# Feature Dependency in Benefit Maximization: A Case Study in the Evaluation of Bank Loan Applications

**Nazlı İkizler and H. Altay Güvenir**
*Bilkent University*
*Department of Computer Engineering, 06533 Bilkent - Ankara*
{*inazli,guvenir*}*@cs.bilkent.edu.tr*

**Abstract**

In most of the real-world domains, benefit and costs of classifications can be dependent on the characteristics of individual examples. In such cases, there is no static benefit matrix available in the domain and each classification benefit is calculated separately. This situation, called *feature dependency*, is evaluated in the framework of our newly proposed classification algorithm *Benefit Maximizing classifier with Feature Intervals* (BMFI) that uses feature projection based knowledge representation. This new approach has been evaluated over bank loan applications and experimental results are presented.

**Keywords:** Machine learning, classification, cost-sensitivity, feature projections.

## 1 Introduction

Cost-sensitive classification branch of machine learning deals with non-uniform costs of classifications and aims to provide realistic learning models that are suitable for real-world applications. One important characteristic of a cost-sensitive domain is its sensitivity to differing costs of misclassifications. However, most of the studies in the cost-sensitive learning literature ignore the fact that these costs may not be homogeneous and can be dependent on individual examples. Consider the loan grant scheme: When a customer does not repay the credit amount he is granted, the bank loses the entire loan amount. On the other hand, if the bank refuses a good customer who is likely to pay the money back, the bank loses the interest amount that is proportional to the loan amount. This situation can be illustrated with the so-called benefit matrix shown in Table 1. Here "approve" means to grant the loan amount and "deny" means to reject the customer's request for loan. The term $f(x)$ in benefit table denotes the amount requested by customer $x$. Obviously, in such a situation, bank officials should be more careful with the high amount requests, because losses and gains will be much higher. This is also what the officials do in real world. For example, when a customer's request for $10000 is approved and he has defaulted, the benefit of the bank is -$10000, whereas in another application of the same case, if the loan amount is $100, the loss will be much lower, i.e., -$100. So, the benefit of the bank from each application is dependent on the loan amount feature of the application. Such a situation is an example of *feature-dependent* benefits in a domain.

This non-homogeneity in benefit and cost of classifications introduce a new dimension to the optimal classification problem. Classifiers should make decisions so as to maximize

the total benefit achieved in the result of the classification process. This sort of classification is different than the regular classification process. Instances are evaluated individually and depending on the relative importance of outcomes, a decision is made uniquely for each instance.

Table 1: Benefit matrix for a loan application domain where benefits are dependent on individual instances

| | Actual class | |
|---|---|---|
| Prediction | approve | deny |
| approve | $0.5f(x)$ | $-f(x)$ |
| deny | $-0.5f(x)$ | 0 |

Domains in which benefits can be feature-dependent can be categorized as follows:

- *Financial Domains*: As described above, in loan applications, benefits can be functions of the amount queried. In fraud detection of transaction problems, benefits are functions of transaction magnitudes. Moreover, in bankruptcy datasets, benefits might be represented as the size of the bank in dollars. Donation amount prediction as in KDD'98 Cup is another example domain for instance-dependent benefit amounts [1].
- *Medical Diagnosis Domains*: Benefits of classification can be based on the age of the patients. The younger the patient, the more effective a medication can be in some circumstances. Additionally, there may be temporal parameters associated with patient's health from which benefit functions can be estimated.
- *Temporal Domains*: In domains where benefits of decisions change over time, it would be more appropriate to specify $f(x)$'s in the benefit table as functions of time. For example, in geo-scientific predictions, like predicting earthquakes, natural disasters, time of prediction is a vital component and the benefit of prediction mostly depends on this parameter. The earlier the prediction is, the more precautions can be taken.
- *Spatial Domains*: Benefits can be represented as measures of distances in domains where the locality of prediction is important. In weather predictions for instance, the accuracy in the area of rainfall is important, and can be a functional parameter for degree of benefit.

