ISSN 2229-5518

Mining knowledge using Decision Tree Algorithm

Mrs. Swati .V. Kulkarni

Abstract— Industry is experiencing more and more competitions in recent years. The battle is over their most valuable customers. With massive industry deregulation across the world, each customer is facing an ever-growing number of choices in telecommunications and financial services. As a result, an increasing number of customers are switching from one service provider to another. This phenomenon is called customer "churning" or "attrition," which is a major problem for these companies and makes it hard for them to stay profitable. The data sets are often cost sensitive and unbalanced. If we predict a valuable customer who will be an attritor as loyal, the cost is usually higher than the case when we classify a loyal customer as an attritor. Similarly, in direct marketing, it costs more to classify a willing customer as a reluctant one. Such information is usually given by a cost matrix, where the objective is to minimize the total cost. In addition, a CRM data set is often unbalanced; the most valuable customers who actually churn can be only a small fraction of the customers who stay. The main focus is on the output of decision tree algorithms as the input to post processing algorithms. This algorithm relies on not only a prediction, but also a probability estimation of the classification, such as the probability of being loyal. Such information is available from decision trees.

Index Terms— Attritor, Churning, CRM, cost matrix, decision tree, post processing algorithm, ROC curve.

1 INTRODUCTION

Extensive research in data mining[1] has been done on discovering distributional knowledge about the underlying data. Models such as Bayesian models, decision trees, support vector machines, and association rules have been applied to various industrial applications such as customer relationship management (CRM) [2]. Despite such phenomenal success, most of these techniques stop short of the final objective of data mining, which is to maximize the profit while reducing the costs, relying on such post processing techniques as visualization and interestingness ranking. While these techniques are essential to move the data mining result to the eventual applications, they nevertheless require a great deal of human manual work by experts. Often, in industrial practice, one needs to walk an extra mile to automatically extract the real "nuggets" of knowledge, the actions, in order to maximize the final objective functions. In this paper, a novel post processing technique is presented to extract actionable knowledge from decision trees. To illustrate my motivation, customer relationship management CRM is considered, in particular, the Mobile communications industry is taken as an example [3]. This industry is experiencing more and more competitions in recent years. With massive industry deregulation across the world, each customer is facing an ever-growing number of choices in communications and financial services. As a result, an increasing number of customers are switching from one service provider to another. This phenomenon is called customer "churning" or "attrition," which is a major problem for these companies and makes it hard for them to stay profitable. In addition, a CRM data set is often unbalanced; the most valuable customers who actually churn can be only a small fraction of the customers who stay. In the past, many researchers have tackled the direct marketing problem as a classification problem, where the cost-sensitive and unbalanced nature of the problem is taken into account. In management and marketing sciences, stochastic models are used to describe the response behavior of customers. In the data mining area, a main approach is to rank the customers by the estimated likelihood to respond to direct marketing actions and compare the rankings using a lift chart or the area under curve measure from the ROC curve.

Like most data mining algorithms today, a common problem in current applications of data mining in intelligent CRM is that people tend to focus on, and be satisfied with, building up the models and interpreting them, but not to use them to get profit explicitly.

In this paper, a novel algorithm for post processing decision trees is presented to obtain actions that are associated with attribute-value changes, in order to maximize the profit-based objective functions. This allows a large number of candidate actions to be considered, complicating the computation. More specifically, in this paper, two broad cases are considered. One case corresponds to the unlimited resource situation, which is only an approxima-

tion to the real-world situations, although it allows a clear introduction to the problem. Another more realistic case is the limited-resource situation, where the actions must be restricted to be below a certain cost level. In both cases, the aim is to maximize the expected net profit of all the customers as well as the industry. It can be shown that finding the optimal solution for the limited resource problem and designing a greedy heuristic algorithm to solve it efficiently is computationally hard [4]. An important contribution of the paper is that it integrates data mining and decision making together, such that the discovery of actions is guided by the result of data mining algorithms (decision trees in this case)[5]. An approach is considered as a new step in this direction, which is to discover action sets from the attribute value changes in a non sequential data set through optimization. The rest of the paper is organized as follows:

First a base algorithm is presented for finding unrestricted actions in Section 2. Then formulate two versions of action-set extraction problems, and show that finding the optimal solution for the problems is computationally difficult in the limited resources case (Section 3). then greedy algorithms are efficient while achieving results very close to the optimal ones obtained by the exhaustive search (which is exponential in time complexity). A case study for Mobile handset manufacturing and selling company is discussed in section 4. Conclusions and future work are presented in section 5.

