

Beyond Classification and Ranking: Constrained Optimization of the ROI

Lian Yan
NCO Group, Inc.
507 Prudential Road
Horsham, PA 19044
lian.yan@ncogroup.com

Patrick Baldasare
NCO Group, Inc.
507 Prudential Road
Horsham, PA 19044
patrick.baldasare@ncogroup.com

ABSTRACT

Classification has been commonly used in many data mining projects in the financial service industry. For instance, to predict collectability of accounts receivable, a binary class label is created based on whether a payment is received within a certain period. However, optimization of the classifier does not necessarily lead to maximization of return on investment (ROI), since maximization of the true positive rate is often different from maximization of the collectable amount which determines the ROI under a fixed budget constraint. The typical cost sensitive learning does not solve this problem either since it involves an unknown opportunity cost due to the budget constraint. Learning the ranks of collectable amount would ultimately solve the problem, but it tries to tackle an unnecessarily difficult problem and often results in poorer results for our specific target. We propose a new algorithm that uses gradient descent to directly optimize the related monetary measure under the budget constraint and thus maximizes the ROI. By comparison with several classification, regression, and ranking algorithms, we demonstrate the new algorithm's substantial improvement of the financial impact on our clients in the financial service industry. The proposed algorithm can also be applied to several other areas such as maximizing average returns of stock selection and identifying tax auditing targets of highest values.

Categories and Subject Descriptors

H.4 [Database Management]: Database Applications - Data Mining; I.2.6 [Artificial Intelligence]: Learning; I.5.2 [Pattern Recognition]: Design Methodology - classifier design and evaluation

General Terms

Algorithms

Keywords

return on investment, neural networks, constrained opti-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

UBDM'06, August 20, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-440-5/06/0008 ...\$5.00.

mization

1. INTRODUCTION

Classification has been commonly used in many data mining projects in the financial service industry. We have used a classifier to predict defection of mutual fund accounts for a major US mutual fund company [9], where the positive samples are defined as those accounts with a net redemption amount (redemption minus purchase) of 35% or more of the account balance within a two-month window. We set up a control group for the project to evaluate the model's accuracy. Table 1 shows the real-world evaluation results for the control group over a four-month window, which consists of two levels of defection risk and three segments based on account values.

	Higher risk	
	Def. rate	Avg. net redem.
Segment 1	4.8%	-\$5,145
Segment 2	10.5%	\$14,494
Segment 3	5.7%	\$2,733
	Lower risk	
	Def. rate	Avg. net redem.
Segment 1	1.9%	\$2,494
Segment 2	1.6%	\$13,864
Segment 3	2.4%	\$2,686

Table 1: Defection rate and average net redemption amount for the control group for a US mutual fund company.

We can see that the model was successful at predicting defecting accounts as evidenced by the higher defection rate in the higher risk groups for all the three segments. However, the average net redemption amounts in the higher risk groups were *not* significantly higher than those in the lower risk groups. Especially, for Segment 1, even though the higher risk group had a much higher defection rate than the lower risk group, the negative net redemption amount in the higher risk group indicates a positive net purchase. This model can be used to reduce defection rate, but it would not be the best model used to prevent the highest redemption amount. For a fixed budget, the return on investment (ROI) of the project is determined by the amount of redemptions prevented (rather than by the reduction of defection rate). There are many significant factors other than the model affecting the retained amount, but simply classifying the accounts as defection or non-defection does not enable

the mutual fund company to reach out to those accounts with the highest redemption amount.

As another example, a classifier can be used to predict collectability of delinquent accounts receivable for credit card issuers using credit, demographic, and account data, where a binary class label is created based on whether a payment for an account is received within a certain period since the account was placed into the collection process. Typically, the budget restricts how many accounts can be placed into a specific collection process. While the true positive rate among those accounts in the collection process is a meaningful measure of classification accuracy, maximization of the true positive rate is often different from maximization of the collectable amount for the specific collection process. It is the collection amount rather than the true positive rate that determines the ROI under the fixed budget.