In this study, we have analyzed an example domain, namely bank-loans domain, in which benefits are dependent on the amount of loan feature of individual instances. We present a naive approach that is incorporated into the feature projection method within the framework of BMFI algorithm. The paper is organized as follows: In Section 2, we present the related work in this area. In Section 3, we give the fundamentals of benefit maximization problem. Section 4 introduces our benefit maximization algorithm BMFI together with a brief presentation of feature intervals concept. In section 5, experimental results of BMFI in feature-dependent bank-loans data is presented. Section 6 concludes our discussion and outlines future research directions.

## 2 Related Work

In the literature of cost-sensitive learning, there are few studies that have included feature-dependent aspects of cost matrices. Fawcett et al. are among the first ones who incorporated feature-dependent costs in their classification problem [5]. In cellular phone cloning fraud detection, they used a variable cost matrix based on the fraudulent airtime.

In the credit card fraud detection domain, Chan et al. [2] represented the cost model in terms of overheads, which are equivalent to operational costs that are needed for each investigation and transaction amounts of instances. If the amount of transaction is smaller than the overhead, the net gain will be lower even if that transaction is fraudulent, so it is not worthwhile to make an investigation. They have examined the effects of cost-based sampling, which samples instances in proportion to their transaction amount ratios, but their concluded performance is not much different from random sampling.

Recently, Hollmen et al. [6] have examined feature-dependency and pointed out that there is an indisputable area of applications in which cost matrices of functions should be used instead of fixed cost matrices. They present a cost model and decision function based on Bayesian formulation. Posterior probabilities they use are obtained by a Hidden Markov model (HMM). The observed variables are assumed to be conditionally dependent on a discrete hidden variable in the HMM structure. Their input-dependent cost model exhibits promising results in terms of profit performance, when compared to cost-neutral and fixed-cost approaches in fraud detection of telecommunications domain. However, Hollmen et al. note that this approach is favorable when the input-dependent cost model is easily formulated.

Elkan [3] asks the question "What will happen if instance-dependent costs $C(i,j,x)$ are unknown for some labels $i$ and $j$, for some training examples $x$?". Such situations occur when costs are functions of features that are dependent on the class label, such as charity donation amounts, and are practically impossible to be known beforehand. The method Zadronzy et al. use to predict instance-dependent cost amounts is least-squares multiple linear regression [11]. By looking at the examples in the training set, the costs for test instances are predicted. Simple methods are used for probability and cost estimations. More sophisticated regression methods for cost estimation are likely to give more satisfactory improvements on this issue.

## 3 Benefit Maximization Problem

Recent research in machine learning has used the terminology of costs when dealing with misclassifications. However, those studies mostly lack the information that correct classifications may have different interpretations. Besides implying no cost, accurate labeling of instances may result in gains. Elkan points out the importance of these gains [4]. He states that doing accounting in terms of benefits is commonly preferable because there is a natural baseline from which all benefits can be measured, and thus, it is much easier to avoid mistakes.

Benefit concept is more appropriate to real world situations, since net flow of gain is more accurately denoted by benefits. If a prediction is profitable from the decision agent's point of view, its benefit is said to be positive. Otherwise, it is negative, which is the same as cost of wrong decision. To incorporate this natural knowledge of benefits to cost-sensitive learning, we have used *benefit matrices*.

***Definition*:** $B=[b_{ij}]$ is a $n \times m$ benefit matrix of domain $D$ if $n$ equals to the number of prediction labels, $m$ equals to the number of possible class labels in $D$ and $b_{ij}$'s are such that

$$b_{ij} = \begin{cases} \geq 0 & \text{if } i = j \\ < b_{ii} & \text{if } i \neq j \end{cases} \quad (1)$$

Here, $b_{ij}$ represents the benefit of classifying an instance of true class $j$ as class $i$. The structure of the benefit matrix is similar to that of the cost matrix, with the extension that entries can either have positive or negative values. In addition, diagonal elements should be non-negative values, ensuring that correct classifications can never have negative benefits.