This is the unlimited-resource case. The data set consists of descriptive attributes and a class attribute. For simplicity, discrete-value problem is considered in which the class values are discrete values. Some of the values under the class attribute are more desirable than others. For example, in the banking application, the loyal status of a customer "stay" is more desirable than "not stay". The overall process of the algorithm can be briefly described in the following four steps:

1. Import customer data with data collection, data Cleaning, data preprocessing, and so on.

2. Build customer profiles using an improved decision tree learning algorithm from the training data. In this case, a decision tree is built from the training data to predict if a customer is in the desired status or not. One improvement in the decision tree building is to use the area under the curve (AUC) of the ROC curve to evaluate probability estimation (instead of the accuracy). Another improvement is to use Laplace correction to avoid extreme probability values.

3. Search for optimal actions for each customer (see Section 2 for details).

4. Produce reports for domain experts to review the actions and selectively deploy the actions.

In the next section, we will discuss the leaf-node Search algorithm used in Step 3 (search for optimal actions) in detail.

2 Leaf-Node Search in the Unlimited Resource Case:

First consider the simpler case of unlimited resources where the case serves to introduce computational problem in an intuitive manner. The leaf-node search algorithm searches for optimal actions to transfer each leaf node to another leaf node with a higher probability of being in a more desirable class.

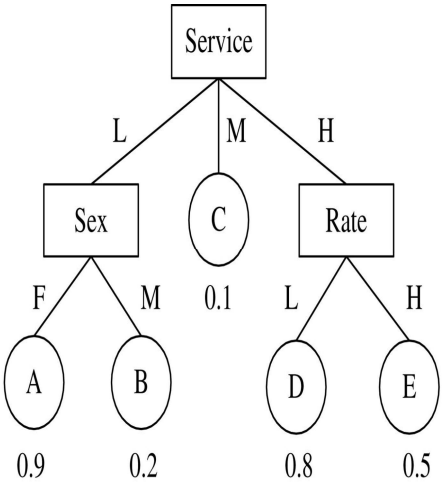


Fig. 1 An example of customer profile.

After a customer profile is built, the resulting decision tree can be used to classify, and more importantly, provide the probability of customers in the desired status such as being loyal or high-spending. When a customer, who can be either a training example used to build the decision tree or an unseen testing example, falls into a particular leaf node with a certain probability of being in the desired status, the algorithm tries to "move" the customer into other leaves with higher probabilities of being in the desired status. The probability gain can then be converted into an expected gross profit. However, moving a customer from one leaf to another means some attribute values of the customer must be changed. This change, in which an attribute A's value is transformed from v1 to v2, corresponds to an action. These actions incur costs. The cost of all changeable attributes is defined in a cost matrix (see Section 2.3) by a domain expert. The leaf-node search algorithm searches all leaves in the tree

so that for every leaf node, a best destination leaf node is found to move the customer to. The collection of moves is required to maximize the net profit, which equals the gross profit minus the cost of the corresponding actions. Based on a domain-specific cost matrix (Section 2.3) for actions, we define the net profit of an action to be as follows:

P Net = PE X P gain – cost I

Where P Net denotes the net profit, PE denotes the total profit of the customer in the desired status, P gain denotes the probability gain, and cost I denotes the cost of each action involved.

In Section 3.1, this definition is extended to a formal definition of the computational problem.

The leaf-node search algorithm for searching the best actions can thus be described as follows:

Algorithm leaf-node search

1. For each customer x, do
2. Let S be the source leaf node in which x falls into;
3. Let D be a destination leaf node for x the max net profit P net
4. Output (S,D, P net).

To illustrate, consider an example shown in Fig. 1, which represents an overly simplified, hypothetical decision tree as the customer profile of loyal customers built from a bank. The tree has five leaf nodes (A, B, C, D, and E), each with a probability of customers' being loyal. The probability of attritors is simply 1 minus this probability. Consider a customer Jack who's record states that the Service = Low (service level is low), Sex = M (male), and Rate = L (mortgage rate is low). The customer is classified by the decision tree. It can be seen that Jack falls into the leaf node B, which predicts that Jack will have only a 20 percent chance of being loyal (or Jack will have an 80 percent chance to churn in the future). The algorithm will now search through all other leaves (A, C, D, and E) in the decision tree to see if Jack can be "replaced" into a best leaf with the highest net profit.

