

OFFER: Making the Right Offer at Customer Contact

Arno Siebes and Ad Feelders

Institute of Information and Computing Sciences

Utrecht University

{arno, ad}@cs.uu.nl

Abstract

We introduce OFFER, an algorithm that uses lists of association rules to make an appropriate product offer at customer contact. We describe how the algorithm uses an initial collection of association rules to generate a list of rules, with updated confidence and support, for each product. We also show how the output of OFFER is used to make the right offer at customer contact. In the experimental evaluation OFFER is applied to a real life data set containing information about the product ownership of the customers of an insurance company.

1 Introduction

A customer contact, e.g., by phone or the web, is often a good moment to offer a new product to this customer. The problem is: “what product should be offered”? It is a problem that many companies face. In this paper, we assume that the decision “what to offer” is to be based on the products the client already owns. In other words, the problem is:

- Given a set of customers, each with the set of products they own.
- Determine for an arbitrary customer (i.e. an arbitrary set of products) the product that can be offered to the client best.

To turn this problem into a data mining problem, there are two issues that we have to resolve. Firstly, what type of model should we infer. Secondly, how do we quantify “best”.

For the type of model, observe that this is not a simple classification problem. Firstly, because there is no well-determined class attribute. Secondly, because a client should, of course, never be offered a product that he or she already owns. Hence, a single classification tree will never do

the trick. Rather, we need a model that considers the “quality” of each product the client doesn’t own and determine the best offer among these products

Since we base our decision on products a client already owns, association rules are a natural option. We could sort the association rules with a singleton righthand-side on their consequent and base our decision on this sorted collection of rules. However, association rules are not mined for their predictive power as argued by Freitas in (Freitas, 2000). Hence, this simple solution could lead to unwelcome surprises.

Liu et al. (Liu et al., 1998) have shown how to create classification systems from a set of association rules. We adopt their method, but with a twist. We do some recalculations after the addition of a rule, because the fact that a customer doesn’t fit a particular rule gives new information. See section 2.1 for more details.

The definition of what constitutes the best offer is very much a business problem. It could be defined as the offer with the highest expected direct revenue, the offer with the highest expected lifetime revenue or something completely different. In this paper we have chosen *confidence* as the function to optimize. Many other functions can be used, with trivial modifications of our algorithm. We discuss some of these options in section 5.

2 OFFER

In this section we discuss the OFFER algorithm. In section 2.1 we describe how the rules for a product are ordered, and how their confidence and support are updated during this process. In section 2.2 we explain how to construct a collection of such list from an initial set of association rules.

2.1 Order

Let P be a product and L the list of association rules with P at its righthand-side. Furthermore, let $conf(R)$ denote the confidence of rule R , and let $sup(R)$ denote the support of rule R .

The order of the rules in L is determined as follows:

1. if $conf(R_1) > conf(R_2)$ then $R_1 \prec R_2$;
2. if $conf(R_1) = conf(R_2)$ and $sup(R_1) > sup(R_2)$ then $R_1 \prec R_2$;
3. otherwise the order is arbitrary.

Hence, $R_1 \prec R_2$ should be read: “ R_1 is a better rule than R_2 ”. This order is similar to the one used in (Liu et al., 1998).

If a client C fits¹ the first rule, $R_1 \in L$, $conf(R_1)$ is a reasonable estimate of the probability that C is interested in P . If C does not fit R_1 , but does fit the second rule $R_2 \in L$, $conf(R_2)$ is *not necessarily* a good estimate of that probability.

To illustrate this claim, let $\sigma(R)$ denote the subset of tuples that fit rule R . For the sake of the argument, assume that $\sigma(R_1) \cap \sigma(R_2) \neq \emptyset$ and that $conf(R_1) > conf(R_2)$. We define

1. c_1 : the confidence for R_2 computed on $\sigma(R_1) \cap \sigma(R_2)$ and
2. c_2 : the confidence for R_2 computed on $\sigma(R_2) \setminus \sigma(R_1)$.

We can write $conf(R_2)$ as the weighted average

$$conf(R_2) = w \times c_1 + (1 - w) \times c_2$$

Now we can see that $conf(R_2)$ is equal to c_2 only if $c_1 = c_2$. In other words, $conf(R_2)$ is in general a biased estimate for the probability that C (who doesn’t fit R_1) is interested in P .