Given a benefit matrix $B$, the optimal prediction for an example $x$ is the class $i$ that maximizes expected benefit ($EB$) defined as

$$EB(x,i) = \sum_j P(j \mid x) \times b_{ij} \quad (2)$$

where $P(j|x)$ is the probability that $x$ has true class $j$, The total expected benefit of the classifier model $M$ over the whole test data is

$$EB_M = \sum_x \arg\max_{i \in C} EB(x,i) = \sum_x \sum_j P(j \mid x) b_{ij} \quad (3)$$

where $C$ is the set of possible class labels in the domain.

## 4 BMFI with Feature-dependent Voting

As shown in [8], feature intervals classification is a rapid, user-friendly method that produces very effective results when compared to well-known classification algorithms. For this reason, we have chosen to extend its predictive power to involve benefit knowledge.

In a particular classification problem, given the training dataset which consists of $p$ features, an instance $x$ can be thought as a point in a $p$-dimensional space with an associated class label $x_c$. It is represented as a vector of nominal or linear feature values together with its associated class label, i.e., $<x_1,x_2,..,x_p,x_c>$. Here, $x_f$ represents the value of the $f$th feature of instance $x$. If we consider each feature separately, and take $x$'s projection onto each feature dimension, then we can represent $x$ by the combination of its feature projections.

Training process of BMFI algorithm is given in Fig. 1. In the beginning, for each feature $f$, all training instances are sorted with respect to their value for $f$. This sort operation is identical to forming projections of the training instances for each feature $f$. A *point interval* is constructed for each projection. Initially, lower and upper bounds of the interval are equal to the $f$ value of the corresponding training instance. If the $f$ value of a training instance is unknown, it is simply ignored. If there are several point intervals with the same $f$ value, they are combined into a single point interval by adding the class counts.

At the end of point interval construction, vote for each class label is assigned by a predefined voting method.

```
train(TrainingSet, BenefitMatrix)
begin
   for each feature f
      sort(f, TrainingSet)
      interval_list←make_point_intervals(f,TrainingSet)
      for each interval i in interval_list
        vote_i(c)← voting_method (i,f,BenefitMatrix)
       if f is linear
         interval_list←generalize(interval_list,BenefitMatrix)
end.
```

Fig. 1. Training stage of BMFI algorithm..

The voting method we use here is *expected benefit voting* (*EBV*) which is basically founded on optimal prediction approximation given by Eq. (2) and makes direct use of the benefit matrix. EBV casts votes to class $c$ in interval $I$ according to the following formulation:

$$EBV(c, I) = \sum_{k \in C} b_{ck} \times P(k \mid I) \qquad (4)$$

$P(k|I)$ is the estimated probability that an instance falling to interval $I$ will have the true class $k$, and is calculated as

$$P(k \mid I) = \frac{N_k}{classCount(k)} \qquad (5)$$

In feature projection concept, each feature is assumed to be an independent unit of knowledge and they individually contribute to voting with equal prediction power. However, in feature-dependent conditions, the dependent variable would have some effect on decision of other feature dimensions as well. For this reason, it is not very straightforward to incorporate feature dependency concerns to an autonomous environment like feature projections. In this study, we handle feature dependency on the dependent variable's dimension only. Fig. 2 summarizes the routine followed in feature-dependent voting scheme.

```
assign_dependent_votes (interval_list, f )
begin
for each i in interval_list do
        if f is a dependent variable
        /* take average of dependent value in the interval */
            avg ← average(i.upper , i.lower)
            vote_i(c) ← EBV(i,f,BenefitMatrix)×avg
        else
    vote_i(c) ← EBV(i,f,BenefitMatrix)
end.
```

Fig. 2: Algorithm for assigning feature-dependent votes

After the initial assignment of votes, point intervals are generalized to form *range intervals*, in order to eliminate redundancy and avoid overfitting. The generalization process is illustrated in Fig. 3. Here, *merge_condition*() is a comparison function that evaluates relative properties of each interval and returns true if sufficient level on similarity between those intervals is reached.

```
generalize (interval_list)
begin
   I← first interval in interval_list
   while I is not empty do
          I' is the interval after I
          I'' is the interval after I'
          if merge_condition(I,I',I'') is true
                 then merge I'(and/or I'') into I
          else I←I'
end.
```

Fig. 3 Generalization of intervals step

Besides adding more prediction power to the algorithm, proper generalization of intervals reduces the number of intervals, and by this way, decreases the classification time.