1. Consider leaf A. It does have a higher probability of being loyal (90 percent), but the cost of action would be very high (Jack should be changed to female), so the net profit is a negative infinity.
2. Consider leaf C. It has a lower probability of being loyal, so the net profit must be negative, and we can safely skip it.
3. Consider leaf D. There is a probability gain of 60 percent (80 percent - 20 percent) if Jack falls into D. The action needed is to change Service from L (low) to H (high). Assume that the cost of such a change is $200 (given by the bank). If the bank can make a total profit of $1,000 from Jack when he is 100 percent loyal, then this probability gain (60 percent) is converted into $600 (1000 X 0.6) of the expected gross profit. Therefore, the net profit would be $400 (600 -200).
4. Consider leaf E. The probability gain is 30 percent (50 percent - 20 percent), which transfers to $300 of the expected gross profit. Assume that the cost of the actions (change Service from L to H and change Rate from L to H) is $250, then the net profit of moving Jack from B to E is $50 (300 - 250). Clearly, the node with the maximal net profit for Jack is D, with suggested action of changing Service from L to H.

2.1 Cost Matrix

In the discussion so far, we assume that attribute-value changes will incur costs. These costs can only be determined by domain knowledge and domain experts. For each attribute used in the decision tree, a cost matrix is used to represent such costs. In many applications, the values of many attributes, such as sex, address, number of children, cannot be changed with any reasonable amount of money. Those attributes are called "hard attributes." For hard attributes, users must assign a very large number to every entry in the cost matrix. If, on the other hand, some value changes are possible with reasonable costs, then those attributes such as the Service level, interest rate, and promotion packages are called "soft attributes". Note that the cost matrix needs not to be symmetric. One can assign $200 as the cost of changing service level from low to high, but infinity (a very large number) as the cost from high to low, if the bank does not want to "degrade" service levels of customers as an action. For continuous attributes, such as interest rates that can be varied within a certain range, the numerical ranges can be discretized first using a number of techniques for feature transformation.

3 POSTPROCESSING DECISION TREES: THE LIMITED RESOURCE CASE

3.1 A Formal Definition of BSP

Previous case considered each leaf node of the decision tree to be a separate customer group. For each such customer group, we were free to design actions to act on it in order to increase the net profit. However, in practice, a company may be limited in its resources. The limited-resource problem can be formulated as a precise computational problem. Consider a decision tree DT with a number of source leaf nodes that correspond to customer segments to be converted and a number of candidate destination leaf nodes, which correspond to the segments we wish customers to fall in. Formally, the bounded segmentation problem (BSP) is defined as follows:

Given:

1. a decision tree DT built from the training examples, with a collection S of m source leaf nodes and a collection D of n destination leaf nodes

2. a prespecified constant k (k <= m), where m is the

total number of source leaf nodes,

3. a cost matrix C(attri,u,v),i=1,2,….. which specifies the cost of converting an attribute attri's value from u to v, where u and v are indices for attri's values,

4. a unit benefit vector Bc(Li) denoting the benefit obtained from a single customer x when the x belongs to the positive class in a leaf node
Li, i=1,2,……..N ,Where N is the number of leaf nodes in the tree DT, and

5. a set Ctest of test cases.

A solution is a set of k goals {Gi , i=1, 2, . . .k}, where each goal consists of a set of source leaf nodes Sij and one designation leaf node Di, denoted as : ({Sij,j=1,2,….,|Gi|}→Di) where Sij and Di are leaf nodes from the decision tree DT. The meaning of a goal is to transform customers that belong to the source nodes S to the destination node D via a number of attribute-value changing actions [6].

Here the main aim is to find a solution with the maximal net profit (defined below).

Goals. Given a source leaf node S and a destination leaf node D, we denote the objective of converting a customer x from S to D as a goal, and denote it as S → D. The concept of a goal can be extended to a set of source nodes: To transform a set of leaf nodes Si to a designation leaf node D, the goal is {Si,i=1,2,….}→D.

Actions.

n order to change the classification of a customer x from S to D, one may need to apply more than one attribute-value changing action. An action A is defined as a change to an attribute value for an attribute attr. Suppose that for a customer x, the attribute attr has an original value u. To change its value to v, an action is needed. This action A is denoted as A={attr ,u→v}.

Action Sets.

