Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 93 (2016) 19 – 25
Procedia Computer Science

6th International Conference On Advances In Computing & Communications, ICACC 2016, 6-8 September 2016, Cochin, India

Predicting financial savings decisions using sigmoid function and information gain ratio

Mahalingam P.R*, Vivek S

Muthoot Institute of Technology and Science, Varikoli P.O, Puthencruz, Ernakulam – 682308, India

Abstract

Planning for savings remains one of the most critical decisions for any user. The most important factor in this process is decision making. Most of the time, we can take decisions on how much money to save at a time based on our spending pattern. But automating this process is not easy, since it involves a number of parameters. Here, we attempt to incorporate intelligence into this decision making. The algorithm will attempt to predict the maximum amount to save based on the current account balance, clubbed with the entire database of available transactions on that account / user. Every transaction will be assigned an impact factor based on the time of occurrence, relative to the current date. Every month is divided into four quarters to track recurring expenses like EMIs. These impacts have to be taken by a machine learning algorithm to predict the maximum possible savings in that quarter. The impact factor will also depend upon the fraction of balance being spent on that quarter. If there is a goal set for savings, it will also be taken into consideration. If a considerable expense is predicted for that month, the savings will be kept low so that the account won't go into overdraft. Recurring expenses are kept in check and accounted for to the maximum extent using information gain ratio from the transaction list.

Keywords: machine learning; decision making; automation; finance; information gain

*Corresponding author. Tel.: +91-8281134511 .
E-mail address: prmahalingam@gmail.com

doi:10.1016/j.procs.2016.07.176

1. Introduction

Financial savings is a critical decision point for any person irrespective of the age group. Coming under a common category of "financial technologies", these applications deal with different aspects of finance, like online wallets, savings monitor, etc. The algorithm being introduced here is part of such a project which aims at introducing people to automated savings management.

The paper by C.P.S Nayar[1] explores how traditional and modern financial systems can co-exist. In the current scenario, the direction is towards automating financial processes with ample stress on internet and mobile based ecosystems. The concept of i-banking (internet enabled banking) and m-banking (mobile based banking) is gaining traction with most banking institutions coming out with smartphone applications.

In a survey conducted in Yale School of Management[2], it was found that mobile based banking is gaining steadily in India with a good accepting audience in most strata of financial status. As mentioned in the survey, upper classes find it as an easier alternative, while middle class find it a cost effective and accessible solution. It defines the current model as follows:

- Mobile payment is restricted to bank account holders, residents of India and transactions in Indian rupees
- No interbank network available
- Banks can make multilateral agreements to create mobile switches

Trust and security are the primary points of concern in a growing market like India.

2. Current status of work

We are designing an application that eases savings by providing automated decisions. Decision making involves a fair amount of "intelligence". Human nature is automatically wired to make decisions. But when we consider a digital application, it cannot decide on its own. Instead we have to help it learn how to decide based on a given scenario.

As mentioned in a survey published by S. Liu et al.[3], an integrated decision support system is supported by the following factors – Data, Models, Process and Service. The algorithm should be able to take all of these together and arrive at a decision that can give suggestions to the user.

2.1. Data

The data required for this is purely financial in nature. To manage the trust and security concerns, the application should not perform any unintended operation on the user's account. Such a separation can be provided only by avoiding direct access completely. Hence, we have to keep track of the messages and alerts sent by the bank to the user's mobile.

2.2. Models

Data models depend on the data itself. Here, we consider 4 main factors to model the transactions retrieved from the user's mobile - Account number, Transaction date, Transaction amount, and Balance after transaction. From these data we should be able to obtain what fraction of the available balance was spent on each transaction. The date is important since we may have to consider recurring expenses like EMIs before arriving at a decision.

2.3. Process and service

Decisions should be taken based on the spending pattern of the user. If the user has a high probability to spend more in a particular time period, we should adjust the amount to be pulled towards savings to avoid overdrafts.

To support this, we use the fraction we calculate from the model. Thus, even if we ignore credit messages, the debit messages can give a clear picture of the spending patterns. The credit message will be needed only when a transaction message doesn't specify the final balance.

3. Foundations of the proposed system

The proposed application will allow the user to set savings targets to be achieved over a period of time (For example, save Rs.10,000 over the next 4 months). The suggestions should be able to guide the user towards the goal as much as possible.