In this work, we have used three interval-join methods. The first one, called, SF (same frequent) joins two consecutive intervals if the most frequently occurring instances are of the same class. The second method, SB (same beneficial) joins two consecutive intervals if they have the same *beneficial class*. A class *c* is the beneficial class of an interval *i* iff for $\forall j \in C$ and $j \neq c$ $\sum_{x \in i} B(x,c) \geq \sum_{x \in i} B(x,j)$. If the beneficial classes of two consecutive intervals are the same, then it can be more profitable to unite them into a single interval. Third method, HC (high confidence) combines three consecutive intervals to a single one, when the middle interval has less confidence on its votes than the other two. The confidence of an interval is measured as the difference between benefits of the most beneficial class and second beneficial class. All these three interval generalization methods are applied in the presented order.

After the prediction model is constructed on the training data, it is time to classify previously unseen data. The classification (querying) process of the BMFI algorithm is given in Fig. 4. BMFI classification stage is very similar to that of CFI [7] and it involves a voting scheme where each feature acts as an independent unit and casts its vote for the particular instance's class.

The process starts by initializing the votes of each class label to zero. If the value of the query instance *q* for a feature *f*, i.e., $q_f$ is unknown (missing), then that feature does not involve in the voting process. Rather than altering the characteristics of the instance (i.e., by assigning average quantities for unknown feature values), simply ignoring that feature dimension is a more natural and straightforward way of handling missing values. If $q_f$ is known, then the interval *I* into which $q_f$ falls is searched. If the $q_f$ does not fall in any interval of *f*, then again, the feature *f* does not participate in the voting. If an interval *I* is

found that covers the $q_f$ value, then the votes of that particular interval are the votes it casts in the overall voting operation. Once all features have completed casting their votes, the class that received the highest amount of the votes is predicted as the class of the query instance $q$.

```
classify (q) /* q: query instance to be classified */
begin
   /* initialize total votes */
   for each class c
         v_c ← 0
   /*go over each feature dimension and sum up votes */
   for each feature f
         if q_f is known
            I ← search_interval(f, q_f)
         for each class c
            v_c ← v_c + interval_vote( I, c)
   /* predicted class is the one with the maximum votes */
   prediction ← arg max_c (v_c)
   return prediction
end.
```

Fig. 4: Classification phase of BMFI

When querying examples using a constant benefit table, the total benefit is calculated by simply adding up the corresponding benefit matrix entry for each test instance. In that case, each type of classification, e.g. classifying *i* as *j*, has identical revenue. On the other hand, if a feature dependent benefit table is available in the domain, for each query example there is a different benefit gained and this benefit is the functional form of feature values identified in the table. For each instance in the test set, these functional measures are summed up to calculate the total resultant benefit.

## 5 Experimental Results

Cost-sensitive algorithms cannot be evaluated by the standard classification accuracy metric, since the aim in this problem is to maximize the total benefit rather than to minimize the quantity of errors made. A simple representation that reflects the precision of classification in terms of benefit is *benefit accuracy*.

