Maximizing Benefit of Classifications Using Feature Intervals

Nazlı İkizler and H. Altay Güvenir

Bilkent University Department of Computer Engineering
06533, Ankara Turkey
{inazli,guvenir}@cs.bilkent.edu.tr

Abstract. There is a great need for classification methods that can properly handle asymmetric cost and benefit constraints of classifications. In this study, we aim to emphasize the importance of classification benefits by means of a new classification algorithm, *Benefit-Maximizing classifier with Feature Intervals* (BMFI) that uses feature projection based knowledge representation. Empirical results show that BMFI has promising performance compared to recent cost-sensitive algorithms in terms of the benefit gained.

1 Introduction

Classical machine learning applications try to reduce the quantity of the errors and usually ignore the quality of errors. However, in real-world applications, the nature of the error is very crucial. Further, the benefit of correct classifications may not be the same for all cases. *Cost-sensitive classification* research addresses this imperfection and evaluates the effects of predictions rather than simply measuring the predictive accuracy. By incorporating cost(and benefit) knowledge to the process of classification, the effectiveness of the algorithms in real-world situations can be evaluated more rationally. In this study, we concentrate on costs of misclassifications and try to minimize those costs, by maximizing the total benefit gained during the process of classification.

Within this framework, we propose a new cost-sensitive classification technique, called *Benefit-Maximizing classifier with Feature Intervals* (BMFI for short),that uses the predictive power of feature projection method previously proposed in [6]. In BMFI, voting procedure has been changed to impose the cost-sensitivity property. Generalization techniques are implemented to avoid overfitting and to eliminate redundancy. BMFI has been tested over several benchmark datasets and a number of real-world datasets that we have compiled.

The rest of the paper is organized as follows: In Section 2, benefit maximization problem is addressed. Section 3 gives the algorithmic descriptions of BMFI algorithm along with the details of feature intervals concept, voting method and generalizations. Experimental evaluation of BMFI is presented in Section 4. Finally, Section 5 reviews the results and presents future research directions on the subject.

2 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 entail indisputable gains. Elkan points out the importance of these gains [3]. 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 attained. 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*. $B=[b_{ij}]$ 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

$$b_{ij} = \begin{cases} \geq 0 & \text{if } i = j \\ < b_{ii} & \text{otherwise} \end{cases} . \tag{1}$$

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 instance x is the class i that maximizes expected benefit (EB), that is

$$EB(x,i) = \sum_j P(j|x) \times b_{ij} . \tag{2}$$

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

$$EB_M = \sum_x \arg\max_{i \in C} EB(x,i) = \sum_x \arg\max_{i \in C} \sum_j P(j|x) \times b_{ij} . \tag{3}$$

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

3 Benefit Maximization with Feature Intervals

As shown in [6], feature projections based classification is a fast and accurate method, and the rules it learns are easy for humans to verify. 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

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

Fig. 1. Training phase of BMFI

linear feature values together with its associated class, i.e., $<x_1, x_2, .., x_p, x_c>$. Here, x_f represents the value of the fth feature of the 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 determined by using one of the two voting methods. The first one is the voting method of CFI algorithm [5], called *VM1* in our context. VM1 can be formulated as follows:

$$VM1(c, I) = \frac{N_c}{classCount(c)} \; . \tag{4}$$

where N_c is the number of instances that belong to class c in interval I and $classCount(c)$ is the total number of instances of class c in the entire training set. This voting method favors the prediction of minority class in proportion to its occurrence in the interval. The second voting method, called *VM2*, is basically founded on optimal prediction approximation given by (2) and makes direct use of the benefit matrix. VM2 casts votes to class c in interval I as

$$VM2(c, I) = \sum_{k \in C} b_{ck} \times P(k|I) \; . \tag{5}$$

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

$$P(k|I) = \frac{N_k}{classCount(k)} \; . \tag{6}$$

```
generalize($interval\_list$)
begin
    $I \leftarrow$ first interval in $interval\_list$
    while $I$ is not empty do
        $I' \leftarrow$ interval after $I$
        $I" \leftarrow$ interval after $I'$
        if $merge\_condition(I, I', I")$ is true
            merge $I'$(and/or) $I"$ into $I$
        else $I \leftarrow I'$
end.
```

Fig. 2. Generalization of intervals step in BMFI

After the initial assignment of votes, for linear features, intervals are generalized to form *range intervals* in order to eliminate redundancy and avoid overfitting. The generalization process is illustrated in Fig. 2. Here, *merge_condition*() is a comparison function that evaluates relative properties of each interval and returns true if sufficient level of similarity between neighboring intervals is reached.

