algorithms would not consider this a cluster, and instead treat it as a noisy record. To avoid these problems, we introduce a parameter θ. During this step, RBClust checks each $C_{ij} \in C$ to see if it has a minimum number of records. If it does not, all records from C_{ij} are moved to the nearest cluster (Lines 25-29). The minimum number of records S_i^{min} for an i^{th} class cluster is calculated in Line 23 by multiplying the total number of i^{th} class records in D by θ. This combats each of the previously mentioned problems.

[1]The classifier must produce rules which include all training records and are mutually exclusive. C4.5 is an example

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Input: A dataset D, a rule-based classification method m (default: C4.5), minimum cluster percentage θ (default: 0.02)
Output: A set of clusters C where C_{ij} is the j^{th} cluster specific to the class with index i

Step 1: Rule Discovery
  Let R be a set of rules;
  $R \leftarrow RetrieveRules(D, m)$;
end
Step 2: Record Sorting
  Let X be a set of records where X_{ijk} is the k^{th} record which follows the j^{th} rule and belongs to the i^{th} class;
  Similarly, let X_{ij} be the set of records which follow the j^{th} rule and belong to the i^{th} class;
  $X \leftarrow ClassifyRecords(D, R)$;
end
Step 3: Cluster Formation
  foreach $X_{ij} \in X$ **do**
    if *numberOfRecords*$(X_{ij}) > 0$ **then**
      C_{ij}.add(X_{ij});
    end
  end
end
Step 4: Cluster Merging
  Let $S_i \in S$ be the number of records in D with the class that has the i^{th} index;
  $S_i \leftarrow CountClassRecords(D)$;
  Let S_i^{min} be the minimum number of records allowed for a class-specific cluster with class index i;
  foreach $S_i \in S$ **do**
    $S_i^{min} \leftarrow S_i * \theta$;
  end
  foreach $C_{ij} \in C$ **do**
    if *numberOfRecords*$(C_{ij} < S_i^{min})$ **then**
      Find the nearest cluster (distance between centroids) to C_{ij}. Let it be $C_{ij}^{neighbor}$;
      Combine C_{ij} and $C_{ij}^{neighbor}$ into one cluster;
      Update C accordingly;
    end
  end
  return C;
end

Algorithm 1: RBClust

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

4 Experiments

Setup

We empirically compare the proposed method *RBClust* against a basic version of k-means *SimpleKMeans*, and *Affinity Propagation (AP)*. We chose the basic k-means since it is what was used in an early work which required CSCs [2]. AP was chosen due to its popularity within traditional clustering literature. We compare the methods over 4 class-specific problems which come from 2 separate datasets. The transfusion dataset is available from the UCI Machine Learning Repository [11]. Since class-specific clustering is a major component in several class imbalance related methods [1][2][3], we used a dataset (ecoli1) from the imbalanced dataset repository in KEEL [12]. We used the implementation of SimpleKMeans from WEKA [13] and the implementation of AP from ELKI [14] (version 0.7). For k-means, we set k to 4 to be consistent with [2]. For RBClust, we set θ to 0.02 based off our experiments. All other parameters are set to default values. For the cluster evaluation metric, we use the well-known silhouette coefficient [7].

Table 1: Methods Comparison - Silhouette Coefficient

	Data Information		Methods		
<Dataset-ClassValue>	#Records	C4.5 AUC	kMeans	AP	RBClust
Transfusion-1	178	0.717	0.307	0.179	**0.408**
Transfusion-0	570	0.717	-0.028	0.074	**0.227**
ecoli1-P	77	0.924	0.26	0.194	**0.345**
ecoli1-N	259	0.924	-0.006	0.184	**0.235**

Discussion of Results

Table 2: Time to Find all CSCs (ms)

Dataset	kMeans	AP	RBClust
Transfusion	116	5496	197
ecoli1	128	192	199

The results for the cluster silhouette comparison are shown in Table 1. RBClust achieves the highest silhouette coefficient (shown in bold) in all four problems compared to the existing methods. kMeans can achieve poor results due to the set number of clusters (k). However, this is not an issue for RBClust since the number of clusters is not pre-defined (also true for AP). The performance of the C4.5 tree (measured in AUC[2]) that was used in RBClust is also shown in Table 1. We can see that RBClust is able to perform better than the existing methods even when the performance is significantly lower (-0.207 AUC). We also recorded the computation time needed for finding all CSCs within each dataset. This information is shown in Table 2. It is apparent that for the larger dataset

[2]Ranges from 0 to 1. Higher the better

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

(Transfusion), AP scales poorly in computation time. However, RBClust is able to perform the clustering process in a small fraction of the time. Even though kMeans is slightly faster in both datasets, we can see in Table 1 that it does not perform as well as RBClust. Based on these results, we believe that RBClust has promise for delivering high quality clusters with low computation time. Our future work involves extending RBClust for higher quality without significantly increasing the computation time.

Code Availability

To benefit the reproducibility of our results, the code used to run RBClust will be available at "www.mikesiers.com/software" and "http://csusap.csu.edu.au/˜zislam/" at the time of publication.

References

[1] Michael J Siers and Md Zahidul Islam. Standoff-balancing: A novel class imbalance treatment method inspired by military strategy. In *AI2015 (Accepted)*. 2015.

[2] Nathalie Japkowicz. Concept-learning in the presence of between-class and within-class imbalances. In *Advances in Artificial Intelligence*, pages 67–77. Springer, 2001.

[3] Adam Nickerson, Nathalie Japkowicz, and Evangelos Milios. Using unsupervised learning to guide resampling in imbalanced data sets. *Proceedings of the Eighth International Workshop on AI and Statitsics*, pages 261–265, 2001.

[4] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 1993.

[5] Michael J Siers and Md Zahidul Islam. Cost sensitive decision forest and voting for software defect prediction. In *PRICAI 2014: Trends in Artificial Intelligence*, pages 929–936. Springer, 2014.

[6] Michael J Siers and Md Zahidul Islam. Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. *Information Systems*, 51:62–71, 2015.

[7] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[8] Md Anisur Rahman and Md Zahidul Islam. A hybrid clustering technique combining a novel genetic algorithm with k-means. *Knowledge-Based Systems*, 71:345–365, 2014.

[9] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[10] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[11] M. Lichman. UCI machine learning repository, 2013.

[12] J Alcalá, A Fernández, J Luengo, J Derrac, S García, L Sánchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2010.

[13] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[14] Elke Achtert, Hans-Peter Kriegel, and Arthur Zimek. Elki: a software system for evaluation of subspace clustering algorithms. In *Scientific and Statistical Database Management*, pages 580–585. Springer, 2008.