One-Benefit learning: Cost-sensitive learning with restricted cost information

Bianca Zadrozny
IBM T.J. Watson Research Center
1101 Kitchawan Road, Route 134
Yorktown Heights, NY 10598
zadrozny@us.ibm.com

ABSTRACT

This paper presents a new formulation for cost-sensitive learning that we call the One-Benefit formulation. Instead of having the correct label for each training example as in the standard classifier learning formulation, in this formulation we have one possible label for each example (which may not be the correct one) and the benefit (or cost) associated with that label. The goal of learning in this formulation is to find the classifier that maximizes the expected benefit of the labelling using only these examples. We present a reduction from One-Benefit learning to standard classifier learning that allows us to use any existing error-minimizing classifier learner to maximize the expected benefit in this formulation by correctly weighting the examples. We also show how to evaluate a classifier using test examples for which we only the benefit for one of the labels. We present preliminary experimental results using a synthetic data generator that allows us to test both our learning method and our evaluation method.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning - Induction; H.2.8 [**Database Management**]: Applications - Data Mining

General Terms

Algorithms

Keywords

data mining, cost-sensitive learning

1. INTRODUCTION

In standard classifier learning, we are given a training set of examples of the form (x, y), where x is a feature vector and y is a class label. These examples are assumed (at least, implicitly) to be drawn independently from a fixed distribution D with domain $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a feature space and $\mathcal{Y}$ is a (discrete) class label space. The goal is to learn a classifier $h : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected error rate on examples drawn from D, given by

$$E_{x,y\sim D}[I(h(x) \neq y)] \tag{1}$$

where $I(\cdot)$ is the indicator function that has value 1 in case its argument is true and 0 otherwise.

The traditional formulation assumes that all errors are equally costly. However, this is not true for many domains for which one would like to obtain classifiers. For example:

- In one-to-one marketing, the cost of making an offer to a person who does not respond is small compared to the cost of not contacting a person who would respond.
- In medicine, the cost of prescribing a drug to an allergic patient can be much higher than the cost of not prescribing the drug to a nonallergic patient.
- In image or text retrieval, the cost of not displaying a relevant item may be lower or higher than the cost of displaying an irrelevant item.

One extension to the standard classifier learning formulation that has received considerable attention in the past few years is the cost matrix formulation [2, 4, 3]. In this formulation, we specify a cost matrix C for the domain in which we would like to learn a classifier. If there are k classes, the cost matrix is a $k \times k$ matrix of real values. Each entry $C(i, j)$ gives the cost of predicting class i for an example whose actual class is j. Now, instead of minimizing the error rate given by equation 1, we would like to find a classifier h that minimizes the expected cost of the labeling, given by

$$E_{x,y\sim D}[C(h(x), y)]. \tag{2}$$

Research on cost-sensitive learning has traditionally been couched in terms of costs, as opposed to benefits or rewards. However, in many domains, it is easier to talk consistently about benefits than about costs. The reason is that all benefits are straightforward cash flows relative to a baseline wealth of \$0, while some costs are counterfactual opportunity costs [3]. Instead of specifying a cost matrix, we can equivalently specify a benefit matrix B, where each entry of the matrix describes the benefit (or reward) of predicting class i for an example whose actual class is j. Then, instead of minimizing 2, we maximize

$$E_{x,y\sim D}[B(h(x), y)].$$

The benefit matrix formulation assumes that the benefits are fixed, i.e., that they only depend on the predicted and actual classes, but not on the example itself. However, more often than not, benefits in real-world domains are example-dependent. For example, in direct marketing, the benefit of classifying a respondent correctly depends on the profit that the customer generates. Similarly, in credit card fraud detection, the benefit of correctly identifying a fraudulent transaction depends on the amount of the transaction.