Besides adding more prediction power to the algorithm, proper generalization reduces the number of intervals, and by this way, decreases the classification time. In this work, we have experimented with three interval generalization methods. The first one, called SF (same frequent) joins two consecutive intervals if the most frequently occurring class of both are the same. 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. The third method, HC (high confidence) combines three consecutive intervals into 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 votes of the most beneficial class and second beneficial class.

Table 1. List of evaluated cost-sensitive algorithms

Name	Description
MetaNB	MetaCost on Naive Bayes
MetaJ48	MetaCost on J4.8
C1NB	CostSensitiveClassifier with reweighting on Naive Bayes
C2NB	CostSensitiveClassifier with direct minimization on Naive Bayes
C1J48	CostSensitiveClassifier with reweighting on J4.8
C2J48	CostSensitiveClassifier with direct minimization on J4.8
C1VFI	CostSensitiveClassifier with reweighting on VFI
C2VFI	CostSensitiveClassifier with direct minimization on VFI

```
classify($q$)
begin
    for each class $c$
        $v_c \leftarrow 0$
    for each feature $f$
        if $q_f$ is known
            $I \leftarrow$ search_interval($f, q_f$)
            for each class $c$
                $v_c \leftarrow v_c + interval\_vote(c, I)$
    $prediction \leftarrow argmax_c(v_c)$
end.
```

Fig. 3. Classification step in BMFI

The classification process of the BMFI algorithm is given in Fig. 3. The choice of voting method to be used depends on the characteristics of the domain. Based on our empirical results, we propose to use VM1 voting together with SF, SB and HC techniques when the correct classification of the minority class is more beneficial than the other classes. On the contrary, when the benefit matrix is not correlated with the distribution, VM2 can be employed together with SB and HC to boost up the benefit performance. Experimental results presented in Sect. 4 are achieved by using this general rule-of-thumb.

4 Experimental Results

For evaluation purposes, we have used benchmark datasets from UCI ML Repository [1]. These data sets do not have predefined benefit matrices, so we formed their benefit matrices in the following manner. In binary datasets, one class is assumed to be more important to predict correctly than the other by a constant benefit ratio, b. We have tested our algorithm by using five different b values that are 2, 5, 10, 20, 50. Note that when b is equal to 1, the problem reduces to the classical classification problem. Further, we have compiled four new datasets. Their benefit matrices have been defined by experts of each domain. For more information about the datasets and benefit matrices the reader is referred to [7]. We have compared BMFI with MetaCost [2] and CostSensitiveClassifier of Weka [4] on well-known base classifiers which are Naive Bayesian Classifier, C4.5 decision tree learner and VFI [6]. Table 1 lists these algorithms with their base classifiers (J4.8 is Weka's implementation of C4.5 in Java).

MetaCost is a wrapper algorithm that takes a base classifier and makes it sensitive to costs of classification [2]. It operates with a bagging logic beneath and learns multiple classifiers on multiple bootstrap replicates of the training set. MetaCost has become a benchmark for comparing cost-sensitive algorithms. In addition to MetaCost, we have compared our algorithm with two cost sensitive classifiers provided in Weka. The first method uses reweighting of training

Table 2. Comparative evaluation of BMFI with wrapper cost-sensitive algorithms. The entries are benefit per instance values. Best results are shown in bold

domain	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
breast-cancer	**4.0**	3.8	**4.0**	**4.0**	3.9	3.7	3.7	2.8	3.9 (VM1)
pima-diabetes	2.8	2.8	**3.0**	2.7	2.9	2.5	-1.5	2.8	2.7 (VM1)
ionosphere	5.7	**6.5**	6.1	6.0	**6.5**	5.7	6.4	6.1	**6.5** (VM2)
liver disorders	5.3	5.3	5.2	**5.4**	**5.4**	4.4	4.3	5.3	**5.4** (VM2)
sonar	3.3	4.6	4.5	4.0	4.6	3.3	0.0	4.0	**4.9** (VM2)
bank-loans	-0.8	-0.4	-0.9	-0.6	**0.1**	-0.5	-1.2	-2.8	-0.1 (VM1)
bankruptcy	7.8	7.5	7.7	7.4	7.5	7.3	7.7	7.8	**7.9** (VM1)
dermatology	7.5	7.2	**7.5**	**7.5**	7.2	7.3	6.9	5.6	7.4 (VM2)
lesion	8.7	7.8	8.9	**9.0**	7.8	7.7	6.4	4.0	**9.0** (VM1)

instances and the second method makes direct cost-minimization based on probability distributions [8]. We call these two classifiers C1 and C2, respectively.