Note that we are always addressing a budget constraint, which determines, among other things, how many mutual fund accounts the customer service team can reach out every month and how many accounts receivable can be placed into a specific collection process. In our applications we represent the budget constraint by the pull rate r which is the percentage of accounts to pull out for a specific intervention/collection process. Let us denote x as the target monetary measure, e.g., collection amount, which directly determines the ROI. Then the goal is to find a function $y(e)$, where e is the independent variables such as credit, demographic, and account data, so that the accounts in the top $r\%$ by y correspond to those in the top $r\%$ by the target x . Thus the problem of maximizing the ROI can be formally defined as

$$\text{Max} \sum_{y(e_i) \in \text{Top } r\%} x_i, \quad (1)$$

where $i = 0, 1, \dots, n - 1$, and n is the total number of accounts.

1.1 Related work

One might immediately suggest cost-sensitive learning, e.g., [4], and ranking, e.g., [2], [3], would solve the above problem. For cost-sensitive learning, we have the cost matrix in Table 2. Assuming $c_{00} = 0$ and $c_{11} = 0$, a typical sensitive learning algorithm tries to minimize the cost

$$C = \sum_{i \in P} (1 - q_i) c_{01} + \sum_{j \in N} q_j c_{10} \quad (2)$$

over the training set, where P , N are the sets of positive and negative samples, respectively, and q_i , q_j are both posterior probabilities of belonging to the positive class. In our application, the actual positives are those accounts which are in the top $r\%$ of x , and predicted positives are those accounts which have a score y in the top $r\%$. It is straightforward that $c_{01} = x$, since if an actual positive is placed out of the top $r\%$ (a predicted negative), the company will not be able to collect $\$x$ or retain the net redemption of $\$x$. If an actual negative is placed among the top $r\%$, the company will lose the opportunity to reach out to one of the accounts with a *larger* x in the top $r\%$, since the number of accounts to be contacted is pre-determined by the pull rate r . Thus, c_{10} is an opportunity cost that is not a constant and unknown. One might still try to train a classifier with sample weights intuitively based on x . In Sections 3 and 4, we compare our algorithm's results with such a classifier's.

	actual negative	actual positive
predicted negative	c_{00}	c_{01}
predicted positive	c_{10}	c_{11}

Table 2: A cost matrix for cost-sensitive learning.

If we can learn a regression model so that $y(e_i) = x_i$, $i = 0, \dots, n - 1$, or a ranking model so that $y(e_i) > y(e_j)$ for any $(i, j) \in \{(i, j) | x_i > x_j, i, j = 0, \dots, n - 1\}$, $\sum_{y(e_i) \in \text{Top } r\%} x_i$ would be optimized for each r . However, both regression and ranking try to solve an unnecessarily difficult problem, and often lead to poorer results for our specific target at pull rate r . Maximization of $\sum_{y(e_i) \in \text{Top } r\%} x_i$ requires only the correct ranking between the Top $r\%$ and the others. The ranking *within* the Top $r\%$ or the others is not necessary, neither is the estimate of x itself by regression. In Sections 3 and 4, we compare our model with a regression and a ranking model, which uses the algorithm in [2].

We present the new algorithm in the next section, where we also describe the several classification, regression, and ranking algorithms which we compare with in our projects. In Section 3, we use the proposed algorithm to predict collectability of accounts receivable for delinquent consumer loan accounts from several US financial institutions. In Section 4, the new algorithm is applied to predicting defection of mutual fund accounts for a major US mutual fund company. Finally, we discuss several algorithmic and applied extensions of the proposed algorithm in Section 5.

2. CONSTRAINED OPTIMIZATION OF THE ROI

For a model with $0 \leq y \leq 1$, assume that the specified pull rate r can be achieved at a decision threshold β ($0 < \beta < 1$), i.e., the accounts in the pull are those with an output larger than β . In this case, maximization of $\sum_{y_i \in \text{Top } r\%} x_i$ can be solved by the following constrained optimization over y_i , $i = 0, \dots, n - 1$, and β :

$$\text{Max} \sum_{i=0}^{n-1} x_i \cdot I(y_i, \beta), \quad (3)$$

subject to

$$\frac{\sum_{i=0}^{n-1} I(y_i, \beta)}{n} = r, \quad (4)$$

where

$$I(y_i, \beta) = \begin{cases} 1 & : y_i > \beta \\ 0 & : \text{otherwise} \end{cases} \quad (5)$$

When the constraint $\frac{\sum_{i=0}^{n-1} I(y_i, \beta)}{n} = r$ is satisfied, the number of accounts with the model output $y_i > \beta$ will be exactly $r\%$ of n . The difficulty here is that $I(y_i, \beta)$ is non-differentiable, and gradient based optimization cannot be used to optimize Eq. 8.

