Expert Systems with Applications 36 (2009) 6218–6226

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Calling communities analysis and identification using machine learning techniques

Keivan Kianmehr[a], Reda Alhajj[a,b,*]

[a] Department of Computer Science, University of Calgary, Calgary, Alberta, Canada
[b] Department of Computer Science, Global University, Beirut, Lebanon

ARTICLE INFO

Keywords:
Social communities
Classification
Clustering
Customer behavior
Machine learning

ABSTRACT

The analysis of social communities related logs has recently received considerable attention for its importance in shedding light on social concerns by identifying different groups, and hence helps in resolving issues like predicting terrorist groups. In the customer analysis domain, identifying calling communities can be used for determining a particular customer's value according to the general pattern behavior of the community that the customer belongs to; this helps the effective targeted marketing design, which is significantly important for increasing profitability. In telecommunication industry, machine learning techniques have been applied to the Call Detail Record (CDR) for predicting customer behavior such as churn prediction. In this paper, we pursue identifying the calling communities and demonstrate how cluster analysis can be used to effectively identify communities using information derived from the CDR data. We use the information extracted from the cluster analysis to identify customer calling patterns. Customers calling patterns are then given to a classification algorithm to generate a classifier model for predicting the calling communities of a customer. We apply different machine learning techniques to build classifier models and compare them in terms of classification accuracy and computational performance. The reported test results demonstrate the applicability and effectiveness of the proposed approach.

1. Introduction

Though started as subfield of sociology to identify and study the relations between people and the things they do, with the advance in the computing based prediction and analysis techniques, social network analysis in general and community identification in particular has received considerable and increasing attention over the past decade in different domains, including web mining (Flake, Lawrence, & Giles, 2000; Kleinberg, 1999), biological networks (Girvan & Newman, 2002), among others. So, identifying social communities is an emerging research area that has already attracted the attention of many research groups. The main theme is to analyze logs that reflect social communication between different parties. The analysis leads to valuable discoveries that may have essential social and economical impact. From social perspective, the discoveries may highlight terrorist groups, family relationships, friendship, etc. From economical perspective, the analysis may lead to certain target customer groups. We concentrate on the latter perspective in this paper. In particular, we investigate customer relationships by analyzing call detail records obtained from a telecommunication company. Other researchers concentrated on identifying terrorist groups (e.g., Nasrullah & Larsen, 2006).

In telecommunication, a Call Detail Record (CDR) is a record containing information about recent system usage, such as the identities of sources (points of origin), the identities of destinations (endpoints), the duration of each call, the amount billed for each call, the total usage time in the billing period, the total free time remaining in the billing period, and the running total charged during the billing period. The format of the CDR varies among telecom providers and call-logging softwares.

In recent years, we witness dramatic increase in the competition among telecommunication companies in order to detain their current customers and acquire new ones. For this reason, the ability to dynamically classify and predict customers' behaviors according to their calling patterns obtained from CDR data has attracted considerable attention in the research community; it is beneficial for (Yan, Fassino, & Baldasare, 2005):

- *Churn Prediction:* The goal is to understand when and why company's customers are likely to leave so that appropriate action can be planned. Customers become "churners" when they discontinue their subscription and move their business to a competitor company. This has been developed in the telecommunication industry using data mining techniques. Data mining is applied in this area to perform two major tasks:

* Corresponding author. Address: Department of Computer Science, University of Calgary, Calgary, Alberta, Canada
E-mail address: alhajj@cpsc.ucalgary.ca (R. Alhajj)

1. predict whether a particular customer will churn and when this will happen;
2. understand why particular customers churn.

By predicting which customers are likely to churn, the telecommunication company can reduce the rate of churn by offering the customers new incentives to stay.

- *Identifying Calling Communities:* This can be used for determining a particular customer's value according to the general pattern behavior of the community that the particular customer belongs to. This helps the effective targeted marketing design, which is significantly important for increasing profitability in the telecommunication industry.

The main focus of this study is to use an unsupervised machine learning technique, namely clustering to classify customers of a mobile service provider into appropriate calling communities according to the statistics extracted from the CDR data.

Clustering is one of the most prominent approaches for identifying unknown classes amongst a group of objects, and has been successfully used as a tool in many fields such as biology, image analysis, finance, etc. The classification algorithms evaluated in this paper use an unsupervised learning mechanism, wherein unlabeled training data is grouped based on similarity. Once an acceptable clustering has been found using the similarities and dissimilarities in the training data set, the clustering is transformed into a classifier by employing a classification technique.

In our approach, the clusters are labeled, and a new object is classified with the label of the cluster to which it is most similar. However, the type of the classifier that is to be used to identify calling communities and customers' value is very crucial. For instance, in a marketing campaign, selecting inappropriate customers is very costly for verifying the impact of the marketing campaign when it is intended to modify the strategy toward the target group. In this study, the agglomerative hierarchical clustering approach has been applied for the clustering task. It can produce an ordering of the objects (cluster tree), which may be more informative for the nature of the data being analyzed and investigated.

For building the classifier model, we use supervised Machine Learning (ML) techniques to automatically classify and identify customers' communities based on customers' characteristics extracted from clusters. A ML algorithm constructs a classifier by exploring through data objects (training set) to find a set of rules which determine the class of each object according to its attributes. These rules are later used to predict the class or missing attribute value of unseen objects whose class might not be known. Supervised machine learning techniques have been widely used in real world classification applications such as increasing revenue, preventing theft, medical diagnosis, market analysis signaling potentially fraudulent transactions, etc. In this paper, we evaluate and compare the performance of our classifier model, in terms of accuracy and time complexity, by applying several machine learning techniques. The reported results are promising.

The rest of this paper is organized as follows. Section 2 covers CDR data and calling neighbors. Section 3 presents details of the proposed models. Section 4 reports experimental results. Section 5 is conclusions.