A goal is achieved by a set of actions. To achieve a goal of changing a customer x from a leaf node S to a destination node D, a set of actions that contains more than one action may be needed. Specifically, consider the path between the root node and D in the tree DT. Let {(attri=vi),i=1,2,….ND} be set of attribute-values along this path. For x, let the corresponding attribute-values be {(attri = ui) ,i= 1,2,….ND}. Then, the actions of the form can be generated: Aset={(attri=ui→vi),i=1,2,….ND} where we remove all null actions where ui is identical to vi (thus, no change in value is needed for an attri). This action set ASet can be used for achieving the goal S → D.

Net Profits.

The net profit of converting one customer x from a leaf node S to a destination node D is defined as follows: Consider a set of actions Aset for achieving the goal S →D. For each action attri, u → v in Aset, there is a cost as defined in the cost matrix: C(attri, u, v). Let the sum of the cost for all of Aset be Ctotal, S → D(x). Let the probability of x to belong to the positive class in S be p(+|x,S). Likewise, let the probability of a customer in D be in the positive class be p(+|x,D). Recall that from the input, we have a benefit vector Bc (L)for the leaf nodes L. Thus, we have Bc(S) as the benefit of belonging to node S and Bc (D) as the benefit of belonging to node D. Then, the unit net profit of converting one customer x from S to D is:

Punit (x,S → D) = (Bc(D)* p(+|x,D) - Bc(S)* p(+|x,S)) – Ctotal,S→D(x)

Then, for a collection Ctest of all test cases that fall in node S, the total net profit of applying an action set for achieving the goal S →D is:

P(Ctest,S→D)=Sigma xeCtest(Punit (x,S → D)

When the index of S is i, and the index of D is j, we denote Punit (x,S → D) as Pij for simplicity.

Thus, the BSP problem is to find the best k groups of source leaf nodes (Groupi, i = 1,2, . . . ; k) and their corresponding goals and associated action sets to maximize the total net profit for a given test data set Ctest.

An Example to illustrate, consider an example in Fig. 2.Assume that for leaf nodes L1 to L4, the probability values of being in the desired class are 0.9, 0.2, 0.8, and 0.5, respectively. Now consider the task of transforming L2 and L4 to a higher probability node, such that the net profit of making all To illustrate the process, consider a test data set such that there is exactly one member that falls in each leaf node of this decision tree. In order to calculate the net profit, we assume all leaf nodes to have an initial benefit of one unit. For simplicity, we also assume that the cost of transferring a customer is equal to the number of attribute value changes multiplied by 0.1. Thus, to change from L2 to L1, we need to modify the value of the attribute Status; with a profit gain of transformations is maximized. (0.9*1-0.2*1) - 0.1= 0.6.

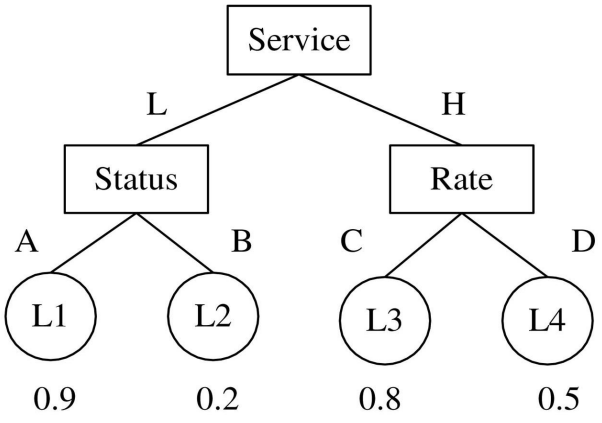


Fig. 2 An example of decision tree

The BSP problem has an equivalent matrix representation. From (2) and (3), we obtain a profit matrix M=((Pij),i=1,2,…m,j=1,2,….n)formed by listing all source leaf nodes Si as the row index and all the action sets ASetj, for achieving the goal (Si → Dj), as the column index (here, we omit Si in the column headings). In this matrix M, (Pij >= 0), 1 < i < m (where m is the number of source leaf nodes), and 1< j < n (n is the number of destination leaf nodes). Pij denotes the profit gain computed by applying ASetj to Si for all test-case customers that falls in Si. If Pij > 0, that is, applying ASetj to transfer Si to the corresponding destination leaf node can bring about a net profit, we say that the source leaf node Si can be covered by the action set ASetj. From (2) and (3), the computation of the profit matrix M(..,..) can be done in O(m *n).