For ease of operation and convenience for the user, the algorithm should not alert the user every day. Hence, an interval of 8 days has been defined (approximating to a quarter of a month). This results in the application giving suggestions every month on the 1st, 9th, 17th and 25th days. To avoid any future overdrafts, the application restricts every instance of savings to 20% of the available balance. This results in the scenario described in Table 1, considering the user starts with Rs.10,000 at the beginning of the month.

Table 1 : Example savings scenario

Date	*Initial balance*	*Savings @ 20%*	*Final balance*
1	10,000	2,000	8,000
9	8,000	1,600	6,400
17	6,400	1,280	5,120
25	5,120	1,024	4,096

At an ideal scenario, the user will be able to push almost 59% of his earnings into savings considering the given ratio. If you consider the target mentioned earlier, this rate is sufficient to achieve the target in under 4 months. But in a practical situation, this will lead to problems because the user may not be able to spare 59% of his earnings every month. The decision algorithm should suggest a proper ratio between 0% and 20% every quarter to the user.

4. Design of multivariable function

The decision algorithm is a function of message list. It has to extract the parameters from the messages and work on them. For that, we define an "impact score" for each transaction. The impact score is defined in a specific way. Consider the following variables - cm is the current month number (1 – 12) and cq is the current quarter (based on date). Then,

$$\text{today} = (\text{cm} - 1) * 4 + \text{cq} \tag{1}$$

Then, for each transaction in the message list, the following actions are taken. *tm* is the month number for the current transaction (1 – 12) and *tq* is the quarter for the current transaction (based on date). Then,

$$\text{txn} = (\text{tm} - 1) * 4 + \text{tq} \tag{2}$$

c_i defines the impact of the current transaction and a_i defines the amount transacted in the current transaction.

c_i is assigned using a sigmoidal function which generates the curve in Fig 1(a).We quantize the curve into 24 intervals. The value 24 is critical because that is the maximum difference between *today* and *txn*. The advantage of using sigmoidal function is that initial values keep close to 1, while far off values keep close to 0. Intermediate values are closer to linear mapping with slope=1. This is illustrated in Fig 1(b).

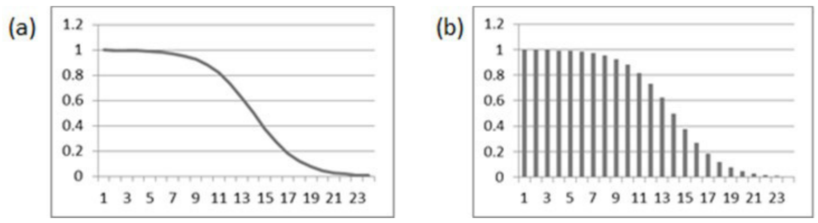

Fig. 1. (a)Sigmoidal variation for impact factor, (b)Quantized sigmoid function.

The quantization level is determined by taking |*today-txn*|. Based on the quantization, levels are assigned for c_i as in Table 2.

Table. 2. Quantized values for algorithm.

|today-txn|	c_i	**|today-txn|**	c_i
0	1	13	0.62246
1	0.9985	14	0.5
2	0.99753	15	0.37754
3	0.99593	16	0.26894
4	0.99331	17	0.18243
5	0.98901	18	0.1192
6	0.98201	19	0.075858
7	0.97069	20	0.047426
8	0.95257	21	0.029312
9	0.92414	22	0.017986
10	0.8808	23	0.010987
11	0.81757	24	0.006693
12	0.73106		

The motivation of using sigmoid came from the fact that it is quite similar to real world thought process and adds the element of fuzziness to a conventional linear process[4]. It provides linear growth in middle, while truncating the curve at extremities.

Even as we understand that this is closer to real world thought, there is one major deficiency in this evaluation. When we want to study financial patterns, we mentioned before that recurrent transactions have a vital role. They denote a chance that such a transaction can occur again, and we have to anticipate the same before fixing a suitable amount. Hence, we add a multiplying factor to the expression. The multiplying factor is defined as

$$m = 0.25 * (4 - (|\text{today} - \text{txn}|\,\%4)) \tag{3}$$

This is multiplied with the current quantized values to give a new impact score, which is described in Table 3.