2. CDR Data and calling neighbors

Because of the confidential nature of the customers' information and to preserve the privacy, the information provided by telecommunication companies is very limited for modeling except the CDR data. From the CDR data, it is possible to extract customer's calling destination numbers, duration and frequency for each destination number for a particular period of time. Customer's calling neighbors can be also identified by using the links of who calls whom (Yan et al., 2005). There are two types of calling neighbors:

1. *Direct calling neighbor:* a person who calls the customer or whom the customer calls. The majority of these neighbors of a customer may be outside of the service provider's own network, and no information about them is available. An example of direct calling neighbor can be described as follows: members of a research community call each other heavily.
2. *Indirect calling neighbor:* a person who calls the same number(s) as another customer does. For example, employees of a large organization such as a bank who work in different local branches may call their headquarter frequently, and each employee is possibly a calling neighbor of other employees. Even though employees of different branches may not know each other, they can be classified into the community of colleagues.

To measure the closeness of the first type of calling neighbors, the calling frequency or duration has been used. For the second type of neighbors, a distance measure, which will be presented in more detail in Section 3.2, has been applied to quantify the closeness of a neighbor. Discovering calling neighbors will result in forming calling communities. However, there are two major challenges in using the CDR data for finding the calling communities:

- The majority of the destination phone numbers are outside of the service provider's network. Therefore, information about these customers such as their calling records and links between them is not available.
- The other type of phone numbers are those in the service provider's network, and each of them corresponds to a customer of the service provider. The connectivity between these customers is very sparse because customers more likely call numbers outside their home network.

After identifying the customers communities, the information derived from the calling communities can be used for building a classifier model. This classifier model is able to assign a particular customer to a community according to his/her general calling pattern extracted from the CDR data as well as the closeness of his/her direct and indirect calling neighbors. In terms of market campaign design, the characteristics of communities can be used for increasing the profitability of the service provider company by targeting appropriate communities and customers. Intuitively, people in the same community have more likely similar behaviors because of their influences on the community. For instance, if a customer more frequently calls member of a community in which members generally accept service promotion offers, that particular customer can be considered for targeted service promotions.

3. Model details

In this section, the classifier model that has been developed for the course of this study, will be described in more detail. The system consists of three major phases: (1) data preprocessing, (2) clustering, and (3) classification.

3.1. Data preprocessing

The CDR data used in this work was given by a telecommunication company providing wireless services. Because of the confidential nature of the data and business rules, the CDR data for a short period of time is given. However, the results show that the classifier model built based on the information extracted from calling links within the given CDR data performs classification at a reasonable accuracy.

During the first step of the data preprocessing, all the phone numbers (customers) within the service provider's network have been identified. Since the CDR data includes information about the customers, all the source phone numbers have been considered as the subscribers of the service provider. The given data set consists of 270,000 calling records of 10,200 subscribers (distinct phone numbers within the service provider's network). Based on a careful study of the data, calls with very low duration (less than 5 s) are assumed to have no effect on identifying the subscriber's neighbors and are ignored.

In the second step, for each subscriber, his/her own phone number, called destination numbers (within and outside of the home network), and each destination number's calling duration and frequency in that particular period of time are extracted from the CDR data. During pre-processing of the CDR data, inactive customers have been excluded from the data since experience tells that these numbers greatly influence the distance distribution. Inactive customer refers to a customer who barely makes a phone call within a particular time period.

3.2. Clustering

Among all data mining tasks, clustering is one of the most useful tasks for discovering groups and identifying interesting patterns in the underlying data. It has been widely used in many application domains such as engineering, business and social sciences, among others. Clustering algorithms classify or partition a dataset into categories where there are similar traits or proximity among data points that belong to a given category according to a distance metric.

3.2.1. Distance measures

In order to identify the closeness of a particular customer to his/her direct calling neighbors, a similarity measure weighted by call duration or frequency between two phone numbers has been used. The weighted similarity measure is a first-order distance (Yan et al., 2005) defined as follows:

$$D_1(i,j) = \frac{1}{w_{i,j}}, \tag{1}$$

where $w_{i,j}$ is the call duration or frequency between customers (or phone numbers) i and j.

As described earlier, the direct calling neighbors of customers within the service provider's network are very sparse because most of the phone numbers are outside the service provider's network. Therefore, it is needed to include another distance measure, namely the second-order distance (Yan et al., 2005), which reveals relationships between customers according to their indirect calling patterns where two customers have common direct calling neighbors. In order to define the second-order distance between a pair of customers i and j, the set of phone numbers customers i and j called, respectively, in the specified time period, are weighted by calling duration or frequency:

$$N_w(i) = \{w_i(1), w_i(2), \ldots, w_i(n_i)\}, \tag{2}$$

where n_i is the total number of distinct phone numbers customer i called during the specified time period. Here, we normalize the value of calling duration or frequency, $w_i(k)$, such that $\sum_{k=1}^{n_i} w_i(k) = 1$. Finally, the second-order distance between two customers i and j is defined as follows:

$$D_2(i,j) = 1 - \frac{1}{2}\sum_{x \in N(i)\cap N(j)} w_i(k_i(x)) + w_j(k_j(x)), \tag{3}$$

where x is a common phone number which both customers i and j called during the specified time period, and $k_i(x)$ and $k_j(x)$ are the corresponding weights of common called phone number x in $N_w(i)$ and $N_w(j)$, respectively. According to Eq. (3), two customers are considered very close when they call some common numbers frequently (or heavily), regardless of how many other numbers they regularly call. The distance between a pair of customers is 1 (maximum possible distance) if they do not call any common phone number.