In [8], [9], we demonstrate that the sigmoid function

$$\sigma(y_i, \beta) = \frac{1}{1 + e^{-\kappa(y_i - \beta)}}, \quad (6)$$

where $\kappa > 0$, does not provide a good differentiable approximation to $I(y_i, \beta)$ when $-1 \leq y_i - \beta \leq 1$. Instead, we have

¹For simplicity, we'll omit the independent variable e and use y_i for $y(e_i)$.

proposed the following differentiable approximation

$$f(y_i, \beta + \gamma) = \begin{cases} (y_i - \beta - \gamma)^p & : y_i > \beta + \gamma \\ 0 & : \text{otherwise} \end{cases}, \quad (7)$$

where $p > 1$ and $0 \leq \gamma < 1$. A small but positive γ is often helpful for a better generalization performance over the test set. Now Eq. 3 becomes

$$\text{Max} \sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta + \gamma). \quad (8)$$

However, $\frac{\sum_{i=0}^{n-1} f(y_i, \beta)}{n}$ is not a good approximation to $\frac{\sum_{i=0}^{n-1} I(y_i, \beta)}{n}$, since $f(y_i, \beta)$ is often not close to 1. As in [9], rather than trying to use a differentiable approximation to r , we approximate a related ratio $\frac{r}{1-r}$ by the following differentiable function:

$$\frac{\sum_{i=0}^{n-1} f(y_i, \beta)}{\sum_{i=0}^{n-1} g(y_i, \beta)}, \quad (9)$$

where

$$g(y_i, \beta) = \begin{cases} (\beta - y_i)^p & : y_i < \beta \\ 0 & : \text{otherwise} \end{cases} \quad (10)$$

with $p > 1$. $g(y_i, \beta)$ is a differentiable approximation to the following step function

$$I_p(y_i, \beta) = \begin{cases} 1 & : y_i < \beta \\ 0 & : \text{otherwise} \end{cases}. \quad (11)$$

Since the optimization often moves most y_i close to β in the end, we will see that Eq. 9 can provide a close approximation to $\frac{r}{1-r}$.

Now we convert the constrained optimization into an unconstrained optimization problem by minimizing the following Lagrangian:

$$L = -\frac{1}{n} \sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta + \gamma) + \frac{1}{\mu} \left(\frac{\sum_{i=0}^{n-1} f(y_i, \beta)}{\sum_{i=0}^{n-1} g(y_i, \beta)} - \frac{r}{1-r} \right)^2. \quad (12)$$

During the training iterations, μ is gradually decreased until convergence of the constraint $\left(\frac{\sum_{i=0}^{n-1} f(y_i, \beta)}{\sum_{i=0}^{n-1} g(y_i, \beta)} - \frac{r}{1-r} \right)^2$ is achieved. In practice, we have found that mapping x_i in Eq. 12 to a value between -1 and 1 by

$$\theta(x_i) = \frac{1 - e^{-x_i}}{1 + e^{-x_i}} \quad (13)$$

typically obtains improved results.

In the Appendix, we derive the derivatives for y_i . These derivatives together with the chain rule can then be applied to any parametric model, for which one can optimize the differentiable objective function with respect to the parameters using gradient based methods.² In our projects, we apply the proposed algorithms to a typical multilayer perceptron (MLP) network with softmax outputs between 0 and 1, and with a single hidden layer and direct connection between the input and output layers. β can also be optimized with the model parameters, but we have found that fixing β at 0.5 achieves almost the same results over our data sets.

²We use the limited memory BFGS method in [6].

2.1 Comparing methods

In Sections 3 and 4, we apply the new algorithm to predicting collectibility of accounts receivable and predicting defection of mutual fund accounts, and compare the results of the proposed algorithm with the following four algorithms’.