Table. 3. Impact score for transactions.

|today-txn|	*c_i*	*m*	*New c_i*	*|today-txn|*	*c_i*	*m*	*New c_i*
0	1	1	1	13	0.62246	0.75	0.466845
1	0.9985	0.75	0.748875	14	0.5	0.5	0.25
2	0.99753	0.5	0.498765	15	0.37754	0.25	0.094385
3	0.99593	0.25	0.248983	16	0.26894	1	0.26894
4	0.99331	1	0.99331	17	0.18243	0.75	0.136823
5	0.98901	0.75	0.741758	18	0.1192	0.5	0.0596
6	0.98201	0.5	0.491005	19	0.075858	0.25	0.018965
7	0.97069	0.25	0.242673	20	0.047426	1	0.047426
8	0.95257	1	0.95257	21	0.029312	0.75	0.021984
9	0.92414	0.75	0.693105	22	0.017986	0.5	0.008993
10	0.8808	0.5	0.4404	23	0.010987	0.25	0.002747
11	0.81757	0.25	0.204393	24	0.006693	1	0.006693
12	0.73106	1	0.73106				

This generates the graph as in Fig 2(a). The curve after quantization into 24 intervals will also carry the same properties of the modified sigmoid, as we have seen with the conventional sigmoid. This is given in Fig 2(b).

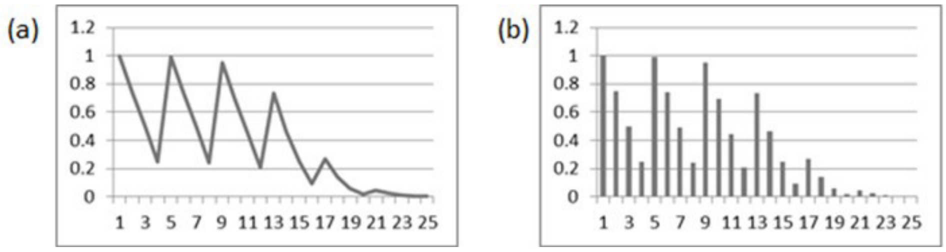

Fig 2. (a)Sigmoid curve infused with recurrent transaction impact, (b) Modified sigmoid quantized with 24 intervals.

The same is repeated for all transactions in the list and we consolidate the same considering the amount transacted.

$$\text{impactamount} = \sum (ci * ai * fi) \tag{4}$$

Amount is consolidated with the impact to find the effect of each transaction on the current quarter. Now, we analyze what fraction of the current balance is going to get affected.

$$\text{ratio} = (\text{impactamount} / \text{currentbalance}) * 100 \tag{5}$$

When you take *100-ratio*, you will get the fraction of amount left after the impacted amount. This is quantized into intervals of 5. This is passed into another sigmoid to give the final recommendation, with a linear scaling to 20. This is given in Table 4.

5. Implementation

After considering the foundations, the algorithm can be formalized. Two arrays are defined based on the precomputed values for the system.

currentImpact[25]={1,0.748875,0.498765,0.248983,0.99331,0.741758,0.491005,0.242673,0.95257,0.693105,0.4404,0.204393,0.73106,0.466845,0.25,0.094385,0.26894,0.136823,0.0596,0.018965,0.047426,0.021984,0.008993,0.002747,0.006693}, which is computed as an output of equation (3).

recommend[20]={0.58624,0.94852,1.51716,2.384,3.6486,5.3788,7.5508,10,12.4492,14.6212,16.3514,17.616,18.4828,19.0514,19.4138,19.6402,19.7802,19.8662,19.9186,19.9506}, which is computed after applying equation (5).

The main advantage of this algorithm is that it is very low in time complexity. This achieved since the algorithm works on a number of precomputed values which remain static throughout. This allows us to define them in arrays as mentioned here. The complexity solely depends on the number of transactions being considered, which is an acceptable overhead when considering the accuracy required. This overhead can be eliminated if we choose a subset of transactions (like last 100 transactions only). But at that point we are reducing the accuracy to a huge extent.

