A Fuzzy Rule-Based Learning Algorithm for Customer Churn Prediction

Bingquan Huang[1(✉)], Ying Huang[1], Chongcheng Chen[2], and M.-T. Kechadi[1]

[1] The INSIGHT Centre, University College Dublin, Belfield, Dublin 4, Ireland
bingquan.huang@insight-centre.org

[2] Key Lab of Spatial Data Mining and Information Sharing of Ministry of Education, Fuzhou University, Fuzhou, People's Republic of China

Abstract. Customer churn has emerged as a critical issue for Customer Relationship Management and customer retention in the telecommunications industry, thus churn prediction is necessary and valuable to retain the customers and reduce the losses. Recently rule-based classification methods designed transparently interpreting the classification results are preferable in customer churn prediction. However most of rule-based learning algorithms designed with the assumption of well-balanced datasets, may provide unacceptable prediction results. This paper introduces a Fuzzy Association Rule-based Classification Learning Algorithm for customer churn prediction. The proposed algorithm adapts CAIM discretization algorithm to obtain fuzzy partitions, then searches a set of rules using an assessment method. The experiments were carried out to validate the proposed approach using the customer services dataset of Telecom. The experimental results show that the proposed approach can achieve acceptable prediction accuracy and efficient for churn prediction.

Keywords: Fuzzy rules · Churn prediction · Rule-based classification

1 Introduction

Service companies in telecommunications suffer from a loss of valuable customers to competitors; this is known as customer churn. In the last a few years, there have been many changes in the telecommunication industry, such as, new services, technologies and the liberalizations of the market increasing competition. For most services, once the number of service subscribes reaches its peak, the company places a special emphasis on the retention of valuable customers because acquiring new customer is difficult and costly. Recently, data mining techniques have been emerged to tackle the challenging problem of customer churn [1–4].

The study of customer churn can be seen as a classification problem. The main goal is to build a robust churn model to predict potential churn customers with the interpretation. Many modelling techniques have been applied to churn prediction problem, including artificial neural networks, CART, Linear Regression, support vector machines, decision trees. For examples, Wei and Chiu [1]

P. Perner (Ed.): ICDM 2016, LNAI 9728, pp. 183–196, 2016.
DOI: 10.1007/978-3-319-41561-1_14

examined an interesting model for churn prediction problem using Decision Trees; Hadden et al. [5] employed Neural Networks, CART, and Linear Regression to build telecommunication churn prediction model; Coussement and Poe [6] investigated the effectiveness of SVM technique over logistic regression and random forests; Wai-Ho et al. [7] proposed the general-to-specific rule learning algorithms for churn prediction. In these techniques, beside obtaining acceptable results, the rule-based modelling techniques only can provide the easiest interpretable concept form of representation.

However, most of rule-based learning techniques were designed and attempt to find an accurate performance over a full range of samples, based on the balanced classes of training dataset [1,6,8–10]. If learning from the dataset with the highly imbalanced distribution of churners and non-churners, these learning techniques tend to be overwhelmed by the non-churn class and ignore the churn class, and consequently might provide poor churn classification results. Therefore, most of rule-based learning techniques may not be more capable towards customer churn classification. Although a number of the techniques were introduced (For example, Huang and Sato [11] introduced the PCA-based and SMOTE [12] sampling methods), the prediction accuracy still need to be improved and the most of them are not based on a learning strategy which is widely used.

In this paper, we introduced a modelling approach – Fuzzy Association Rule-based Classification Learning Algorithm, for customer churn prediction. This proposed learning algorithm finds fuzzy partitions, heuristically searches a set of fuzzy rules, then prunes the unimportant and redundant rules. Finally, the algorithm uses this set of rules with the membership function to classify the observations. In order to handle problem of imbalanced classes in churn prediction, the proposed algorithm adapts CAIM discretization algorithm to obtain fuzzy partitions of the dataset, and uses support weigh factors to calculate the rule support degree during searching rule set. In the experiments, the lift curve and AUC techniques were used to evaluate the performance of the proposed algorithm.

The rest of this paper is organised as follows: Sect. 2 introduces the basic concepts of fuzzy rules. Section 3 describes the proposed algorithm. Section 4 provides the experimental results and discussion. The conclusion and future work are given in Sect. 5.