- *Classification* An ensemble of MLP classifiers is trained by mean squared error based on the defined class label. Since the class prior is typically low, each individual classifier in the ensemble has a modified prior to compensate for the imbalanced data sets [10].
- *Weighted classification* An MLP classifier is trained by mean squared error based on either the defined class label, e.g., whether the net redemption amount is above or below 35% of the account balance, or the ranks of the training samples. Using the ranks to determine the class label is an intuitive idea: labeling those samples in the top $r\%$ of x as positive and the others as negative. When r is the same as the prior of the defined class, these two approaches are the same. During training, the samples are weighted by x or a function of x . To avoid the dominance of those samples with an extreme value of x , we typically use the sigmoid function of x to smooth out the weights.
- *Ranking* Burges et al. propose a ranking algorithm using gradient descent [2]. We apply this algorithm to train an MLP model which ranks x in our applications. The algorithm tries to minimize the cross entropy function

$$\sum_{(i,j) \in S} -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}), \quad (14)$$

where $S = \{(i, j) | x_i \geq x_j, i, j = 0, \dots, n-1\}$, and \bar{P}_{ij} is the target probability of $x_i > x_j$. P_{ij} is the model’s estimate of \bar{P}_{ij} in the form $P_{ij} = \frac{e^{y_i - y_j}}{1 + e^{y_i - y_j}}$. Then the cost function becomes

$$\sum_{(i,j) \in S} -\bar{P}_{ij}(y_i - y_j) + \log(1 + e^{y_i - y_j}). \quad (15)$$

In our experiments, we choose $\bar{P}_{ij} = 1$ if $x_i > x_j$ and $\bar{P}_{ij} = 0.5$ if $x_i = x_j$.

- *Regression* An MLP regressor is trained by mean squared error against x . We map x to a value between 0 and 1 using the sigmoid function.

3. PREDICTING COLLECTIBILITY OF ACCOUNTS RECEIVABLE

Accounts receivable are unpaid customer invoices, and any other money owed to a company by its customers. From credit card issuers to banks, from local retail stores and service businesses, to the federal, state and local governments, if the business or government unit extends credit, offers payment installment plans, or makes assessments, it has accounts receivable. The collection industry serves an important role in the U.S. economy by recovering billions in revenue from charged-off or delinquent accounts receivable for U.S. companies. By returning this money to U.S. companies, the collection industry saves American families

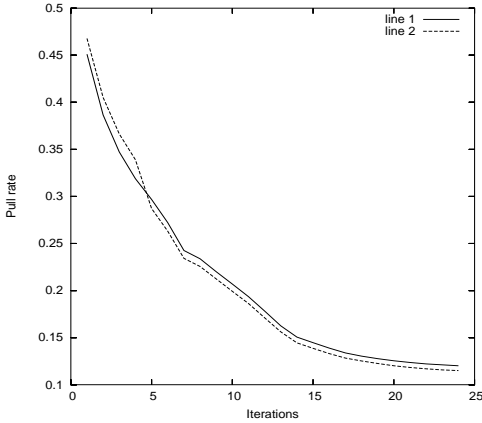


Figure 1: This figure shows convergence of pull rates achieved by the threshold β during the optimization. Line 1 is for the training set, and Line 2 shows the pull rate change over the test set.

on average \$331 a year in money they otherwise would have spent if businesses raised their prices to cover losses to bad debt [1].

The portfolio of accounts receivable we worked on consists of consumer loan accounts from several US financial service institutions. The portfolio includes several types of accounts in terms of account history. For example, some are the so-called Prime accounts which are newly charged off accounts, and some are Seconds which had already gone through a collection process. Our goal is to develop a generic predictive model which can be used to guide the agents' collection efforts. In particular, we would like to identify a high value segment which consists of 11% of the whole portfolio. The 11% is chosen since the payer rate (percentage of paid accounts in the first six months) is 11%. It is clear that the return on investment is determined by the collection amount from the identified 11% accounts in the segment.

The data set includes 684,600 accounts. We randomly split the data set into a training set and a test set of equal size. In addition to the account history and general demographic information, several hundred data fields from a credit score provider about the account owner are also available. The domain experts guided the feature selection, and 30 data fields are used in the final model.³ Missing values for continuous variables are simply imputed by the mean with an added binary column indicating missingness for this variable. Most of the data fields are categorical. For categorical variables with missing values, the sets of distinct values are augmented by another value 'missing'. We encode the categorical variable $C = \{c_1, c_2, \dots, c_k\}$ by replacing c_i with the conditional mean $E(x|c_i)$ and conditional standard deviation $\sigma(x|c_i)$, $i = 1, 2, \dots, k$.