Table. 4 Recommendation values for savings

100-ratio	*Sigmoid*	*Recommendation (%)*	*100-ratio*	*Sigmoid*	*Recommendation (%)*
0-5	0.029312	0.58624	50-55	0.81757	16.3514
5-10	0.047426	0.94852	55-60	0.8808	17.616
10-15	0.075858	1.51716	60-65	0.92414	18.4828
15-20	0.1192	2.384	65-70	0.95257	19.0514
20-25	0.18243	3.6486	70-75	0.97069	19.4138
25-30	0.26894	5.3788	75-80	0.98201	19.6402
30-35	0.37754	7.5508	80-85	0.98901	19.7802
35-40	0.5	10	85-90	0.99331	19.8662
40-45	0.62246	12.4492	90-95	0.99593	19.9186
45-50	0.73106	14.6212	95-100	0.99753	19.9506

6. Experimental Results

Consider the message list in Table 5 for a single account system. Only debits have been considered.

Table 5. Message list

Date	*Transacted amount*	*Balance after transaction*	*Date*	*Transacted amount*	*Balance after transaction*
3-Feb	1000	15000	6-Sep	3255	16869
12-Feb	650	18350	23-Sep	1200	15669
21-Apr	484	17866	9-Oct	922	14747
15-May	395	22471	20-Oct	414	14333
12-Jun	5004	17467	18-Nov	766	13567
6-Aug	943	16524	3-Dec	384	13183
20-Aug	10000	16524	19-Dec	18030	20153
27-Aug	2400	14124	27-Dec	3000	17153

Starting from June 15, savings target set at 12000 in 5 months. Initial balance is Rs.20000. Assume no further debits in that period. The output predictions are summarized in Table 6.

Table 6. Savings predictions

Date	*Saved amount*	*Account balance*	*Pending for target*	*Date*	*Saved amount*	*Account balance*	*Pending for target*
15-Jun	3882	16118	8118	1-Aug	497	8760	760
25-Jun	2635	13483	5483	9-Aug	51	8709	709
1-Jul	2204	11279	3279	17-Aug	207	8502	502
9-Jul	606	10673	2673	25-Aug	0	8502	502
17-Jul	1328	9345	1345	1-Sep	172	8330	330
25-Jul	88	9257	1257	9-Sep	330	8000	0

If there are any other transactions in between, that will also be accommodated. The subsequent predictions will take that into consideration also. This execution doesn't add the savings itself into the message list. But in practical cases, the bank message for that will be taken automatically.

7. Future work – Role of information gain

The multivariable function is enough to predict savings based on current transactions. But it suffers a drawback that all transactions are rated in the same manner. That is not always valid in the practical context since some transactions have to be given additional weightage over others. As mentioned by Kent[5], information gain can be considered as a parameter to weigh the impact of a particular item in a set.

To bring this to effect, we have to consider the type of transaction along with conventional transaction parameters. The augmentation will follow the steps taken as part of ID3 Decision Tree[6]. It considers logarithmic probabilities to weigh the impact of transaction type in the global set of transactions. If it goes beyond a certain threshold, we may have to multiply the transaction probability to the transaction weight so that a highly probable transaction is accounted for, and less probable transactions are given low impact, even if they occurred in a nearby frame of time. Initial implementations have shown favourable results, which are pending validation on a large user transaction list.

8. Conclusion

Savings management is a practical problem faced by most people. It is a problem involving a number of variables. Here, we have attempted to create an algorithm that takes inspiration from neural networks to predict saving patterns for users automatically, based on their spending. Sigmoid functions play their part since the curve is naturally tuned to be a "mid-range filter" that trims off at high and low values. We can see that it performs the task quite well, with little risk of overdrafts. By including information gain into the function, we can add more accuracy into the system.

References

1. Nayar, C. P. S. "Can a traditional financial technology co-exist with modern financial technologies? The Indian experience" in Savings and Development, 1986, pp 31-58.
2. K. Sudhir, Mukesh Pandey, Ishani Tewari, "Mobile Banking in India: Barriers and Adoption Triggers" in China India Insights Program, Yale School of Management Center for Customer Rights, 2012, pp 1-30.
3. Shaofeng Liu, Alex H. B. Duffy, Robert Ian Whitfield, Iain M. Boyle, "Integration of decision support systems to improve decision support performance" in Knowledge Information Systems, Springer, 2009, pp 1-26.
4. Malakooti, Behnam, Ying Q Zhou, "Feedforward artificial neural networks for solving discrete multiple criteria decision making problems" in Management Science 40.11, 1994, pp 1542-1561.
5. Kent, John T. "Information gain and a general measure of correlation." Biometrika 70.1 (1983): 163-173.
6. Faculty of Information Technology, "CSE5230 Tutorial: The ID3 Decision Tree Algorithm", Monash University