2 Basic Concepts

Consider a training dataset consists of $|D|$ samples $x_i = (x_{i1}, ..., x_{im})$, $i = 1, 2, ..., D$ and C classes where x_{ip} is the p-th attribute value ($p = 1, 2, ..., m$) of the i-th training sample. A fuzzy rule of classification usually can be presented as the following form for a classifier:

Rule R_j**: If** $x_i \in A_{1p} \bigwedge ... \bigwedge x_m \in A_{jm}$**, then Class** $= C_j$;

where R_j is is the label of the j-th rule, $x = x_1, ..., x_m$ is an m-dimension input vector, A_{jp} is an antecedent fuzzy set, C_j is a class label. A fuzzy associative classification rule R_j can be measured directly in terms of support and confidence [13] as follows:

$$FS(R_j) = \frac{\sum_{x_i \in C_j} \mu_{A_j}(x_i)}{|D|} \tag{1}$$

$$FC(R_j) = \frac{\sum_{x_i \in C_j} \mu_{A_j}(x_i)}{\sum_{x_i \in D} \mu_A(x_i)} \tag{2}$$

where FS and FC are respectively the fuzzy support and confident values, $|D|$ is a number of samples in the training dataset, $\mu_{A_j}(x_i)$ is the matching degree of input vector x_p with the antecedent and consequent of the rule R_j. $\mu_A(x_i)$ is the matching degree of x_i with the antecedent part of the rule. The match degree can be calculated by the trap function:

$$\mu_{A_j}(x_i) = \max\{1 - \mid \frac{x_i}{F} - (k-1) \mid, 0\} \tag{3}$$

where k is the index of the intervals from 0 to m, m is a number of intervals of the p^{th} attribute, and F is calculated by:

$$F = \frac{1}{m-1} \tag{4}$$

As mentioned above, in a IF-THEN rule, the antecedent part consists of a number of pairs <attribute, interval> and the **consequence** denotes the class label. In this paper, a pair <attribute, interval> is defined as an **item**. If the consequence of a rule is the class c, the items of this rule are called **Class^c_items**. For examples, if the consequence of a rule is class "positive", the items of this rule are **positive_items**.

A rule that has only 1 item in the antecedent part is called a **1_Order_Rule**. If a rule that has 2 items in the antecedent part is called a **2_Order_Rule**. Similarly, a rule that has n items is called a **n_Order_Rule**. If a rule that has more than 1 items is called a **higher_Order_Rule**.

If the consequence of a *n_Order_Rule* is the class label c, this rule is written as a **class^c_n_Order_Rule**. Similarly, the **class^k_higher_Order_Rules** and **class^k_lower_Order_Rules** respectively refer to the higher-Order Rules and lower-Order Rules that have more than 1 items in the antecedent part.

Another basic concept of rule induction is **Cover**. It expresses that a sample satisfies the antecedent part of the rule. That's, when a sample matches the condition of a rule, the rule *Correctly* **Covers** this sample. The task of rule induction is to build a model by generating a set of rules, which either do or do not Cover sample.

General rules and **specific rules** are two important concepts. Usually, a rule with a smaller number of pairs <attribute, interval> in the **antecedent** part

is more **general** than the one having more. The reverse sample is more *specific*. The *Most general* rules are the First-Order-Rules, while the *most specific* means the number of antecedent elements is the total number of attributes. Normally, the most specific rules should not be considered since they are too specific and not able to describe a relative big data space.

3 Rule-Based Classification Algorithms

The objective of the new algorithm is to generate a set of rules for churn classification. Generally, this proposed approach (1) uses the adapted fast CAIM Discretization algorithm [14] to find the intervals of each continue attribute; (2) applies the heuristic methods with the fuzzy logic membership function to generate classification rules with the fuzzy support and confidence calculated by using the adapted method; (3) prunes these rules; (4) uses these set of rules to predict the given observations.

3.1 Fuzzy Partition of Continue Attributes

As the first step, a continuous attributes is discretized into a number of intervals. Based on these intervals, the classical triangle membership function is used to calculate the fuzzy degrees. There are two kinds of discretization algorithms: supervised and unsupervised techniques. In the contrast of them, the supervised discretization algorithms are outperform for classification tasks [14]. As one of promising supervised discretization algorithm, the Fast CAIM (F-CAIM) algorithm [14] outperforms most of discretization algorithms (e.g. cluster algorithms) [14]. The goal of fast CAIM is to maximize the class-attribute interdependence and to generate a minimal number of discrete intervals. The detail of this algorithm can be found in [14].