Therefore, we cannot use L to compute the probability that C is interested in product P . Rather, we have to recompute the support and confidence of the remaining rules each time a rule is added to the list:

ORDER (RS : Ruleset for product P)

$db := database$

$Res := empty\ list$

While $db \neq emptyset \wedge RS \neq \emptyset$ **do**

$R := a\ minimal\ element\ wrt\ \prec\ of\ RS$

$Res := Res \oplus \langle R, conf(R) \rangle$

$RS := RS \setminus \{R\}$

$db := db \setminus \sigma(R)$

For $r \in RS$ **do**

recompute support and confidence

if either is too small remove r from RS

Return Res

In which \oplus adds its righthand operand to its lefthand operand as the last element.

Clearly, ORDER is an expensive procedure. However, if we associate $\sigma(R)$ with each $R \in RS$, we can compute all steps necessary for the addition of R to the result-list in one pass over the data-structure; thus reducing the costs.

2.2 OFFER

Using ORDER, the procedure to compute the lists for all products is straightforward. First we compute the association rules with a singleton righthand-side. Then we ORDER them for each consequent:

OFFER ($minsup, minconf$)

$Res := \emptyset$

$RS := \{the\ singleton\ righthand-side$

association rules for $minsup$ and $minconf$ \}

For $P \in Prod$ **do**

$RS[P] := \{rules\ in\ RS\ with\ P\ as\ consequent\}$

$Res := Res \cup \{\langle P, ORDER(RS[P]) \rangle\}$

Return Res

3 Using OFFER

After application of OFFER we obtain for each product P a list of association rules with P as the single righthand-side element. In general, each list is ordered from high confidence to low confidence, although on occasion the confidence may actually go up, due to the recomputation of confidence each time a rule is added to the list.

The next problem is to use these rulelists to make a good offer to a client C . First of all we only offer products that are not already owned by C . This is a realistic assumption in many cases, for example we would not expect C to be

¹That is, C has all products mentioned in the lefthand-side of R_1 .

interested in a fire policy if he or she already has one.

Therefore we only consider the rulelists for products that are not owned by C in making the offer. For each such rulelist we go down the list and find the first rule that fits C and then record the confidence of this rule. After doing this, we have for each product not owned by C an estimate of the probability that C is interested in buying that product. Finally, we select the product with the highest confidence and offer it to C .

4 Experiments

For the experiments we used the insurance company benchmark of the COIL 2000 challenge, available from the UCI KDD archive (Hettich and Bay, 1999). This data set contains information about the customers of an insurance company, and consists of 86 variables concerning product ownership and socio-demographic characteristics derived from postal area codes. The data has been divided in a training set of 5822 observations (customers) and a test set of 4000 observations.

For the association rule discovery we ignored the socio-demographic variables (1-43) in the data set. The remaining variables consist of the target variable of the original challenge (whether or not someone owns a caravan policy) and variables concerning the ownership (number of policies) and contribution (money amount) for 21 other insurance products. In order to make the data suited for analysis with binary association rules, they were discretized in the obvious way, i.e. new value = I(old value > 0). After this discretization the variables concerning amounts contain exactly the same information as those concerning numbers of policies so we discarded the variables concerning the contribution amount.

This leaves us with a training and test set consisting of 22 binary variables about a total of 5822 and 4000 customers respectively. Finally, we removed the observations with all variables equal to zero, resulting in a training set of size 5213 and a test set of size 3588.

Table 1 provides a description of the variables and the corresponding relative frequencies of product ownership in both the training and test data.

We analysed the training data with the Apriori algorithm (Agrawal and Srikant, 1994) using a minimum support of 0.01 (53 records). All rules with 1 item (product) on the righthand-side were generated from the large itemsets found. This yielded a total of 108 rules. Table 2 shows the numbers of rules for all products. Table 3 shows the numbers of rules with a confidence higher than 0.75, for all products.

As might be expected from table 1, rules for products 1,4 and 16 are dominant because these products are owned by many more people than the other products (leading to higher support and confidence).