Experimental results are presented in Table 2. In this table, results of binary datasets are benefit per instance values for b=10. All results are recorded by using 10-fold cross validation. As the results demonstrate, BMFI algorithm is very successful in most of the domains and remarkably comparable to other algorithms in all of the domains. In ionosphere, liver, sonar, bankruptcy and lesion domains, BMFI attains the maximum benefit per instance value. In the remaining datasets its performance is very high and comparable to other algorithms. We have observed that benefit achieved is highly dependent on the nature of the domain, i.e., benefit matrix information, distribution of classes, etc, as expected.

In addition, it is worthwhile to note that BMFI outperforms cost-sensitive versions of its base classifier VFI (C1VFI and C2VFI). This observation suggests that using benefit knowledge inside the algorithm itself is more effective than wrapping a meta-stage around to transform it into a cost-sensitive classifier.

In binary datasets, we observed that the success of BMFI increases as the benefit ratio increases. This is an important highlight of BMFI and is mostly due to its high sensitivity to benefit of classifications. This aspect of BMFI has been illustrated with the results of pima-diabetes dataset given in Table 3.

Table 3. Benefit per instance values of pima-diabetes dataset with different benefit ratios. Best results are shown in bold

b	MetaNB	MetaJ48	C1NB	C2NB	C1J48	C2J48	C1VFI	C2VFI	BMFI
2	0.5	0.6	**0.7**	0.6	0.6	0.6	0.0	0.0	0.5
5	1.2	1.2	**1.5**	1.3	1.2	1.2	-0.5	1.1	1.2
10	2.8	2.8	**3.0**	2.7	2.9	2.5	-1.5	2.8	2.7
20	5.8	5.8	6.2	6.1	6.1	5.6	-3.3	**6.3**	**6.3**
50	16.6	16.2	16.2	16.6	16.3	14.7	-9.0	16.7	**16.8**

5 Conclusions and Future Work

In this study, we have focused on the problem of making predictions when the outcomes have different benefits associated with them. We have implemented a new algorithm, namely BMFI that uses the predictive power of feature intervals concept in maximizing the total benefit of classifications. We make direct use of benefit matrix information provided to the algorithm in tuning the prediction so that the resultant benefit gain is maximized.

BMFI has been compared to MetaCost and two other cost-sensitive classification algorithms provided in Weka. These generic algorithms are wrapped over NBC, C4.5 and VFI. The results show that BMFI is very effective in maximizing the benefit per instance values. It is more successful in domains where the prediction of a certain class is particularly important. Empirical results we obtained also show that using benefit information directly in the algorithm itself is more effective than using a meta-stage around the base classifier.

In benefit maximization problem, we have observed that individual characteristics of the datasets influence results significantly, due to the extreme correlation between cost-sensitivity and class distributions.

As future work, feature-dependent domains can be explored in depth and feature-dependency aspect of BMFI can be improved. Benefit maximization can be extended to include the feature costs. Feature selection mechanisms that are sensitive to individual costs of features can be utilized.

References

[1] Blake C. L. and Merz C. J.: UCI repository of machine learning databases. University of California, Irvine, Department of Information and Computer Sciences (1998) [http://www.ics.uci.edu/~mlearn/MLRepository.html] 343

[2] Domingos P.: Metacost: A general method for making classifiers cost-sensitive. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, San Diego, CA (1999) 155-64 343

[3] Elkan C.: The Foundations of Cost-Sensitive Learning. In: Proceedings of the Seventeenth International Joint Conference on Articial Intelligence (2001) 340

[4] Frank E. et al.: Weka 3 - Data Mining with Open Source Machine Learning Software in Java. The University of Waikato (2000) [http://www.cs.waikato.ac.nz/~ml/weka] 343

[5] Güvenir H. A.: Detection of abnormal ECG recordings using feature intervals. In: Proceedings of the Tenth Turkish Symposium on Artificial Intelligence and Neural Networks (2001) 265-274 341

[6] Güvenir H. A, Demiröz G., and İlter N.: Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. Artificial Intelligence in Medicine, Vol. 13(3) (1998) 147-165 339, 340, 343

[7] İkizler N.: Benefit Maximizing Classification Using Feature Intervals. Technical Report BU-CE-0208, Bilkent University (2002) 343

[8] Ting K. M.: An instance weighting method to induce cost-sensitive trees. IEEE Transactions on Knowledge and Data Engineering, Vol. 14(3) (2002) 659-665 344