As mentioned above, the distribution of classes (churn and nonchurn) of our application is very imbalanced. However, fast CAIM Algorithm was designed under the hypothesis of the class balanced in the dataset. Therefore, this paper adapts the F-CAIM algorithm to discrete data for our application. Based on the F-CAIM algorithm, we only changes the CAIM discretization criterion into the following:

$$CAIM'(C, D|F) = \frac{\sum_{r=1}^{N} \frac{\max \frac{q_{ir}}{M_{i+}}}{M_{+r}}}{n} \tag{5}$$

where: q_{ir} is the total number of continuous values belonging to the i^{th} class that are within interval $(d_{r-1}, d_r]$. M_{i+} is the total number of objects belonging to the i^{th} class, and M_{+r} is the total number of continuous values of attribute F that are within the interval $(d_{r-1}, d_r]$, for $i = 1, 2..., S$ and, $r = 1, 2, ..., n$. $\frac{q_{ir}}{M_{i+}}$ is the portion of the i^{th} class objects. Therefore, when measuring the dependency

between class variable and the discretization variable for an attribute, the modified criterion 5 take the class ratio to account, and consider less of the number of objects for an variable interval. This can lead the adapted discretization algorithm to more successfully discretize the continuous attributes in the imbalanced class data.

3.2 Rule Learning

As mentioned above, there is a very challenge for traditional rule-based learning techniques to learn from the dataset which consists of the highly imbalanced distribution of churners and non-churners, as consequently, the classifiers would easy ignore the minority classes (churners). In order to handling this issue, we adapt the support of rules to strengthen the significant of classes by multiplying weight factors, called support weight factors. The measurement of rule support is replaced by the following formula:

$$FS'(R_j) = w_{C_j} * \frac{\sum_{x_i \in C_j} \mu_{A_j}(x_i)}{|D|} \tag{6}$$

where w_{C_j} is the support weight factor of the j-th class. When the weight factor w_{C_j} increases, the degree of significance of the rule R_j and the rule weight will increase, according to Eqs. (7), (8) and (9). Therefore, this formula can effect the rule evaluation, pruning and classification (see Sects. 3.3 and 3.4). Usually, we expect that the weight factor w_{C_j} for the more interesting class is large in such a way keep related class rules for the classification. The best weight factor w_{C_j} can be selected by using the AUC technique Sect. 4.3: (1) use a number of different value for the weight factor w_{C_j} to build a number of models and validate the training data, (2) calculate AUC values, and (3) select the value which obtain the maximum AUC value as the best weight factor w_{C_j}.

The proposed algorithm applies the strategy of general-to-specific to generate a set of rules from the dataset. According to the class labels of training data, the algorithm firstly generates the first-order rules (the most general rules) of each class. Next, the learning algorithm iteratively constructs the higher-order rules of each class, based on the lower-order rules. Suppose SET^c_{rules} is the set of generated rules of class c, FullSET_{rules} is the full set of rules, k is the number of orders, data-subset_c is the discretised sub-dataset of class c, and Max_num_order is the maximal number of orders. The process of this learning rules can be illustrated by Algorithm 1.

First-Order Fuzzy Rules. As mentioned in Sect. 2, the first-order fuzzy rules are **the most general rules**, which the number of antecedent elements is 1. The first-order rules can be obtained by two steps. First, for each interval of each attribute, any single pair <attribute, interval> is considered as the antecedent of a new rule; then this new rule is evaluated (which is described in Sect. 3.3); if this

Algorithm 1. Rule-based Learning Algorithm

$FullSET_{rules} \leftarrow \phi$;
$c \leftarrow 1$;
while $c \leq C$ **do**
  $SET^{c}_{rules} \leftarrow \phi$;
  $SET^{c}_{rules} \leftarrow Class^{c}\_1\_order\_rules$(training dataset);
  $k \leftarrow 2$;
  while $k \leq Max\_num\_order$ **do**
    $class^{c}\_k\_higher\_order\_rules \leftarrow Class^{c}\_high\_order\_rules$;
    $SET^{c}_{rules} \leftarrow SET^{c}_{rules} \bigcup class^{c}\_k\_higher\_order\_rules$;
    $k++$;
  end while
  $\text{FullSET}_{rules} \leftarrow \text{FullSET}_{rules} \bigcup \text{SET}^{c}_{rules}$;
  $c++$;
end while
return FullSET_{rules}

new rule is valid, it will be selected to combine a higher-order rules. Algorithm 2 illustrates the details process of generating the first-order fuzzy rules.