3.2.2. The employed clustering technique

Using the first- and second-order distance measures, a hierarchical clustering approach which works based on links between customers, can be applied to identify communities. A combined distance function needs to be defined in order to incorporate both the first- and second-order distance measures into the similarity measure of the clustering algorithm. The combined distance measure should assign a small distance to customers who make direct calls to each other frequently. Customers who are far away in context of the first-order distance, but still close through indirect calls should be assigned small distances as well. The distance function should assign higher distances to customers who are far from each other according to both the first and second similarity measures. One function that incorporates two distance functions into one combined distance measure that satisfies the above requirements is a linear combination of two individual distance measures. Accordingly, we have defined the following combined distance measure for our hierarchical clustering algorithm:

$$D(i,j) = (1-\alpha)D_1(i,j) + \alpha D_2(i,j). \tag{4}$$

The term α controls the degree of relevance of the first- and second-order distance measures and typically depends on the network characteristics of the mobile service provider. As $\alpha \to 0$, the similarity measure approaches the first- order distance measure. Intuitively, α captures the customers indirect calling neighbors. In other words, α is a weight factor that determines the impact of both first- and second-order measures in distance calculation. Eq. (4) makes it possible for the clustering algorithm to merge clusters with the most number of links, which are defined as the common neighbors of the customers based on both direct and indirect calling patterns. That is, the clustering algorithm identifies communities whose members have short first-order distance and large second-order distance between each other, as well as communities whose members have large first-order distance and short second-order distance between each other.

For discovering calling communities in this study, the agglomerative hierarchical clustering algorithm (Johnson, 1967; D'andrade, 1978) has been used. It is a bottom up clustering approach that investigates grouping in the given data by creating a cluster tree according to a particular distance measure. The tree is a multi-level hierarchy, where clusters at one level are grouped together to form clusters at the next higher level. This allows the user of the system to decide what level or scale of clustering is most appropriate for customers of a mobile service provider. The basic procedure to perform hierarchical clustering in our model is as follows:

1. Find the similarity and dissimilarity (including both first- and second-order distances) between every pair of customers in the CDR data: In this step, we calculate the distance between pair of objects using our similarity function defined in Eq. (4).
2. Create a cluster tree which groups the customers into a binary hierarchical tree: In this step, pairs of customers that are in close proximity are linked together using a linkage function. The proximity of customers to each other is measured using the distance information generated in step 1. A hierarchical tree is then formed as newly formed binary clusters are recursively grouped into larger clusters.

3. Decide on the level to divide the hierarchical tree into clusters: In this step, we divide the customers in the hierarchical tree into clusters using a clustering function. Our clustering function can create clusters by detecting natural groupings in the hierarchical tree by considering the accuracy of a classifier, which is built based on the obtained clusters.

3.3. Classification

Data objects stored in a database are identified by their attributes, and the main idea of classification (Han & Kamber, 2000) is to explore through data objects (training set) to find a set of rules which determine the class of each object according to its attributes. These rules are later used to build a classifier to predict the class or missing attribute value of unseen objects whose class might not be known. As it is obvious from the description, classification consists of two steps. In the first step, which is also known as the supervised learning process (i.e., the class labels of the training samples are provided), the aim is to build a model that describes the characteristics of data classes and attributes of the training set by generating a set of rules. In the second step (the testing process), the model is run over a test set to predict the classes of objects or data whose classes are not provided.

Classification is beneficial for several applications, such as preventing theft and saving lives, medical diagnosis, increasing revenue and market analysis, better decision making, predicting customer behavior, etc. An attractive real world application of classification is fraud-detection. The training set includes complete records of both valid and fraudulent activities, each classified as either valid or fraudulent activity. The classifier training algorithm uses these pre-classified examples to generate a set of rules and parameters required for building a classifier model. Once the classifier is built, it is applied to a test set of classified samples to determine the quality of the model. The quality and efficiency of the model are usually evaluated by a combination of statistical techniques and domain expertise. When the quality of the classifier is approved, it is used to discriminate between the validity and fraudulent of future data.

A more formal definition of the classification problem may be stated as follows (Dunham, 2000). Let D be a database of tuples $\{T_1,T_2,\ldots,T_n\}$ and $C = \{C_1,C_2,\ldots,C_m\}$ be the set of class labels. The classification problem is a mapping f: $D \rightarrow C$, where each tuple T_i in D belongs to one and only one of m classes. A class, C_j contains precisely those tuples mapped to C_j, i.e., $C_j = \{T_i | f(T_i) = C_j, 1 \leqslant i \leqslant n, \wedge T_i \in D\}$. This mapping function can then be used to predict the classes of the tuples in a testing set. On the other hand, in fuzzy classification a tuple may belong to more than one class based on the degree of membership determined by evaluating a corresponding function. The former type of classification is considered in this paper.

As the study described in this paper is concerned, after identifying user communities, it is time to transform the clusters into a classifier model which is able to predict a community where a new customer belongs to according to his/her calling patterns. We use the community of a customer as his/her class label and then we derive several features using the information extracted from calling links and clusters. Finally, we create a training set in which every column represents a distance feature and every row represents values for the features of a particular customer.

In a classification task, every data item is modeled by a feature vector. A feature vector consists of values that represent the key properties of the data item to be classified. In our customer classification problem, for every customer within the service provider's network, a feature vector that represents the key characteristics of a particular customer so as her/his relationship with all other customers within the existing communities is built. Then, a set of feature vectors, each of which corresponds to a specific customer, is used as the training set for the learning algorithm. There exist many classification algorithms for supervised learning. The performance of different learning algorithms strongly depend of the classification domain and dataset. In this study, we have used seven different machine learning techniques for the classification task.