***Definition****: Benefit accuracy* of a classification model *M* in domain *D* over the instance set *S* is the normalized ratio of gained benefit to the maximum possible benefit, i.e.,

$$\text{Benefit Accuracy}_{\text{D}}(M,S) = \frac{\text{Benefit}_{\text{D}}(M,S) - \text{minB}_{\text{D}}(S)}{\text{maxB}_{\text{D}}(S) - \text{minB}_{\text{D}}(S)}$$

Here, $\text{Benefit}_D(M,S)$ is the benefit obtained by model $M$ on domain $D$, $\text{minB}_D(S)$ and $\text{maxB}_D(S)$ is the minimum and maximum benefit obtainable in domain $D$ respectively. That is, $\text{minB}_D(S)$ is the total benefit achievable when all the test instances are classified as the worst wrong case. Similarly, $\text{maxB}_D(S)$ is obtained when all instances in $S$ are classified

correctly. When $Benefit_D(M,S)$ is equal to the minimum benefit possible, then benefit accuracy of the model is 0. Correspondingly, when it equals the maximum benefit possible, then benefit accuracy is 1, as expected. In other words, this metric maps the obtained benefit to [0,1] range as in the case of conventional predictive accuracy. To be more specific, benefit accuracy is the general form of classical predictive accuracy. When the diagonal elements of the benefit matrix are one and non-diagonal elements are all zero, i.e., all types of classifications have equal importance and there is no cost for misclassifications, then benefit accuracy equals predictive accuracy used in comparison of error-based classifiers.

It should be noted that benefit accuracy metric not only compares relative benefits of two classifiers, but also indicates the algorithm's efficiency over a particular domain. For this reason, we have chosen to evaluate our algorithm mainly with regards to benefit accuracy. The accuracy results presented in this study are the average benefit accuracies achieved when 10-fold cross validation is utilized over the entire datasets. By this process, it is guaranteed that the training sets are disjoint and each instance in the whole dataset is classified exactly once. Resultant benefit accuracy is the average of the accuracy values of these 10 runs.

Bank-loans data is a direct application area for feature-dependency. The raw form of this dataset has been compiled by a private Turkish bank. We have preprocessed it by eliminating redundant data and missing attribute values. The entire raw dataset consists of more than 24000 instances. In this study, we have investigated a small, yet representative portion of it, consisting 1443 instances. In the literature of machine learning, this data has been investigated initially in [10]. There are 13 attributes in the domain, 7 of them are linear and 6 are categorical. The intention is to predict whether a customer is likely to pay the loan back or not.

If the benefit matrix is assigned so as to indicate the net cash flow in the bank with respect to granted loans, then for each customer, there will be a different benefit dependent on the amount requested. This situation can be formulated with the benefit matrix as follows:

| | **Actual class** | |
|---|---|---|
| **Prediction** | Repay | Default |
| Repay | $r \times la$ | $-la$ |
| Default | $-r \times la$ | $0$ |

Here, $r$ is the interest rate that the bank utilizes (logically, $0<r<1$) and $la$ is the loan amount that the customer requests. According to this matrix, if the money is granted and customer pays the loan back, then net money gain from the bank's perspective is $r \times la$. If money is not granted to a good customer who will pay it back, the bank loses $r \times la$ amount of profit. On the contrary, if the loan is granted but the customer does not pay it back, then the bank loses the entire loan amount. The net cash flow is 0 when no money is given to a bad customer who will actually default.

Fig. 5 illustrates the benefit accuracy of BMFI and its change with respect to interest rate that the bank uses for loan applications. In this chart, the results indicate that the lower

the interest rate, the higher benefit accuracy BMFI acquires. Since most of the Turkish banks apply a combined interest policy with an interest rate around 0.08 per month, then we can say that overall accuracy of BMFI on bank-loans domain is approximately 0.78.