In Algorithm 2, $num\_attribute$, $num\_interval$ and num_class respectively denotes the number of attributes, the number of fuzzy intervals of the current attribute and the number of classes. The set of $first\_order\_rules$ is initially empty. $rule_{ijk}$ expresses a rule: "IF f_i is s_j THEN $Class_k$". Evaluate(rule_{ijk}) is the function which evaluates whether rule_{ijk} is valid or interested. If Evaluate(rule_{ijk}) returns true, then $rule_{ijk}$ can be accepted as a valid rule (see Sect. 3.3). $Class^{k}\_1\_order\_rules$ is a set of first-order rules that can cover the samples of class K. The $first\_order\_rules$ (1_order_rules) includes all of $\text{Class}^{k}\_1\_\text{order}\_\text{rules}$ ($0 \leq k \leq C$).

Algorithm 2. $First\_order\_rules$

$1\_\text{order}\_\text{rules} \leftarrow \phi$;
for $k = 1$; $k <= num\_class$; $k++$ **do**
  $Class^{k}\_1\_order\_rules \leftarrow \phi$;
  for $i = 1$; $i <= num\_attribute$; $i++$ **do**
    for $j = 1$; $j <= num\_interval$; $j++$ **do**
      if Evaluate(rule_{ijk}) returns $true$ **then**
        $Class^{k}\_1\_order\_rules \leftarrow Class^{k}\_1\_order\_rules \bigcup rule_{ijk}$;
      end if
    end for
  end for
  $1\_\text{order}\_\text{rules} \leftarrow 1\_\text{order}\_\text{rules} \bigcup Class^{k}\_1\_order\_rules$;
end for
return 1_order_rules

High-Order Fuzzy Rules. Basically, the proposed learning algorithm iteratively combines the lower-order rules with items (pairs of ⟨attribute, interval⟩) into higher-order rules by using the heuristical hill-climbing method for each class. The learning algorithm uses the $(n-1)\_order\_rules$ to construct n_order_rules for each class. The learning algorithm starts to construct 2-order-rules based on the 1-order rules, and stops when completed the rules of maximum order.

Suppose: $Class^c\_item$ is a set of pairs <attribute, interval> that are from the rules of class c; $num\_Classes$ is the number of classes; $num\_Class^c\_(n-1)\_rules$ is the number of rules of class c; all_high_rules are the rules of all of classes; $hill\_climbing()$ is the function of heuristical hill-climbing method; $\min\_FC$ is the threshold of fuzzy confidence; The procedure of constructing a set of high rules can be illustrated by Algorithm 3. Algorithm 3 first extracts all exclusive $items$ from the Class$^c_1_order_rules$, secondly uses $hill\_climbing()$ to combine a lower-order rule $Class^c\_(n-1)\_rule$ with 1 $Class^c\_item$ into a new $high\_rule$ (n_order rule), thirdly evaluates this new rule by the function $Valid()$. If this new rule is valid, the algorithm calculate the confidence by Eq. (2). If the fuzzy confidence FC_i^c of this new rule is not less than the threshold of rule confidence values, then this new rule is added into $Class^c\_n\_rule$. This procedure is iteratively carried out to generate the set of $Class^c\_(n)\_rules$, based on the set $Class^c\_(n-1)\_rules$ and the item set of Class$^c_1_order_rules$. Finally, the algorithm combines all of the high-order rules of different classes $\{Class^c\_(n)\_rules,\ \ 1 \le c \le C\}$ into one set of all_high_rules.