3.3.1. Feature set

Hasan et al. (Hasan, Chaoji, Salem, & Zaki, 2006) have categorized the features appropriate for link prediction in social networks into three categories: proximity, aggregated and topological features. In the context of our problem, we have found aggregated and topological features to be useful for modeling the characteristics of a particular customer individually, and also in relation with existing clusters. Therefore, the following input features have been constructed for each customer from his/her calling neighbors based on the first- and second-order distances. The features are defined as follows:

- Total number of customer's direct calling neighbors: this is an aggregated feature that describes if a particular customer is socially connected to many other customers through direct links.
- Percentage of a customer's calls made to her/his closest direct calling neighbor: this aggregated feature reveals if a customer is a socialize member of a network by not making most of the phone calls to her/his closest direct neighbor.
- Percentage of a customer's direct calling neighbors which are within the service provider's network: this aggregated feature describes how likely a user is to call customers inside her/his home network. In other words, it shows if a particular customer has many social conductivities within the home service network.
- Percentage of a customer's direct calling neighbors which are outside of the service provider's network: this aggregated feature describes how likely a user is to call customers outside her/his home network. In other words, it shows if a particular customer has many social conductivities outside the home service network.
- The shortest distance of a customer to every class (community): this topological feature represents the closeness of a particular customer to all existing classes. We have considered the distance between a particular customer and her/his closest peer from every community as the shortest distance between the customer and the community where that peer belongs. Eq. (4) has been used to measure the distance between a particular customer and her/his calling peers. If a customer does not have any calling peer in a particular community, then she/he is far apart from that community and has the maximum possible distance from that particular community. In other words, the advantage of using Eq. (4) in distance calculation is that it incorporates both the first- and second-order distances which give, in turn, weights to the calling links based on the frequency and duration of the calls.
- Percentage of direct calls to neighbors within every class (community): this is an aggregated feature that measures the social connections of a particular customer to different classes (communities) in terms of direct calling neighbors.
- Percentage of indirect calling neighbors within every class (community): this aggregated feature quantifies the social connections of a particular customer to different classes (communities) with respect to indirect calling neighbors.

3.3.2. Machine learning techniques for classification

There exist many classification algorithms for supervised learning. The performance of different learning algorithms strongly depends on the classification domain and dataset. We use supervised machine learning techniques for building the classifier model for our problem. Supervised ML algorithms have been widely used for many classification problems. A classification algorithm explores among a training set of sample instances, each of which belongs to a predefined class, to discover a set of rules that relate every data sample to its class label. This set of rules, called classifier model, is then able to predict the class label of an unknown data item by examining its feature values. In this study, we have used the following algorithms to build the classifier model.

k-Nearest Neighbor:

The k-nearest neighbor (KNN) algorithm (Cover & Hart, 1967) is one of the simplest machine learning algorithms. A new instance is classified by a majority vote of its neighbors, i.e., the instance is assigned to the dominant class among its KNNs; k is a positive integer, typically small. The simplest case of KNN is when $k = 1$ (nearest neighbor), then the new instance is simply assigned to the class of its nearest neighbor. In order to measure the closeness of neighbors, a distance function such as Euclidean distance or Manhattan distance is used. The smaller the distance between two instances is, the closer neighbors they are. The main difference between KNN algorithm and other ML techniques is that no explicit training step is required.

Naive Bayes:

A naive Bayes classifier is a simple probabilistic classifier that applies Bayes' theorem (John & Langley, 1995) with strong independence assumptions, i.e., given the class of the example, the values of the attributes of an example are assumed to be independent. Although this is a very rough assumption in real world application, recent work has shown the effectiveness of naive Bayes classifier in practice. Formally, let F_1 through F_n be feature variables and C be a dependent class variable with a small number of classes. Given a new instance with feature values v_1 through v_n, the class variable is set to c such that $P(C = c|F_1 = v_1 \wedge \ldots \wedge F_n = v_n)$ is maximal. Using Bayes theorem this probability is calculated as follows:

$$P(C = c|F_1 = v_1 \wedge \ldots F_n = v_n) = \frac{P(C = c)P(F_1 = v_1 \wedge \ldots \wedge F_n = v_n|C = c)}{P(F_1 = v_1 \wedge \ldots \wedge F_n = v_n)}. \qquad (5)$$

In practice, we are only interested in the numerator of Eq. (5), since the denominator does not depend on C, and the values of the features F_i are given, so it does not play any role in the class decision making process. Further, the numerator is equivalent to the join probability model and can be rewritten as follows:

$$P(C = c, F_1, \ldots, F_n) = P(C)\prod_{i=1}^{n} P(F_i|C = c). \qquad (6)$$

Finally, under the independence assumptions and using Eq. (6), the conditional distribution over the class variable C can be defined as follows:

$$P(C = c|F_1 = v_1 \wedge \ldots \wedge F_n = v_n) = \frac{1}{z}P(C = c)\prod_{i=1}^{n} P(F_i|C = c), \qquad (7)$$

where z is a constant for every instance from the training set since the feature variables are known. A new instance is assigned to a class that gives the highest conditional probability. Finally, it is worth mentioning that the naive Bayes classifier is the fastest learning algorithm among all ML techniques.

Decision Tree:

A decision tree is a tree whose internal nodes are arrangements of appropriate tests at every step in an analysis and whose leaf nodes are classes. In general, each path from the tree root to a leaf is a conjunction of attribute tests, and the tree itself is a disjunction of these conjunctions. More specifically, decision trees classify instances by sorting them down the tree starting from the root node down to some leaf node; the latter node provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the decision tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the current value of the attribute. This process is then repeated at the reached node on this branch, and so on, until a leaf node is reached. The decision tree construction algorithm can be summarized as follows.Given a set of training samples, $S = \{s_1, s_2, \ldots, s_m\}$, such that each sample $s_i = x_1, x_2, \ldots, x_n$ is a vector, where $x_1, x_2, \ldots, x_n$ represent attributes or features of the sample; and let $C = \{c_1, c_2, \ldots, c_n\}$ be the possible class labels for the available samples.