Evaluation of our product offering method is performed using the test set. Each test customer is dropped down all rulelists (i.e. the rulelists for all products) and for each product we record the first matching rule with the corresponding confidence. Finally we offer the product with the highest confidence. Note that we do not check whether the customer already owns the product before we offer it. For the purpose of evaluation we temporarily assume the customer doesn't own the product. To determine whether our offer would be accepted, we check if the item offered is actually owned by the customer. Clearly, this is a simplification of the problem since we ignore all elements of timing. It is however a reasonable way of evaluating our method with the limited information we have available.

Product	1	4	16
Number	8	16	16

Table 3: Numbers of rules with confidence > 0.75 per product

The default strategy, not using association rules, would be to offer all test customers the product with the highest ownership in the training data. From table 1 we can read that this is product 16 with an ownership of 0.605. We can also read from the same line in the table that this would lead to a succes rate of 0.614 on the test customers.

To make a useful comparison to the default strategy, we use a minimum confidence of 0.605 in OFFER. This means we only make an offer if the best match for a customer yields a con-

Variable	Description	Mean (training)	Mean (test)
1	private third party insurance	0.449	0.440
2	third party insurance (firms)	0.016	0.014
3	third party insurance (agriculture)	0.023	0.025
4	car policy	0.571	0.563
5	delivery van policy	0.009	0.012
6	motorcycle/scooter policy	0.043	0.039
7	lorry policy	0.002	0.001
8	trailer policy	0.012	0.011
9	tractor policy	0.027	0.029
10	agricultural machines policy	0.004	0.003
11	moped policy	0.076	0.077
12	life insurance policy	0.056	0.062
13	private accident insurance policy	0.006	0.004
14	family accidents insurance policy	0.007	0.011
15	disability insurance policy	0.004	0.004
16	fire policy	0.605	0.614
17	surfboard policy	0.001	0.002
18	boat policy	0.006	0.003
19	bicycle policy	0.028	0.028
20	property insurance policy	0.009	0.010
21	social security insurance policy	0.016	0.015
22	mobile home insurance policy	0.067	0.066

Table 1: Description of insurance data set

Product	1	2	3	4	6	9	11	12	16	19	21	22
Number	16	1	7	24	7	7	2	7	25	3	2	7

Table 2: Numbers of rules per product (products with no rules left out)

confidence of at least 0.605. In that case we make an offer to 56% of the test customers. Of those customers 76% accepts the offer (as opposed to 61% for the default strategy). The mean confidence of the offers made was 87% which is considerably higher than the actual success rate of 76%. Of the 2001 offers we make, 1674 are for product 16, 215 for product 1, and 112 are for product 4. This is again no surprise, considering the high ownership of these policies.

As a more sophisticated alternative to the default strategy (offer 16 to all test customers) we considered the use of lists of association rules per item, but without the updating of confidence and support as is done in OFFER. Again, taking a minimum confidence of 0.605 we get the following results. We make an offer to 72% of the customers, but only 64% accepts the offer. This is only a slight improvement over the

default rule. The mean confidence of the offers made was 85% which is dramatically higher than the actual success rate of 64%. Of the 2566 offers we make, 1782 are for product 16, 624 for product 1, and 160 are for product 4. The overconfidence of this method is partly due to the wrong confidence estimates that are used in the rulelists (only the estimates of the first rule in each list are guaranteed to be correct). These wrong estimates also lead to picking the wrong rule as witnessed by the low success rate of 64%.

The difference between OFFER and the rulelists without updating seems to diminish at higher levels of minimal confidence. Using OFFER with a minimal confidence of 0.8 we make an offer to 48% of the customers, of which 87% accepts the offer. The mean confidence of the offers made was 90% which is again higher than the actual success rate of 87%.

minconf	Default	OFFER			Rulelist		
		0.5	0.605	0.8	0.5	0.605	0.8
offer (%)	100	69	56	48	92	72	50
success (%)	61	62	76	87	50	64	88
offer \times success (%)	61	43	43	42	46	46	44
mean confidence (%)	61	80	87	90	79	85	92

Table 4: Results of the default strategy, OFFER and rulelists without updating for different levels of minimal confidence

Without updating confidence and support, we make an offer to 50% of the customers with a success rate of 88%. The estimated success rate was 92%.