Algorithm 3. $Higher\_Order\_Rules(first\_order\_rules)$

```
all_high_rules ← φ;
n ← 1;
while n ≤ Maxim_Order do
    n_order_rules ← φ;
    for c = 1; c ≤ num_Classes; c++ do
        Class^c items ← extract all exclusive items from Class^c 1_order_rules;
        Class^c n_order_rules ← φ;
        accuracy_list ← φ;
        for i = 1; i ≤ num_Class^c (n − 1)_rules; i++ do
            high_rule ← hill_climbing(Class^c (n − 1)_order_rule_i, Class^c items);
            if Valid(high_rule) returns true then
                FC_i^c ← FC(high_rule);
                if FC_i^c ≥ min_FC then
                    Class^c n_order_rules ← Class^c n_order_rules ⋃ high_rule;
                end if
            end if
        end for
        n_order_rules ← n_order_rules ⋃ Class^c n_order_rules;
    end for
    all_high_rules ← all_high_rules ⋃ n_order_rules;
    n++;
end while
return all_high_rules;
```

The heuristical hill-climbing method for generating $high\_order$ rules is illustrated by Algorithm 4. In Algorithm 4, FC_list is a list of different FC values and is initially empty. Basically, one piece of $high\_order$ is based on one piece of low_order rule and a $new\ item$, which has been mentioned earlier. If the $new\ item$ is not included in the antecedent of the low_order rule, then the hill-climbing algorithm combines this $lower\_rule$ and this $new\ item$ into t one new $high\_order$ rule. Meanwhile, the FC value of the newly built this $high\_order$ rule is stored in the $accuracy\_list$. This algorithm returns the $high\_order$ rule with the highest of FC values.

Algorithm 4. *hill_climbing*(*one_lower_rule*, *all_low_items*)

```
FC_list ← φ;
for i = 1; i <= num_low_items; i++ do
    if low_rule does not include item_i then
        high_rule ← combination of low_rule and item_i;
        FC_i ← FC(high_rule);
        FC_list ← FC_list ⋃ FC_i;
    end if
end for
BEST ← one of high order rules with the highest FC value;
return BEST
```

3.3 Rule Assessment

Due to the redundant information and the over-fitting problem, the proposed rule learning algorithm applies χ^2 statistic test to measure the quality of rules in such way to find the interesting rules. In this paper, χ^2 is used to measure the correlation between the antecedent part and the consequence of a rule. The stronger correlation usually presents more significance. Suppose there is a rule which A is the antecedent, B is the consequent, and $A \Rightarrow B$. The χ^2 statistic value of this rule can be calculated by:

$$\chi^2(A \Rightarrow B) = \frac{(O(AB) \times w_C - E(AB))^2}{E(AB)} \tag{7}$$

where $O(AB)$ is the observed frequency of the rule, w_C is the support weight factor for the consequent B (B is C_j), and $E(AB)$ is the expected frequency which can be calculated by Eq. (8):

$$E(AB) = \frac{(|D| \times FS'_A) \times (|D| \times FS'_B)}{|D|} = FS'_A \times FS'_B \times |D| \tag{8}$$

Where $|D|$ is a number of samples in training data, FS'_B and FS'_A are respectively the fuzzy support values of B and A, which can be obtained by Eq. 6.

Algorithm 5 illustrates the pruning method. α is the critical value for χ^2 significance test, and it is set to be 3.84 in this research. $minS$ and $minC$ are two threshold values, which define the minimal value of support and confidence, respectively. In this research, the minimum value of support and confidence were set to be 0.01 and 0.5, respectively. A rule can be retained if it satisfies all the threshold values.

Algorithm 5. Evaluate(*rule*)

```
Flag ← false;
if chi-square statistics > α AND Support_rule > minS AND Confidence_rule > minC then
    Flag ← true;
end if
return Flag;
```

3.4 Classification

Once the proposed algorithm obtains a set of different class rules from the given training dataset, the algorithm can use this set of rules to classify the observations. Suppose the observation is x_i, the classification procedure can be described as: (1) the algorithm calculates weight values of x_i in each rule by Eq. (9), then (2) assigns the class label with the highest weight value to the observation x_i.

$$weight(R_j, x_i) = \mu(x_i) \times FC(R_j) \times FS'(R_j) \tag{9}$$

where $\mu(x_i)$ is the aggregate membership function of the multiple conditions of R_j, and is calculated by $\mu_{A_j}(x_{ip}) \cdot ... \cdot \mu_{A_j}(x_{ip}) \cdot ... \cdot \mu_{A_j}(x_{im})$;

4 Experiments and Discussion

4.1 Dataset

Experiments were conducted by the telecommunication dataset of the Ireland Telecoms [15]. The telecommunication dataset also was divided into one training dataset and one testing datasets. There are 6000 churners, 94000 nonchurners and total 100000 customers in the training dataset. In the testing dataset, there are 39000 customers which includes 2000 churners and 37000 nonchurners. the 122 features were extracted using constant non-time series data (e.g. demographic profiles, payment method, etc.) and time series data (e.g. call-details that consists of five-month call-details M 4, ..., M 1, M. The time series training data lags one month behind testing data. The detail of these features can be found in [3,16].