We set $r = 11\%$ and fix β at 0.5 for the new algorithm, trying to maximize the average collection amount among the top 11% accounts. We choose γ and p in Eq. 12 so that the number of training samples with the model output $y > \beta$ is close to $r\%$ and the average collection amount among the

³While we are still working on several feature selection algorithms trying to reveal more useful data features, up to now we have only achieved marginal improvement by adding more data fields.

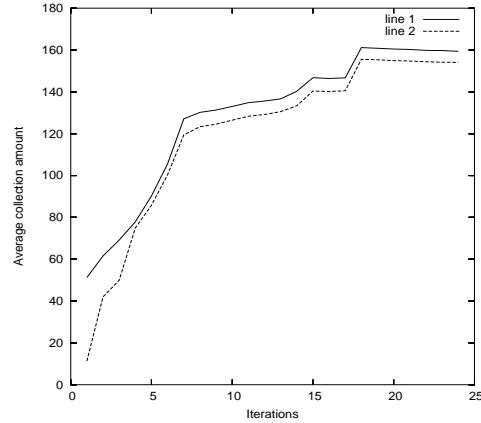


Figure 2: This figure shows the improving average collection amount among the top 11% accounts during the optimization. Line 1 is for the training set, and Line 2 is over the test set.

top $r\%$ in the training set is the largest. Here we chose $\gamma = 0.01$ and $p = 2^4$, and the number of hidden units is 5.⁵ In Figures 1 and 2, we show the optimization process along with iterations of μ , which is initialized at 100, and is updated by $\mu_{t+1} = 0.75\mu_t$ during the optimization, where t is the iteration index. Figure 1 shows that the optimization converged when the number of training samples with $y > \beta$ reached 12%, which is quite close to the target pull rate 11%. We can see that the pull rate over the test set, achieved by the same threshold β , is also very close to the target 11%. This demonstrates that Eq. 9 provides a good approximation to $\frac{r}{1-r}$. In Figure 2, with the iterations, a steadily improving average collection amount among the top 11% is observed for both the training and test sets. We rarely observe obvious overfitting, and this justifies the use of the training set to choose γ and p .

Table 3 presents the average collection amount in the top 11% accounts over the test set for five different models. The classification model is an ensemble of 25 MLP networks with a modified class prior between 0.02 and 0.5 [10]. For the model of weighted classification, during training the samples are weighted by $\sigma(x) = \frac{1}{1+e^{-x}}$, which is also the target variable for the regression model. For ranking, most accounts (89%) have a tied collection amount of zero. We can see that the new algorithm is clearly exceeding all other algorithms. Comparing with the classification model, the ROI is improved by 25%. Note that the average collection amount over the whole portfolio is \$36 only.

new model	class.	weighted class.	ranking	regress.
\$157	\$126	\$106	\$61	\$116

Table 3: The average collection amount in the top 11% accounts over the test set for five different models.

⁴In some cases, by choosing different p values in Eq. 8 and Eq. 9, better results over the training set can be achieved.

⁵We have observed that the number of hidden units, varying from 0 to 10, does not have a significant effect on the results over our large data sets. Therefore all the MLP structures in the paper have 5 hidden units.

4. PREDICTING DEFECTION OF MUTUAL FUND ACCOUNTS

Worldwide the mutual fund industry houses 15 trillion US dollars – about 8 trillion from US investors and another 7 trillion from investors in other countries. Today, the US mutual fund industry holds about 18% of all households’ financial assets and about 22% of all outstanding US corporate stock [5]. However, in the end of 2003 the industry wide redemption rate stood at 24.2%, implying that the investor base completely turns over in 4 ($1/0.242$) years. To illustrate the magnitude of redemptions in the mutual fund industry, the Investment Company Institute estimated that in 2003 1.086 trillion new dollars flowed into equity funds but, over the exact same measurement interval, 934 billion (86%) flowed back out [5]. The costs associated with keeping track of this flowing river of money, adding and deleting client information to databases, filing required tax forms with federal, state and local taxing authorities as well as simply cutting checks to redeeming clients is an enormous drain on any funds’ expense ratio, not mentioning the revenue drop of fund companies because of the decreased assets under management (AUM) due to redemption. In recent years, more and more mutual fund companies have recognized the importance of early identification of investors at risk of redeeming their assets (i.e., defectors), so that proactive client service and educational programs could be initiated to “plug” the outflow of assets.