1. Split S into smaller partitions by performing a test on one or more attributes as required.
2. Check the split results:
   If all records in every particular partition belong to the same class C_i, all partitions are considered to be pure: label the leaf node with the class label C_i and stop.
   Else: recursively (go to step 1) split partitions that are not pure.

There are several techniques for constructing the decision tree such as ID3, C4.5 (Quinlan, 1993) and CART (Breiman, Friedman, Olshen, & Stone, 1984). In this study, we use the C4.5 algorithm to construct the decision tree. C4.5 builds a decision tree from a set of training data using the concept of Information Entropy to choose features with the most information gain. Assume that S contains s_i tuples of class c_i, then the information required to classify any arbitrary tuple is calculated by

$$I(s_1, s_2, \ldots, s_n) = -\sum_{i=1}^{n} \frac{s_i}{S}\log_2\left(\frac{s_i}{S}\right). \qquad (8)$$

Now assume that an attribute A with values $\{a_1, a_2, \ldots, a_v\}$ is selected as a candidate root node of a given tree, and then S will be split into sets $\{s_1, s_2, \ldots, s_v\}$. The expected information required to construct the tree with A as the root is

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{nj}}{S} I(s_{1j}, \ldots, s_{nj}). \qquad (9)$$

Eventually, the attribute with the highest information gain is chosen to branch the given tree. The information gained by branching on attribute A is calculated by

$$\mathrm{Gain}(A) = I(s_1, s_2, \ldots, s_n) - E(A). \qquad (10)$$

Multilayer perceptron:

The multilayer perceptron (MLP) belongs to the family of supervised neural networks algorithms (Bishop, 1995; Haykin, 1998). The MLP model consists of a network of nodes arranged in layers. A node (neuron) is a processing element whose output is a combination of multiple inputs from other neurons. The architecture of the multilayer perceptron consists of an input layer, an output layer, and multiple hidden layers. Input layer presents the variable values to the network. Every input neuron represents an input variable value. The value from each input neuron is multiplied by a weight, and the resulting weighted values are added together producing an input for a neuron in the hidden layer. A transfer function is applied to this weighted sum within the hidden layer neuron and generates an output. The outputs from the hidden layer neurons are distributed to the next layer such that the value from each hidden layer neuron is multiplied by a weight; the resulting weighted values are added together producing another combined value. A transfer function is applied to the weighted sum and generates an output. When there is more than one hidden layer, the output from one hidden layer is the input into the next hidden layer, otherwise it is the input to the output layer. The outputs of the output layer neurons are the network outputs.The task of the learning algorithm is to adjust the set of weight values applied in different layers so that the outputs from the MLP match the actual target values as closely as possible. The weight adjustment is performed by using the standard backpropagation algorithm. To adjust the weights, backpropagation algorithm uses input pattern flows forward through the network, and the error signal flows backward. In the forward pass, the predicted outputs corresponding to the given inputs are evaluated. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The network weights can then be adjusted using a gradient-based optimization algorithm. The above process is iterated until the weights converge.

Bayesian network:

A Bayesian network (Pearl, 1985) is a graphical probabilistic model that consists of a directed acyclic graph and a set of conditional probability tables. The nodes in the network represent features or variables and links encode the conditional independence between the variables. The weight of the links is determined using the conditional probability tables. The probability distribution for each node, given its parent nodes, is defined by a probability table. The probability distribution is unconditional for a node without any parents. If a node has one or more parents the probability distribution is a conditional distribution, where the probability of each node value depends on the values of the parents. The learning process in a Bayesian network consists of two stages. First, the network structure is built (structure learning) and then probability distribution estimations are calculated in form of probability tables. Structure learning often has high computational complexity as the number of possible structures is huge. To solve the computational complexity, heuristic and approximate learning algorithms have been proposed (Cooper & Herskovits, 1992; Heckerman, Geiger, & Chickering, 1995). There are many combinations of structure learning and search technique that can be used to create Bayesian networks.

Support vector machines:

Support vector machines (SVM) are known as a powerful technique for classification problems (Cristianini & Taylor, 2000; Vapnik, 1998). The goal of SVM is to construct a separating hyperplane that is maximally distant from different classes of the training data (optimal separating hyperplane). Basically, SVM can be used to solve binary classification problems. However, in order to use SVM for real world classification tasks, the idea has been extended for multi-class problems as well. The extension can be done either during the learning process or during the decision process. One-vs-rest and adaptive code algorithm are two of the most well-known extensions of SVM to multi-class problems (Allwein, Schapire, & Singer, 2000; Rätsch, Smola, & Mika, 2002). The formal definition of linear SVM can be described as follows (Cristianini & Taylor, 2000; Vapnik, 1998). Let the training data of two separable classes with n samples be represented by $(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), \ldots, (\vec{x}_n, \vec{y}_n)$, $i = 1, 2, \ldots, n$, where $\vec{x} \in R^N$ is an N dimensional space (feature vector) and $y_i \in \{-1, +1\}$ is the class label, maximize:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\vec{x}_i^{\mathrm{T}} \cdot \vec{x}_j) \tag{11}$$

under the constraints $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $\alpha_i \geqslant 0$, $i = 1, 2, \ldots, n$. The optimal separating hyperplane can be found from the solutions α_i's to this maximization problem. A soft-margin algorithm as an extension of the basic algorithm is available, when the instances are not linearly separable.

4. Experimental analysis

This section is dedicated to describe the evaluation criteria and the conducted experiments. We summarize the experimental results and highlight the performance and applicability of the system.