4.2 Experiment Set-up

In order to evaluate the proposed learning algorithm, two sets of experiments were carried out in this paper. The first set of experiments is to compare the performance of the proposed learning algorithm and traditional learning algorithms. The second set of experiments is to evaluate the performance of the different rule-based learning algorithms which employ different discretization methods or non-fuzzy rules.

Experiment Set-up I. In this set of experiments, six modelling techniques (DMEL [7], Logistic regression, CN2, MLP, C4.5 and the new proposed algorithm) were used to make a prediction for the telecom dataset. The related software tools used in the experiments are Weka and CN2 software packages of R. Boswell [17]. All the predictors were trained by 10 folds of cross-validations in each experiment. The Wrappers for Performance Enhancement and Oblivious Decision Graphs algorithm (WPEODG) [18] was used to select the best learning parameters for C4.5, MLP and Logistic regression.

The three-layer MLPs, each of which only contains one hidden layer, were used. The number of input neurons in the MPL is the same as the number of dimensions in a feature vector. The number of output neurons of the network is the number of classes. The sigmoid function is selected as the activation function for all MLPs. Each MLP was trained by BP learning algorithm with learning rate 0.1, maximum cycle 1800 and tolerant error 0.05. The number of training cycles to yield the highest accuracy is about 800.

Each DMEL model was trained by the following parameters: population size of 30, number of generations of 300, Z-value of 1.96 for finding interested features, probabilities of mutation and crossover which are 0.1 % and 60 % respectively. The best parameters for C4.5 are when the pruning confidence threshold is 0.1 and the minimum number of samples per leaf is 2 for the Telecom dataset.

For the new proposed rule learning algorithm, the threshold value of fuzzy support (min_fs) is set to 0.01, which is a commonly used value; while the critical value of fuzzy confidence (min_fc) is set in the range [0.5: 0.9] with the step size of 0.02, the optimal min_fc would be the one that leads to the largest AUC value for both datasets. The support weight factors for churner and nonchurner classes (w_{C_0} and w_{C_1}) are 0.8 and 0.2.

Experiment Set-up II. In the second set of experiments, the 4 rule-based learning algorithms (which are the new proposed learning algorithm, CAIM-Fuzzy algorithm, CAIM-Rules algorithm and kmeans-Fuzzy algorithm) were carried out. Comparing to the new proposed algorithm, the CAIM-Fuzzy and kmeans-Fuzzy algorithms use different discretization techniques. The CAIM-Fuzzy algorithm uses the supervised CAIM algorithm, and the kmeans-Fuzzy algorithm uses the unsupervised clustering algorithm – Kmeans algorithm, to discrete the data. Contrasting to new proposed learning algorithm, the CAIM-Rule learning algorithm uses the supervised CAIM to discrete data and the traditional rules for the classification. The value K of the Kmeans algorithm was 20. The parameters used in the learning algorithm are as the same as the ones used in the first set of experiments.

4.3 Evaluation

Generally, the service operators aim to relatively identify more potential churners by spending less cost (i.e., contacting a relatively small amount of customers). Accordingly, it is important for the telecom operators to estimate how much percent of potential churners can be identified by contacting a certain small proportion of customers (e.g. 5 % or 10 %). Therefore, Lift Curve [19] is applied in this work to evaluate the classification models.

The $Lift\ Curve$ method first scores each data sample, then sorts the samples according to the scores, finally, it calculates the portion of samples and the True Positive rate (true churn in this paper), respectively.

However, it is difficult to use the lift-curve for comparing two or more results, since it is difficult to say whether one result is better than the other results.

To overcome this problem, the AUC is also used to evaluate the models. The area under a Lift Curve can be calculated by the following (if it was calculated based on trapezoid):

$$AUC(t) = \sum_{i=1}^{N} 0.5 \times (Y_i(t) + Y_{i+1}(t)) \times (X_{i+1}(t) - X_i(t)) \tag{10}$$

where N is the number of the trapezoid, $X_i(t)$ are the portion of selected samples and $Y_i(t)$ is the true positive rate at time t. The area under lift curve is the sum area of a series of trapezoids, the details of AUC can be found in [19].