Zadrozny and Elkan [6] extend the benefit matrix formulation to the example-dependent case by allowing each entry to depend on the particular feature vector x. In this case, the benefits are given by a function $B(i, j, x)$, where i is the predicted class, j is the actual class and x is the feature vector of the example. Accordingly, we would now like to find a classifier h that maximizes the expected benefit of the labeling, given by

$$E_{x,y\sim D}[B(h(x), y, x)]. \qquad (3)$$

Because the benefit matrix formulation assumes that the benefits are fixed for all examples, it also implicitly assumes that they are known in advance. However, when we allow example-dependent benefits, it might be the case that the benefits are not known for all of the possible labels of all the training examples. An example of an application where this is the case is direct marketing. In this case, x is the description of a customer (which may include, for example, past purchases) and y is a marketing action (such as mailing a catalog or a coupon). The benefit for each y is the profit attained if the customer responds to the action or \$0 if he does not respond to it. Therefore, in order to measure the benefits for each possible y, it would be necessary to take all possible actions with the same customer, which is not feasible[1].

A similar situation occurs in medical treatment, here x is the description of a patient and y is a possible treatment. Usually only one treatment is assigned to each patient, so we only have information about the benefit of one treatment per person. Therefore, the example-dependent formulation is not directly applicable here.

In this paper, we introduce a new formulation of the cost-sensitive learning problem that does not require that all of the benefits (or costs) for each of the labels are known for each training example. We call this formulation the One-Benefit formulation. We present an algorithm for learning under this formulation that is in fact a reduction to standard classifier learning. In other words, it is an algorithm that transforms a cost-sensitive learning problem of this type into a standard classifier learning problem. We call this reduction the One-Benefit reduction. The main advantage of a reduction is that it allows the use of any classifier learner as a black box. For other details and advantages of reductions, see Beygelzimer et al. [1]. We also present a method for evaluating classifiers under the One-Benefit formulation. Finally, we show some preliminary results on a synthetic dataset.

[1]Note that previous work in cost-sensitive learning applied to direct marketing[6] dealt with a special case in which there were only two possible actions (mail/not mail). The "not mail" action had a fixed benefit of zero and all the customers in the training set had been mailed. Therefore, the benefits for each label were known for each training example

2. CLASSIFIER LEARNING UNDER THE ONE-BENEFIT FORMULATION

We assume that we have m training examples (x, y, b) drawn from a joint distribution D with domain $\mathcal{X} \times \mathcal{Y} \times \mathcal{B}$ where $\mathcal{X}$ is an (arbitrary) space, $\mathcal{Y}$ is a (discrete) label space and $\mathcal{B}$ is a (nonnegative, real) benefit space, where the benefit of assigning label y to example x is is given by a stochastic function $B : \mathcal{X} \times \mathcal{Y} \to [0, \infty]$ (that is $b \sim B(x, y)$).

We call this formulation the One-Benefit formulation. Note that instead of having the correct label for each training example as in the standard classifier learning formulation, in this formulation we have one possible label (which may not be the correct one) and the benefit associated with that label. If we know the benefit of more than one possible label, we can create one example per benefit.

Our goal is to find the classifier $h : \mathcal{X} \to \mathcal{Y}$ that maximizes the expected value of the benefit given by

$$E_{x\sim D}[B(x, h(x))] \qquad (4)$$

using only the available examples.

We also want to be able to evaluate an existing classifier using only examples of the form (x, y, b). That is, we want to be able to obtain an estimate of the expected benefit of the classifier (given by equation 4).

Standard classifier learners try to find H to maximize the accuracy

$$\frac{1}{m}\sum_{(x,y)} I(H(x) = y)$$

but, according to the translation theorem in Zadrozny et al. [8], can be made to maximize a weighted loss

$$\frac{1}{m}\sum_{(x,y,w)} wI(H(x) = y), \qquad (5)$$

where w is a importance weight given to each example.

The following theorem shows that the expected benefit in (4) can be rewritten in a way that allows us to use a classifier learner that maximizes (5) to learn the classifier h.