4.1. Evaluation criteria

The MATLAB Statistics Toolbox has been used for building the hierarchical cluster tree. After linking the customers of the service provider network into a cluster tree, it is necessary to decide how to cut the tree at an appropriate level to generate valid clusters. For the community identification problem, the validity of the clusters produced by a clustering algorithm is an important consideration since once the clustering is complete, each of the clusters is to be labeled and then used in the classification task. Because the number of generated clusters is subjective to the accuracy of the classifier, the following approach has been used to determine the validity of the clustering. First, in order to determine the initial number of clusters, the cluster tree has to be cut at a certain level. There are different approaches to decide on the level where to have this cut. The approach that we have used in our study to determine the natural clustering is to compare the height of each link in the generated cluster tree with the heights of neighboring links below it in the tree. If a link has approximately the same height as neighboring links, it indicates that there are similarities between the objects joined at this level of the hierarchy. These links are said to exhibit a high level of consistency. If the height of a link differs from neighboring links, it indicates that there are dissimilarities between the objects at this level in the cluster tree. This link is considered to be inconsistent with the links around it. In cluster analysis, inconsistent links can indicate the border of a natural division in a data set. Second, to ensure the validity of the initial clusters, we compare the accuracy of the classifiers built based on different cluster divisions. That is, we labeled the initial clusters and build a classifier using a learning algorithm. At the next iteration, we increase the number of clusters by 5 and we divide the cluster tree such that it generates that particular number of clusters and we build new classifiers. We repeat this experiment several times to ensure that increasing the number of clusters further does not result in better classifiers. Here it is worth mentioning that this does not indicate that the clusters generated by a lower cut are not valid. However, by increasing the number of clusters, it is hard for the learning algorithms to build accurate classifiers. The second stage of the above approach helps to find valid clusters that will also result in accurate classifiers. We have used the following metrics to evaluate the performance of our classifiers:

- *Accuracy:* is the fraction of correctly classified instances over the total number of training instances.
- *Precision:* indicates the fraction of the detected class members that are actually correct, i.e., the number of class members classified correctly over the total number of instances classified as class members.
- *Recall:* indicates the probability of correctly detecting test samples of class members, i.e., the number of class members classified correctly over the total number of class members.

- *F-measures:* indicates the weighted harmonic mean of precision and recall.

There are many techniques that can be used to evaluate the above measures for classifier models. In this study, we use the well-known cross validation technique. Basically, *n*-fold cross validation is a method in which the data is randomly divided into *n* disjoint groups (Edelstein, 1999). For example, suppose the data is divided into ten groups. The first group is set aside for testing and the other nine are put together for model building. The model built on the 90% group is then used to predict the group that was set aside. This process is repeated a total of 10 times as each group in turn is set aside. Finally, a model is built using all the data. The mean of the 10 independent error rate predictions is used as the error rate for the final model. In our experiments, for all the algorithms, 5-fold cross validation have been used for the reported results. For all learning algorithms, WEKA, a well known machine learning library (WEKA), was used. For algorithms that have parameters need to be set by users, like SVM, we conducted experiments with more than one variation and reported the result that showed the best performance.

The runtime of different learning algorithms is also an important consideration because the model building phase is computationally expensive (time consuming). For the analysis, all operations are performed on a Dell Optiplex 745 with an Intel Core2 Duo 6600 @ 2.4 GHz processor and 4 GB of RAM. The number of data instances in the training set is 10200.

4.2. Experimental results

The target of the first set of experiments is to evaluate the impact of the α threshold on the formation of the cluster tree. Using the CDR data, a hierarchical cluster tree has been built by varying the α value in the range [0,1] in steps of 0.1. In other words, we start the experiment by completely ignoring the second-order distance while calculating the similarity between customers. We incrementally give more weight to the second-order distance by increasing the α value. In order to measure the validity of the cluster tree generated using our similarity measure, it is compared with the original proximity data generated by the distance function. If the clustering is valid, the linking of objects in the cluster tree should have a strong correlation with the distances between objects in the distance vector. In order to compare these two sets of values and compute their correlation, we have used an index validity called *Cophenetic correlation coefficient* (Romesburg, 2004), which refers to the correlation between the actual entity correlations in the similarity measure and the predicted values from the dendrogram. The closer the calculated value of the cophenetic correlation coefficient to 1, the better is the clustering solution. As can be seen in Fig. 1, we achieve the highest cophenetic correlation coefficient value when α is set to 0.2. This indicates that for our dataset this value of α results in the best cluster tree. It also highlights the effectiveness of our proposed similarity measure that combines both the first- and second-order distances to calculate the closeness of customers. The cluster tree generated based on our similarity measure discovers indirect relationships among customers; this can not be observed by just looking at calling detail records.

The dendrogram obtained from cluster analysis with $\alpha = 0.2$ is displayed in Fig. 2. It shows customers in a hierarchical manner, with similar neighbors close together and dissimilar customers placed farther apart. In our experiments, according to the resulted tree, we initially cut the tree at a level where it generates 15 clusters. Then, to validate the goodness of the first cut in terms of building classifiers, we increase the number of clusters in steps

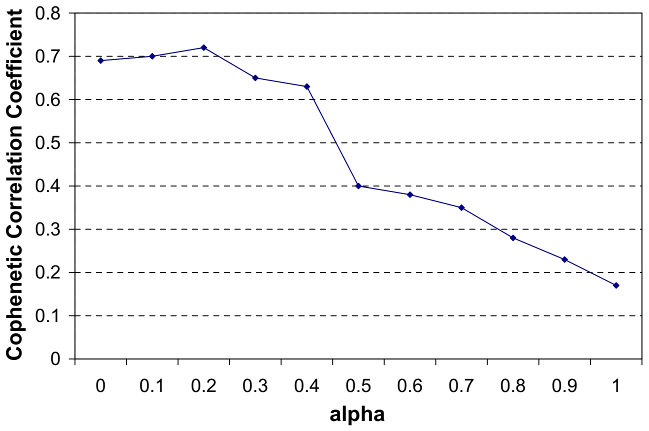

Fig. 1. Cophenetic correlation coefficient as a function of α.

of 5. To further analyze the effectiveness of the clustering algorithm and also to build a classifier model, we label each customer with her/his community and use the labeled customers as the training set for the learning algorithms. The performance of classifier models built by learning algorithms determine how well the clustering algorithm is able to create communities that contain customers with similar behaviors.