Finally, using OFFER with a low minimal confidence of 0.5 we make an offer to 69% of the customers, of which 62% accepts the offer. This is only a slight improvement of the default success rate of 61%. The mean confidence of the offers made was 80% which is again higher than the actual success rate of 62%.

Without updating we make an offer to 92% of the customers with a success rate of 50%. This is actually worse than the default success rate. The estimated success rate was 79%.

The results of the experiments are summarized in table 4. The first row in the table shows the percentage of customers that was offered a product for the different strategies and minimum confidence levels. The second row of the table shows what percentage of the customers that received an offer accepted it. The third row shows what percentage of all customers was made an offer and accepted it. Finally, the fourth row of table 4 shows the mean confidence of the offers that were made.

From these results we draw the following tentative conclusions. Both OFFER and rulelists without updating tend to overestimate the success rate of the offers made. Part of this effect can be explained by the fact that we always *select* the product that yields the highest confidence. Neither method takes this selection into account in the confidence estimation of the rules. For the rulelists without updating we have an additional bias as was explained in section 2.1. In our experiments this bias is shown to be upward, since the overestimation of the success rate is more severe for rules without updating than for OFFER.

We also observe that the overestimation becomes smaller as the minimal confidence of the offers made becomes higher. This can be explained by the fact that with higher minimal confidence, we tend to use rules from the top of the lists. For the method without updating this means the confidence estimates are less biased. We observe however a decrease for OFFER as well, because the upward bias due to selection of the product with highest confidence is also decreased as we increase minimal confidence.

Looking at the number of accepted offers as a percentage of all customers (offers \times success in table 4), we observed that this is fairly stable as minimal confidence goes up. Taking the viewpoint that we don't want to annoy customers with offers they are not interested in, we conclude that we should set minimal confidence fairly high for this application. Note also that total acceptance is somewhat higher for the rulelists without updating, but only at the expense of making more offers.

5 Conclusions and Future Research

Deciding which product to offer to a customer during a customer contact is an important problem that many companies face. In this paper we have shown how association rules can be exploited for this purpose. In particular, we have introduced the OFFER algorithm that produces for each product a list of ranked association rules. We have shown that by updating the confidence and support during the ranking, we can reduce the bias of the confidence estimates that are obtained without updating. The experimental results show that OFFER allows us to target a smaller group of customers with a good offer, resulting in a higher success rate.

There are different ways in which this work could be extended and refined.

We could for example look at different per-

formance measures than just the confidence. One interesting possibility is expected life-time value. Suppose that customers that buy X are less likely to buy A, B, and C. In that case it might be better to offer A, although the short term profit is in X.

As already mentioned we could also take into account the order in which products are purchased. Clients typically follow “routes” in which one route can be far more profitable than the other.

Finally, we could consider to use a different algorithm than Apriori to generate the initial collection of rules. In (Castelo et al., 2001) we describe an algorithm for generating association rules based on conditional independencies observed in the data. Further experimentation with this method will have to show whether this algorithm will generate less rules and better predictions in the context of cross-selling.

References

- R. Agrawal and R. Srikant. 1994 Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases*.
- S. Hettich and S.D. Bay. 1999 *The UCI KDD Archive* [<http://kdd.ics.uci.edu>]. Irvine, CA: University of California, Department of Information and Computer Science.
- Bing Liu, Wynne Hsu and Yiming Ma. 1998 Integrating Classification and Association Rule Mining. In: R.Agrawal, P. Stolorz and G. Piatetsky-Shapiro (eds) *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, AAAI Press, Menlo Park (CA), pp. 80–86.
- R. Castelo, A. Feelders and A. Siebes. 2001 MAMBO: Discovering Association Rules Based on Conditional Independencies. In: F. Hoffmann et al. (eds) *Proceedings of the fourth international symposium on intelligent data analysis (IDA-2001)*, LNCS 2189, Springer, pp.289-298.
- A. Freitas. 2000 Understanding the Crucial Differences between Classification and Discovery of Association Rules – A Position Paper. *SIGKDD Explorations*, vol 2, no 1, pp. 65–69.