Theorem 1. *For all distributions, D, for any deterministic function, $h : \mathcal{X} \to \mathcal{Y}$ and for any stochastic function $B : \mathcal{X} \times \mathcal{Y} \to [0, \infty]$, if we assume that $P(y|x) > 0$ $\forall x, y$ then*

$$E_D[B(x, h(x))] = E_D\left[\frac{b}{P(y|x)} I(h(x) = y)\right]$$

PROOF.

$$
\begin{aligned}
& E_D\left[\frac{b}{P(y|x)} I(h(x)=y)\right] \\
&= E_{x,y,b\sim D}\left[\frac{B(x,y)}{P(y|x)} I(h(x)=y)\right] \\
&= E_D\left[\frac{B(x,y)}{P(y|x)} \middle| h(x)=y\right] P(h(x)=y) \\
&= E_D\left[\frac{b(x,h(x))}{P(h(x)=y|x)} \middle| h(x)=y\right] P(h(x)=y) \\
&= \int_x \frac{B(x,h(x))}{P(h(x)=y|x)} P(x|h(x)=y)P(h(x)=y)dx \\
&= \int_x \frac{B(x,h(x))}{P(h(x)=y|x)} P(x,h(x)=y)dx \\
&= \int_x B(x,h(x))P(x)dx \\
&= E_D[B(x,h(x))]
\end{aligned}
$$

□

From this theorem, it follows that

$$\frac{1}{m}\sum_{(x,y,b)} \frac{b}{P(y|x)} I(h(x)=y) \tag{6}$$

is an unbiased empirical estimate of the expected benefit of classifier h. Thus, if we know $P(y|x)$, that is, the probability that label y is assigned to example x in the training data, we can use a classifier learner to learn the classifier from these examples. Looking back at (5) we see that we simply have to weigh each example (x, y, b) by $\frac{b}{P(y|x)}$.

Note that the theorem holds only if $\forall x, y\ P(y|x) > 0$, that is, in order to guarantee convergence to the optimal classifier, we require that in the training data each label have non-zero probability of being assigned to each example. However, the reduction degrades gracefully even when this is not the case if we define that

$$\frac{I(h(x)=y)}{P(y|x)} = 0$$

when $I(H(x) = y) = 0$ and $P(y|x) = 0$. In this case, it is easy to see that the reduction will converge to a classifier that is optimal, except that label y is not allowed for example x.

This theorem demonstrates that in this formulation we have to account both for the benefits (given by the numerator in the first factor of equation 6) and for the fact that the labels are not assigned at random to the examples (given by the denominator in the first factor of equation 6).

Zadrozny et al.[8] showed that learning from a weighted distribution of examples is not straightforward with many classifier learners but that "costing", a method based on rejection sampling, achieves good results in practice. For this reason, we recommend using costing here, where instead of using misclassification costs as weights we use the ratio $\frac{b}{P(y|x)}$ as a weight for each example (x, y, b). Another option is to use learners that accept weights directly, such as naive Bayes and SVM.

In practice, we may not know the probabilities $P(y|x)$ for the training examples in advance. However, we can estimate these using the available training data by applying a classifier learning method that estimates conditional probabilities or by transforming the outputs of a classifier into accurate probability estimates [7].

```
One-Benefit Reduction(Training Set $S = (x, y, b)$)
  1. Learn a model for $P(y|x)$ using $S$.
  2. Calculate a weight for each example $(x, y, b)$: $w = \frac{b}{P(y|x)}$
  3. Learn a classifier $h$ using a cost-sensitive learner on $S' = (x, y, w)$.
  4. Output $h$.
```

Table 1: The One-Benefit Reduction.

Table 1 shows the pseudo-code for this reduction. Given a training set of the form (x, y, b), we first learn a model for $P(y|x)$. This can be accomplished by using a classifier learner that outputs class membership probability estimates. We then calculate weights for each example (x, y, b) by dividing b by $P(y|x)$. We can now use a cost-sensitive learning method that takes examples (x, y, w) as input, such as the ones presented in Zadrozny et al.[8] to learn a classifier that maximizes the expected benefit.

3. CLASSIFIER EVALUATION UNDER THE ONE-BENEFIT FORMULATION

The most obvious way to estimate the expected benefit of a classifier h in the One-Benefit formulation is to select the test examples (x, y, b) for which $h(x) = y$, since these are the examples that tell us the benefit of the label predicted by the classifier. Then, we can average the benefits b from each of the selected examples to obtain an estimate of the expected benefit of the classifier. This is reasonable if the number of possible labels is small, so that we can obtain enough examples that agree with the classifier.