We have developed a model to predict account defection for a major US mutual fund company. In order to provide early identification of defectors, there is a two month gap between the end of the independent variable (IV) window and the beginning of the two-month dependent variable (DV) window. For example, at the end of February, we would like to predict which accounts will defect in the time period of May and June. The two-month leading time allows the mutual fund company to act on the predicted potential defectors in March and April. For classification purpose, a defector is defined by the domain experts as an account which had a net redemption amount (redemption minus purchase) of at least 35% of the account balance in a two month window. As the training set, we received about 184,000 accounts, each of which had an account balance of at least \$100,000. For training, the IV window is a one-year period ending on May 31, and the DV window is a two-month period of August and September. Based on the definition of defection, the defection rate is below 1% in the two month window. Regardless of the defection definition, the average net redemption amount in the two months over the whole training set was about -\$3,000, where the negative sign means that, on average, the account balance had a net increase. We used a forward time-shifted test set of around 434,000 accounts, which had the one-year IV window ending on September 30 and the DV window consisting of December and January.

The data for each account is a mixture of continuous and categorical variables, including basic account information, asset data, transactions, demographic information, benchmark performance data, and customer service records. There are about 2,000 raw data fields, but the final model uses 123 data fields after conducting feature selection and time series transformation [9]. The mutual fund company set $r = 10\%$ based on the predetermined budget. We will discuss the

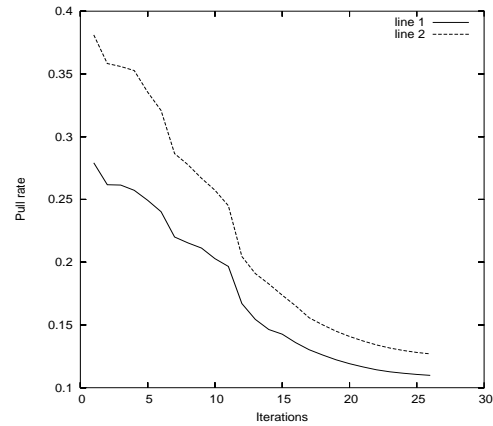


Figure 3: This figure shows convergence of pull rates achieved by the threshold β during the optimization. Line 1 is for the training set, and Line 2 shows the pull rate change over the test set.

savability issue in the next section. Until then let us assume that the return on investment is primarily dependent on the net redemption amount identified among the top 10%.

Again we fix β at 0.5 for the new algorithm and try to maximize the average net redemption amount in the top 10%. We chose $\gamma = 0$ and $p = 2$ since, with these parameters, the number of training samples with $y > \beta$ is close to 10% and the average collection amount among the top 10% in the training set is the largest. We have seen that these two goals are often quite consistent, i.e., when a set of parameters results in the largest average collection amount, it also brings the number of training samples with $y > \beta$ close to the target pull rate. We show the optimization process along with iterations of μ in Figures 3 and 4. Again we can see that the optimization converged when the number of training samples with $y > \beta$ reached 11%, which is quite close to the target pull rate 10%. Over the test set, the number of samples with $y > \beta$ reached 12%, 2% higher than the target rate. Figure 4 shows a quite large difference of the average net redemption amount between the training and test sets. This is due to the overall net redemption is changed in the test set’s DV window, which is four months apart from the training set DV window. The average net redemption amount over all the accounts is now about -\$3,400, comparing with -\$670 over the training set.

In Table 4, the classification model is an ensemble of 25 MLP networks trained with modified priors based on the defection definition of 35% or more redemption. However, the weighted classification model is trained by class labels based on the ranking, i.e., the samples with the top 10% of net redemption amount are positives and the others are negatives. The training samples are weighted by $\frac{1}{1+e^{-|x|}}$, which gives larger weights to samples with a larger (positive or negative) net redemption amount. The regression model is trained against $\sigma(x) = \frac{1}{1+e^{-x}}$. Note that in Table 4 a negative net redemption amount means a positive net purchase. Though the classification model achieves a 39% true positive rate (the new model has a true positive rate of 14%), it cannot effectively identify those accounts with the highest redemption amount.