Fig. 3 shows the average accuracy obtained by using all the learning algorithms incorporated in this study, while different number of clusters are tested. As can be seen from Fig. 3, the best classifiers are obtained from clusters of the initial cut in the cluster tree that generates 15 clusters. Although increasing the number of clusters might generate more precise groups that better represent customers with very similar behavior, it makes the task of classification very challenging for learning algorithms since the number of classes increases. However, increasing the size of the training set might help to generate more accurate classifiers.

In order to measure the average time required to build a classifier for calling communities identification, different number of clusters have been used to label the groups and clusters are transferred to a classifier using the selected learning algorithms. Then, the average build time for every learning algorithm has been reported. As can be seen from Fig. 4, decision tree, naive Bayes and Bayes network are among the fastest learning algorithms that only need less than a second to build a classifier given our dataset. MLP has shown the slowest building time among others in our experiments. The KNN algorithm has not been presented in Fig. 4 since it does not require an explicit training step. Although the classifier can be built once and be used later, shorter build times may be more preferable for many real world application domains such as telecommunication.

Table 1 shows the performance comparison for different classifiers on the training set generated by the initial cut in the cluster tree. Recall that the initial cut has generated 15 clusters and resulted in classifiers with the best average accuracy in our primary experiment. The majority of algorithms perform well with the given CDR data, reporting reasonable accuracy. This indicates that the features that we have constructed according to the generated clusters have strong discriminating ability.

In context of accuracy metrics, linear SVM, KNN and Bayes Network performed the best with an accuracy of 79.52%, 79.31% and 79.30%, respectively. Such small differences in the reported accuracies make it very challenging to pick the best classifier for communication identification. Other classifiers built by SVM with RBF kernel function, decision tree and multi-layer perceptron also have comparable performance with the above classifiers. Among all the

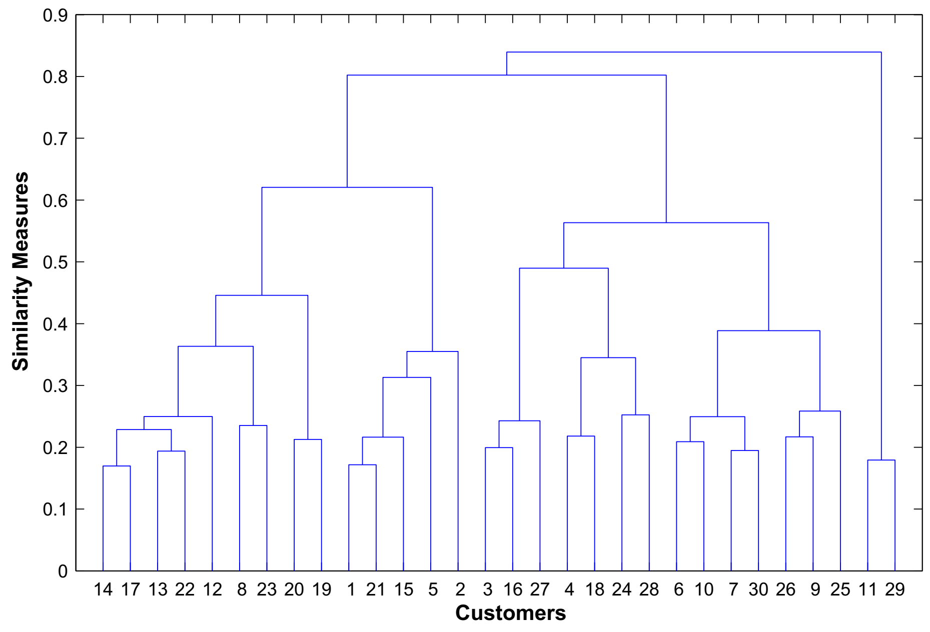

Fig. 2. Dendrogram resulting from cluster analysis of calling neighbors ($\alpha = 0.2$).

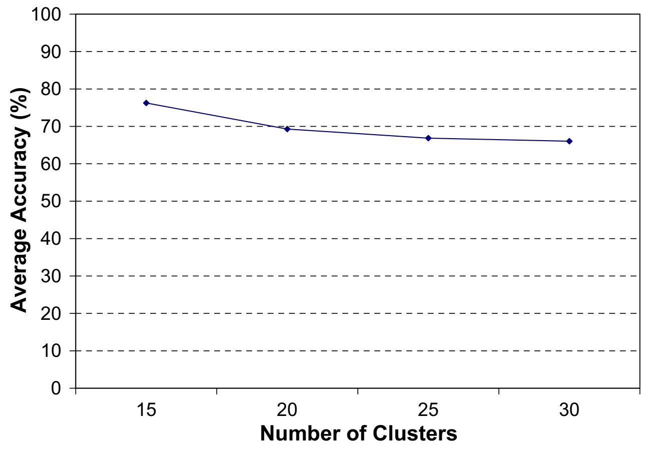

Fig. 3. Average accuracy by number of clusters.

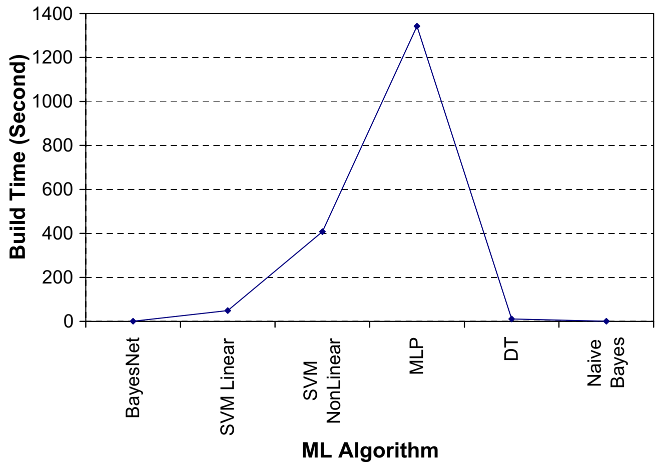

Fig. 4. Average time taken to build the classifier model for ML algorithms.