As an example of the profit matrix computation, a part of the profit matrix corresponding to the source leaf node.

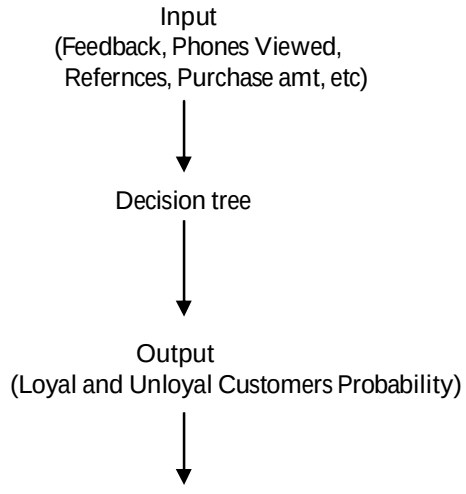
TABLE 1
An Example of the Profit Matrix Computation

	$ASet_1(L_2)$(Goal=→ L_1)	$ASet_2(L_2)$ (Goal=→ L_3)	$ASet_3(L_2)$ (Goal=→ L_4)
L_2	0.6	0.4	0.1
L_4	...	...	...

4 CASE STUDY:

An Application is developed in Java, SQL Server for incrementing the profit of "Mobile handset selling"shop / Industry by applying different discounts to the customers. Before giving discount to the different types of customers, we can check how much profit can be gained by the shop / Industry as well as by the customer. Also we can find the profit gained by a particular customer for a particular scheme and accordingly we can offer most profitable scheme to the customer to gain high profit.

This system relies on not only a prediction, but also a probability estimation of the classification, such as the probability of being loyal. Such information is available from decision trees [7].



Input

Post processing actions is the action performed on the customer, in an attempt to make him loyal……..!:

Post processing Decision Tree:

Actions:

In order to change the classification of a customer x from Loyal and UnLoyal, one may need to apply more than one attribute-value changing action. An action A is defined as a change to an attribute value for an attribute Attr.

Suppose that for a customer x, the attribute Attr has an original value u. To change its value to v, an action is needed.

U is probability that we got…

V is what we expecting..!!

Therefore Action a is to be taken on customer so that he is loyal, and Profit is also not affected..!!

This action A is denoted as A = (Attr, u--v)

5 CONCLUSIONS AND FUTURE WORK

Most data mining algorithms and tools produce only the segments and ranked lists of customers or products in their outputs. In this paper, we present a novel technique to take these results as input and produce a set of actions that can be applied to transform customers from undesirable classes to desirable ones. For decision trees considered, aim is to maximize the expected net profit of all the customers. We have found a greedy heuristic algorithm to solve both problems efficiently and presented an ensemble-based decision-tree algorithm that use a collection of decision trees, rather than a single tree, to generate the actions. Also it is shown that the resultant action set is indeed more robust with respect to training data changes.

In my future work, we will research other forms of limited resources problem as a result of post processing data mining models and evaluate the effectiveness of our algorithms in the real-world deployment of the action oriented data mining.

6 REFERENCES:

[1] C. X. Ling and C. Li. Data mining for direct marketing – specific problems and solutions. In Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), pages 73 – 79. 1998.

[2] J. Dyche, The CRM Handbook: A Business Guide to Customer Relationship Management. Addison-Wesley, 2001.

[3] C.X. Ling, T. Chen, Q. Yang, and J. Cheng, "Mining Optimal Actions for Intelligent CRM," Proc. IEEE Int'l Conf. Data Mining (ICDM), 2002.

[4] X. Zhang and C.E. Brodley, "Boosting Lazy Decision Trees," Proc. Int'l Conf. Machine Learning (ICML), pp. 178-185, 2003.

[5] B. Liu, W. Hsu, L.-F. Mun and H.-Y. Lee. Finding interesting patterns using user expectations. Knowledge and Data Engineering, 11(6):817–832, 1999.

[6] P. S. Usama Fayyad, Gregory Piatetsky-Shapiro. From data mining to knowledge discovery in databases. AI Magazine, 17(11), Fall 1996.

[7] Q. Yang, J. Yin, C.X. Ling, and T. Chen, "Post processing Decision Trees to Extract Actionable Knowledge," Proc. IEEE Conf. Data Mining (ICDM '03), pp. 685-688, 2003.