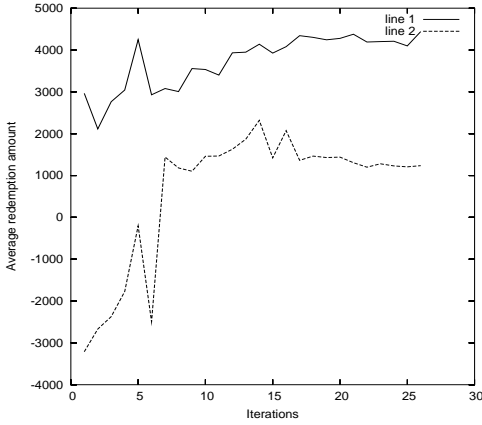


Figure 4: This figure shows the average net redemption amount change among the top 10% accounts during the optimization. Line 1 is for the training set, and Line 2 is over the test set.

new model	class.	weighted class.	ranking	regress.
\$1,236	-\$6,723	-\$18,968	-\$9,923	\$671

Table 4: The average net redemption amount in the top 10% accounts over the test set for five different models.

5. DISCUSSIONS

We have proposed a new learning algorithm which focuses on maximizing the monetary measure under a fixed budget constraint. The two applications demonstrate the substantial improvement of financial impact by the new algorithm. In this section, we discuss several related practical issues and some algorithmic and application extensions.

- Savability** There is no doubt that maximizing the collection amount for the top $r\%$ accounts also maximizes the ROI of the collection efforts which reach out to a predetermined $r\%$. However, it is arguable that maximizing the net redemption amount in the top $r\%$ accounts would maximize the *retained* redemption amount and thus the ROI. Another important factor determining the ROI is the savability of the predicted net redemption amount. It is reasonable to assume that it might be more difficult to retain a substantial redemption amount of a large account, since the redemption rate against the balance might be insignificant and the redemption is just some normal cash flow activity. To model ‘savability’ directly appears not feasible since ‘being savable’ is not observable and cannot be defined correctly. However, we have tried to model savability from the other aspect – ‘being unsavable’, which can be partially defined, i.e., if an account was predicted to defect and was contacted for retention but still became defected, this account was unsavable. Here we implicitly assume retention efforts do not cause originally non-defecting accounts to defect. We have developed such a savability model for a US mutual fund company. However, we have not been able to combine the savability model with scores from the model by the new algorithm in a principled way, since the score is only a ranking indicator rather

than a probabilistic estimate of x . An empirical way to consider savability is to increase the predetermined r by a certain percentage, and to exclude those accounts with a savability score below an empirically determined threshold.

- Valuable false positives** The model trained using the new algorithm does not classify accounts into positive and negative samples defined separately, e.g., by the redemption rate of 35%. We have observed that the model achieving the highest average net redemption amount can have a very low true positive rate based on the defection definition. Some companies will not feel comfortable to see a low true positive rate based on the defection definition given by their domain experts. It would be most desirable to achieve both a high true positive rate *and* a higher average redemption amount among the false positives. We call these false positives valuable false positives since they may have substantial net redemption too. We have tried to simply add an item, which approximates the true positive rate in [9], into Eq. 12 and to minimize the Lagrangian:

$$L = -\frac{1}{n} \sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta + \gamma) - \frac{1}{m} \sum_{i \in P} f(y_i, \beta + \gamma) + \frac{1}{\mu} \left(\frac{\sum_{i=0}^{n-1} f(y_i, \beta)}{\sum_{i=0}^{n-1} g(y_i, \beta)} - \frac{r}{1-r} \right)^2, \quad (16)$$

where $m = |P|$, the number of positive samples. However, this intuitive approach does not appear to work well.

- No budget constraint** In some cases, there is no fixed budget constraint and r is not predetermined. For example, for the collection industry, the goal might be loosely stated as collecting as much as possible by contacting as less accounts as possible. For this goal one might be tempted to minimize the following Lagrangian:

$$L = -\frac{1}{n} \sum_{i=0}^{n-1} x_i \cdot f(y_i, \beta + \gamma) + \frac{1}{\mu} \frac{\sum_{i=0}^{n-1} f(y_i, \beta)}{\sum_{i=0}^{n-1} g(y_i, \beta)}. \quad (17)$$

This approach does not work since it always tries to get to a contact rate close to zero. In theory, the maximum profit or ROI is achieved when the marginal collection cost equals to marginal revenue (collection amount). Typically, we can assume the marginal collection cost is a constant. By searching over different r values, for each of which a model needs to be trained by minimizing Eq. 12, the optimum r can be found so that the marginal collection cost is equal to the marginal revenue and the ROI is maximized.