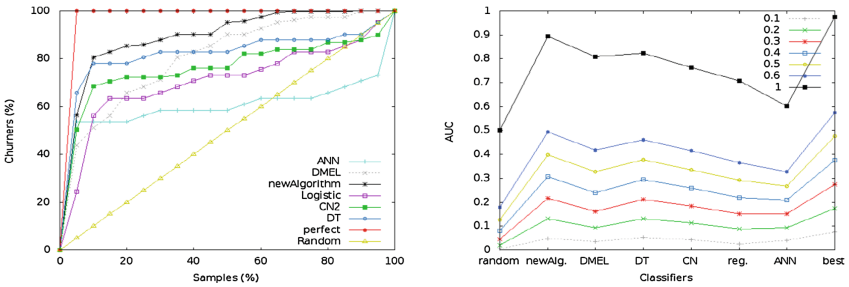
4.4 Results and Discussion

In order to retain the customers with less costs, the service operators select a small portion of customers to offer special consideration or attractive services. The lift curve technique was used to find a relative small amount of customers that are the relative more potential churners. In order more accurately evaluate those modelling techniques, the AUC technique was used to analyses the experimental results and the learning modelling algorithms. Figures 1 and 2 show the experimental results which refer the lift charts with AUC from the first and second set of experiments, respectively.

In each lift chart figure, the horizontal axis represents the portion of the considered customers, the vertical axis represents the true positive rate, and the different colours represent the different modelling techniques used in the experiments. In each AUC figure, the x-axis represent different modelling techniques and the y-axis represents the values of AUC, and the different colors presents different portion of customers (10 %, 20 %, 30 %, 40 %, 50 %, 60 % and 100 %) were selected.

Result I. In a lift chart, the lift curve which is closer to the left-top corner (the area under it is the larger) is better than the ones that are farther to the left-top corner. In Fig. 1(a), the theory with the red curve is the best one, while the yellow curve that was obtained by randomly selecting samples is the worse. From these two figures, when a small portion of the selected samples are about 10 % less, the best curve is from DT and the better one is from the new proposed technique, but both of them are very similarly covered together. It also shows, generally, the block curve from the new proposed learning algorithm is closer than others, excepting these theory curves (the red one).

The related AUC is plotted plotted into Fig. 1(b). It shows the AUC values obtained by different modelling algorithms when the different portions of samples were selected. From the AUC, we can obtain that: (1) when the sample portions from 30 % to 100 % were selected for the Telecom dataset, the new proposed technique obtained the largest AUC values. (2) when the sample portion of 10 % or 20 % was selected for the Telecom dataset, the largest AUC values are from the new proposed technique while DT and AUC values were obtained by the

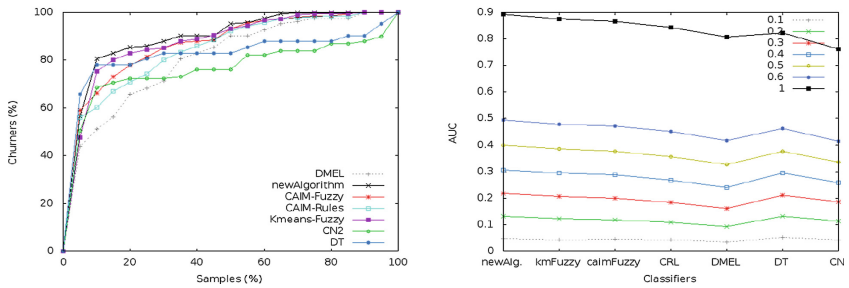


(a) Lift charts (b) AUC

Fig. 1. Lift curves and the AUC of traditional and new modelling techniques (Color figure online)

new proposed technique is approximately equal to the ones obtained by DT. Therefore, the new proposed modelling technique which obtained larger AUC values are more suitable for churn prediction problem than others that obtained lower AUC values.

Result II. The comparative results from the 4 rule-based learning algorithms are showed in Fig. 2. These two figures show: (1) which rule-based learning algorithm obtained larger AUC values than others for a specified portion of the considering samples. For examples, among these 7 learning algorithms (including the new proposed learning algorithm, CAIM-Fuzzy algorithm, CAIM-Rule algorithm and kmeans-Fuzzy algorithm, DMEL, CN and DT), the AUC values of DT and the new proposed algorithm generate are larger when the sample portion of 10 % of considering samples is 20 % less. (2) when the sample portion of 10 % was selected from the Telecom dataset, the AUC values of the 4 rule learning algorithms are approximately equal. (3) when the sample portion increasing, among these 4 learning algorithms, the AUC values generated by the new proposed algorithm increase largest.