Table 1
Performance of different learning algorithms when the number of clusters is 15

ML algorithm	Accuracy	Precision	Recall	*F*-measure	Build time (s)
BayesNet	79.30	72.53	69.96	70.85	0.44
SVM linear	79.52	74.53	67.96	70.81	28.47
SVM nonlinear	74.65	82.51	49.34	52.03	441.47
MLP	78.23	72.00	65.33	68.13	631.43
DT	77.14	73.72	64.87	68.58	6.77
Naive Bayes	65.68	54.77	66.89	55.83	0.30
KNN	79.31	78.35	65.21	70.20	NA

classifiers, naive Bayes performs the worst and is not able to extract useful patterns for building an accurate classifier. Therefore, naive Bayes might not be the one that is appropriate for the automated communities identification.

We have also reported other performance metrics such as Precision, Recall and *F*-measure for the positive class. The reported value for each measure is the metric average for all classes. Even considering the above metrics, the order of the classifiers do not considerably change. This shows that the precision-recall behavior of all the models are almost similar.

5. Summary and conclusions

In this paper, we demonstrated how cluster analysis can be used to effectively identify calling communities by using information derived from the CDR data. We have proposed a similarity measure that combines both the first- and second-order distances for cluster formation. The agglomerative hierarchical clustering approach has been applied for the clustering task. We use the information extracted from the cluster analysis to identify customer calling patterns. We have constructed a variety of features in order to represent the customer behavior within her/his own cluster as well as toward other clusters. Customers calling patterns are then given to a classification algorithm to generate a classifier model for predicting the calling communities of a customer. Different machine learning algorithms have been evaluated for automated communities identification in terms of performance and the classifier build time. Our work is especially important for targeted marketing campaigns in the telecommunication industry since the CDR data is often the only primary data source available for the customers. Based on the assumption that customers in the same calling community might behave similarly, targeted efforts can be focused on for certain communities. Further, by identifying the community leader, marketing efforts can be directed to the community leaders since they are believed to have influence on other community members' behavior.

In the context of automated calling communities identification, the type of the classifier that is to be used to identify calling communities and customers' value is very crucial. For instance, in a marketing campaign, selecting inappropriate customers is very costly for verifying the impact of the marketing campaign when it is intended to modify the strategy toward the target group. The fuzzy classification is one possible approach that offers more convenience for selecting customer subgroups and for measuring the efficiency and validity of the communities regarding the marketing campaign design. As the next step, we would like to build a flexible fuzzy classifier system that takes advantage of the application of membership functions to increase or decrease the homogeneity between the targeted customers depending on whether the proposed products are very specific or intended for a large community.

References

Allwein, E., Schapire, R., & Singer, Y. (2000). *Reducing multiclass to binary: A unifying approach for margin classifiers*. AT&T Corp.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees. Belmont, CA.

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–347.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21–27.

Cristianini, N., & Taylor, J. S. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press.

D'andrade, R. (1978). U-statistic hierarchical clustering. *Psychometrika, 4*, 58–67.

Dunham, M. H. (2000). *Data mining techniques and algorithms*. Englewood Cliffs, NJ: Prentice-Hall.

Edelstein, H. A. (1999). *Introduction to data mining and knowledge discovery* (3rd ed.). Potomac, MD: Two Crows Corp.

Flake, G. W., Lawrence, S. & Giles, C. L. (2000). Efficient identification of web communities. In *Proceedings of ACM international conference on knowledge discovery and data mining* (pp. 150–160).

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences, 99*(12), 7821–7826.

Han, J., & Kamber, M. (2000). *Data mining concepts and techniques*. San Mateo, CA: Morgan Kanufmann.

Hasan, M., Chaoji, V., Salem, S. & Zaki, M. J. (2006). Link Prediction using supervised learning. In *Proceedings of the workshop on link analysis, counter-terrorism and security (in conjunction with SIAM data mining conference)*, Bethesda, MD, April 2006.

Haykin, S. (1998). *Neural networks: A comprehensive foundation*. Englewood Cliffs, NJ: Prentice-Hall.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*, 197–243.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the international conference on uncertainty in artificial intelligence* (pp. 338–345). San Mateo, CA: Morgan Kaufman.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika, 2*, 241–254.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM, 46*(5), 604–632.

Nasrullah, M., & Larsen, H. L. (2006). Structural analysis and mathematical methods for destabilizing terrorist networks. In *Proceedings of the international conference on advanced data mining applications*. *Lecture notes in artificial intelligence (LNAI 4093)* (Vol. 4093, pp. 1037–1048). Berlin: Springer.

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the conference of the cognitive science society* (pp. 329–334). Irvine, CA: University of California. August 15–17.

Quinlan, J. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Rätsch, G., Smola, A. J. & Mika, S. (2002). Adapting codes and embeddings for polychotomies, *RSISE, CSL, Machine Learning Group*, Canberra, Australia and Frauenhofer FIRST, Berlin, Germany.

Romesburg, H. C. (2004). *Cluster analysis for researchers*. Lulu Press.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley. 732 pp.

WEKA 3.4.4, Available from <http://www.cs.waikato.ac.nz/ml/weka/>.

Yan, L., Fassino, M. & Baldasare, P. (2005). Predicting customer behavior via calling links. In *Proceedings of IEEE international joint conference on neural networks* (Vol.4, pp. 2555–2560).