- Other applications** The new algorithm can also be applied to several other areas. For example, maximizing average returns of stock selection, identifying tax auditing targets of highest values, and identifying fundraising targets with the highest contributions – all these tasks involve a predetermined budget and only concern the related average monetary value in the top $r\%$ determined by the budget. Even in the typical customer relationship management area, e.g., churn prediction for wireless service providers [10], since the

ultimate concern is the loss of revenue due to service disconnection, we can apply this algorithm to identify those accounts with the highest revenue losses. It would be interesting to compare this approach to another approach which combines an estimate of customer value with a predicted churn probability [7].

6. ACKNOWLEDGMENTS

The authors thank Bob Ruff, Vladimir Gold, and William Hastings at NCO Group, Inc. for helpful discussions and/or data preparation. This is a slightly revised version of our paper with the same title appearing in the Proceedings of KDD'06. The authors thank the workshop organizers for the invitation of this paper.

7. REFERENCES

- [1] Association of Credit and Collection Professionals. Industry Overview. <http://www.acainternational.org/about.aspx?cid=320>.
- [2] C. Burges, T. Shaked, et al. Learning to rank using gradient descent. In *Proc. of the 22nd Intl. Conf. on Machine Learning*, 2005.
- [3] R. Caruana, S. Baluja, and T. Mitchell. Using the future to “sort out” the present: Rankprop and multitask learning for medical risk evaluation. *Advances in Neural Information Processing Systems*, 8, pp. 959-965, 1996.
- [4] C. Elkan. The foundations of cost-sensitive learning. In *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence*, 2001.
- [5] Investment Company Institute. 2004 Mutual Fund Factbook. http://www.ici.org/stats/latest/12004_factbook.pdf, May 2004.
- [6] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45, pp. 503-528, 1989.
- [7] L. Yan and P. Baldasare. Optimizing customer value in a decision theoretic framework. In *Proc. of 2005 Intl. Workshop on CRM: Data Mining Meets Marketing*, New York, 2005.
- [8] L. Yan, R. Dodier, M. Mozer, and R. Wolniewicz. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *Proc. of the 20th Intl. Conf. on Machine Learning*, pp. 848-855, 2003.
- [9] L. Yan, M. Fasson, P. Baldasare. Enhancing the lift under budget constraints: an application in the mutual fund industry. In *Proc. of the 11th Intl. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 509-515, 2005.
- [10] L. Yan, R. Wolniewicz, and R. Dodier. Customer behavior prediction in telecommunications. *IEEE Intelligent Systems*, March/April 2004.

APPENDIX

Derivatives for minimizing Eq. 12

The following partial derivatives with the chain rule together form the basis to minimize Eq. 12 against the model parameters, e.g., the weights of an MLP structure.

Let

$$F = \sum_{i=0}^{n-1} f(y_i, \beta) \quad (18)$$

and

$$G = \sum_{i=0}^{n-1} g(y_i, \beta). \quad (19)$$

For $i \in \{i | f(y_i, \beta + \gamma) \leq 0, i = 0, \dots, n-1\}$, we have $\frac{\partial L}{\partial y_i} = 0$. For $i \in \{i | f(y_i, \beta + \gamma) > 0, i = 0, \dots, n-1\}$, we have the following two cases:

- When $f(y_i, \beta) > 0$,

$$\begin{aligned} \frac{\partial L}{\partial y_i} = & - \frac{p}{n} x_i (y_i - \beta - \gamma)^{p-1} \\ & + \frac{2p}{\mu} \left(\frac{F}{G} - \frac{r}{1-r} \right) \frac{1}{G} (y_i - \beta)^{p-1}. \end{aligned} \quad (20)$$

- When $g(y_i, \beta) > 0$,

$$\begin{aligned} \frac{\partial L}{\partial y_i} = & - \frac{p}{n} x_i (y_i - \beta - \gamma)^{p-1} \\ & + \frac{2p}{\mu} \left(\frac{F}{G} - \frac{r}{1-r} \right) \frac{F}{G^2} (\beta - y_i)^{p-1}. \end{aligned} \quad (21)$$