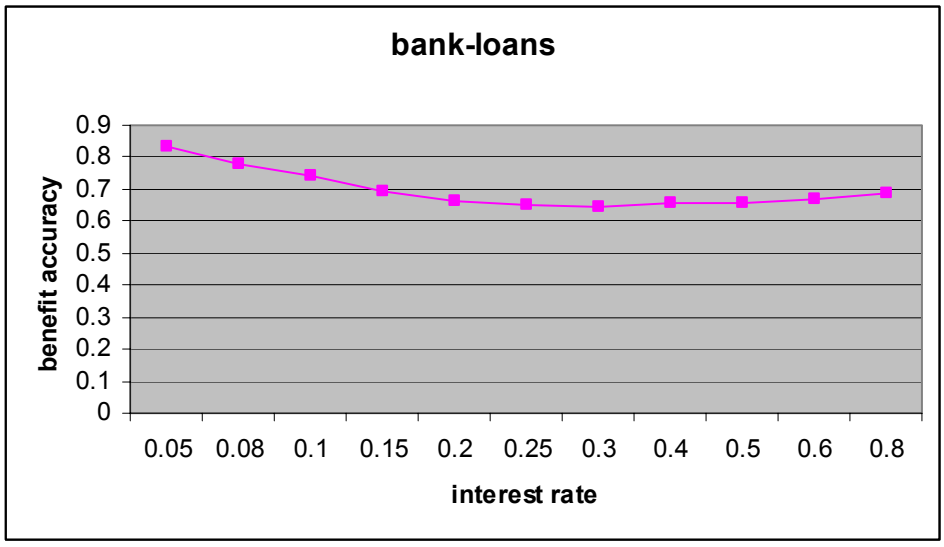


Fig. 5: Change in benefit accuracy with respect to interest rate in bank-loans domain

As the interest rate increases in this experiment, the ratio of importance between classes decreases and this causes a decrease in the performance of BMFI, since it is already observed that BMFI performs better when the benefit ratio between classes becomes higher [9].

## 6 Conclusions and Future Work

In the context of this study, we have dealt with situations when benefits are heterogeneous and dependent on the values of the features. This feature-dependency problem has been investigated using our newly proposed classification algorithm called Benefit Maximizing classifier with Feature Intervals (BMFI). BMFI uses the predictive power of feature projections concept and tries to maximize the total benefit gained for each single instance in consideration. We have tested feature-dependency aspect of BMFI over a recently constructed dataset, bank-loans data. We have achieved promising results in this domain.

As future work, benefit maximization system can be extended to include the feature costs. In order to accomplish this, feature selection mechanisms that are sensitive to individual costs of features can be utilized. This will make the classification algorithm more comprehensive and applicable in real-world domains. In addition, the situation where benefits are not known beforehand can be examined. Furthermore, this sort of benefit maximization research can be extended to handle incremental datasets.

## References

[1] KDD Cup 1998, http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html.

[2] P. Chan and S. Stolfo. Towards scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164-168, 1998.

[3] C. Elkan. Cost-sensitive learning and decision-making when costs are unknown. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning* (*WCSL at ICML-2000*)*,* Stanford University, California, 2000.

[4] C. Elkan. The Foundations of Cost-Sensitive Learning. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, 2001.

[5] T. Fawcett and F. Provost. Adaptive fraud detection. *Journal of Data Mining and Knowledge Discovery*, 1(3): 291-316, 1997.

[6] J. Hollmen, M. Skubacz, and M. Taniguchi. Input dependent misclassification costs for cost-sensitive classifiers. In *Proceedings of the Second International Conference on Data Mining*, pages 495-503, 2000.

[7] H. A. Güvenir. Detection of abnormal ECG recordings using feature intervals. In *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks* (*TAINN'2001*), A. Acan, I. Aybay, and M. Salamah (Eds.), pages 265-274, Gazimagusa, T.R.N.C., 2001.

[8] H. A Güvenir, G. Demiröz, and N. İlter. Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. *Artificial Intelligence in Medicine*, 13(3):147-165, 1998.

[9] N. İkizler. Benefit Maximizing Classification Using Feature Intervals Technical Report BU-CE-0208, Bilkent University, 2002.

[10] N. İkizler and H. A. Güvenir. Mining interesting rules in bank loans data. In *Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks* (*TAINN'2001*), A. Acan, I. Aybay, and M. Salamah (Eds.), pages 238-246, Gazimagusa, T.R.N.C., June 2001.

[11] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, AAAI Press (distributed by MIT Press), 2001.