(a) Lift charts (b) AUC

Fig. 2. The Lift curves and the AUC of rule-based learning techniques

5 Conclusion and Future Works

In this paper, we used the fuzzy rule-based classification technique to predict the customer churn for Telecom services. Lift Curve and AUC techniques were applied to evaluate the predicton techniques in the experiments. The results highlight that the rule-based classification technique is more efficient than the traditional rule-based methods which were designed and attempt to find An accurate model over a full range of samples using on the balanced classes of training dataset.

When using fuzzy-based rules to deal with large number of features, the computational overhead may increases for generating high order rules. In the future, we need to find a method to control the number of high order rules rather than setting the size arbitrarily. In addtion, theoretically, the cost-sensitive learning algorithms and sampling methods (e.g. PCA, SMOTE) can be used with fuzzy-based classification methods to improve results for the churn prediction. This will be studied in the future.

Acknowledgement. The authors gratefully acknowledge the research support from the EU Framework Programme 7, Marie Curie Actions under grant No. PIRSES-GA-2009-247608.

References

1. Wei, C., Chiu, I.: Turning telecommunications call details to churn prediction: a data mining approach. Expert Syst. Appl. **23**, 103–112 (2002)
2. Hung, S.-Y., Yen, D.C., Wang, H.-Y.: Applying data mining to telecom churn management. Expert Syst. Appl. **31**, 515–524 (2006)
3. Huang, B.Q., Kechadi, M.-T., Buckley, B.: Customer churn prediction for broadband internet services. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 229–243. Springer, Heidelberg (2009)
4. Huang, B.Q., Kechadi, M.-T., Buckley, B.: A new feature set with new window techniques for customer churn prediction in land-line telecommunications. Expert Syst. Appl. **37**, 3657–3665 (2010)
5. Hadden, J., Tiwari, A., Roy, R., Ruta, D.: Churn prediction: does technology matter? Int. J. Electr. Comput. Eng. **1**, 6 (2006)
6. Coussement, K., Van den Poel, D.: Churn prediction in subscription services: an application of support vector machines while comparing two parameter-selection techniques. Expert Syst. Appl. **34**(1), 313–327 (2008)
7. Au, W., Chan, C.C., Yao, X.: A novel evolutionary data mining algorithm with applications to churn prediction. IEEE Trans. Evol. Comput. **7**, 532–545 (2003)
8. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: KDD, pp. 80–86 (1998)
9. Show-Jane, Y., Yue-Shi, L.: Cluster-based under-sampling approaches for imbalanced data distributions. Expert Syst. Appl. **36**(3), 5718–5727 (2009)
10. Burez, J., Van den Poel, D.: Handling class imbalance in customer churn prediction. Expert Syst. Appl. **36**(3), 4626–4636 (2009)

11. Sato, T., Huang, B.Q., Huang, Y., Kechadi, M.-T., Buckley, B.: Using PCA to predict customer churn in telecommunication dataset. In: Cao, L., Zhong, J., Feng, Y. (eds.) ADMA 2010, Part II. LNCS, vol. 6441, pp. 326–335. Springer, Heidelberg (2010)
12. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005)
13. Yi-Chung, H., Chen, R.-S., Tzeng, G.-H.: Finding fuzzy classification rules using data mining techniques. Pattern Recogn. Lett. **24**(13), 509–519 (2003)
14. Kurgan, L., Cios, K.: Fast class-attribute interdependence maximization (caim) discretization algorithm (2003)
15. http://www.eircom.ie/cgi-bin/bvsm/mainPage.jsp
16. Huang, B., Kechadi, M.T., Buckley, B.: Customer churn prediction in telecommunications. Expert Syst. Appl. **39**(1), 1414–1425 (2012)
17. http://www.cs.utexas.edu/users/pclark/software/#cn2
18. Kohavi, R.: Wrappers for performance enhancement and oblivious decision graphs. Department of Computer Science, Stanford University (1995)
19. Vuk, M.: Roc curve, lift chart and calibration plot (2006)