Nonetheless, even when this condition is true, selecting the examples in this manner may result in a biased estimate of the expected benefit of the classifier. This can happen because the examples are being selected according to a criteria that is not necessarily independent of the feature vector x. For example, in the direct marketing case, each example x describes a particular customer. If we have a classifier that is more likely to agree with the data for the "rich customers" (who presumably tend to buy more), by using this kind of evaluation we may think it is a very good classifier. However, if we apply the classifier to the general population of customers it may not perform as well.

As we did for learning, we can use theorem 1 for evaluation. According to theorem 1, the expected benefit of a classifier h is given by

$$E_D[B(x,h(x))] = E_D\left[\frac{b}{P(y|x)} I(h(x)=y)\right].$$

Therefore, an empirical estimate of the expected benefit of the classifier is the following sum for a set of m test examples:

$$\frac{1}{m}\sum_{(x,y,b)} \frac{b}{P(y|x)} I(h(x)=y),$$

that is, we sum over the test examples whose labels agree with the label selected by the classifier h, but we weigh them by the ratio of their benefit divided by the conditional probability that the label appears in the data. Again, the probabilities $P(y|x)$ have to be estimated (and validated) using the training data.

4. EXPERIMENTAL RESULTS

We present experimental results using a synthetic data generator that is a modification of the IBM Quest Synthetic Data Generation Code for classification (Quest)[5]. Quest randomly generates examples for a person data set in which each person has the nine attributes described below.

- `Salary`: uniformly distributed between 20000 and 150000.
- `Commission`: if `Salary` ≥ 75000, `Commission` $= 0$, else uniformly distributed between 10000 and 75000.
- `Age`: uniformly chosen from 60 integer values (20 to 80).
- `Education`: uniformly chosen from 4 integer values.
- `CarMake`: uniformly chosen from 20 integer values.
- `ZipCode`: uniformly chosen from 9 integer values.
- `HouseValue`: uniformly distributed from $50000\,k$ to $150000\,k$, where $0 \leq k \leq 9$ and depends on the `ZipCode`.
- `YearsOwned`: uniformly distributed from 1 to 30.
- `Loan`: uniformly distributed between 0 and 500000.

In the original Quest generation code, there are a series of classification functions of increasing complexity that used the above attributes to classify people into different groups. After determining the values of different attributes of an example and assigning it a group label according to the classification function, the values for non-categorical attributes are perturbed. If the value of an attribute A for an example x is v and the range of values of A is a, then the value of A for x after perturbation becomes $v + r * a$, where r is a uniform random variable between -0.5 and +0.5.

We modified Quest to include both label generation functions and benefit generation functions. These are used to generate examples of the form (x, y, b), where x is a person described by the attributes above, y is a label describing a marketing action taken for that person (such as mailing a particular catalog) and b is the benefit received after the action described by y is taken (such as the amount purchased from the catalog).

We use two different label generation functions that were created based on classification functions already implemented in Quest. The functions are non-deterministic. For each example, they specify a probability for each label. The labels are then randomly drawn according to these probabilities. The label generation functions are shown in Table 2.

Given an example x and a label y, the benefit generation function determines a benefit for labeling person x as belonging to class y. We use two different benefit generation functions, which are shown in Table 3.

Labelling Function 1

```
if (Age< 40)
 if (50000≤Salary≤100000)
  probClass1 = 0.3;
 else
  probClass1 = 0.7;
else
if (40 ≤Age< 60)
 if (75000≤Salary≤125000)
  probClass1 = 0.1;
 else
  probClass1 = 0.9;
else
if (25000≤Salary≤75000)
  probClass1 = 0.4;
 else
  probClass1 = 0.6;

if (probClass1>rand())
 y=1;
else
 y=0;
```

Labelling Function 2

```
if (Age<40)
 probClass1 = 0.2;
else
if (40≤Age<60)
 probClass1 = 0.8;
else
 probClass1 = 0.2;

if (probClass1>rand())
 y=1;
else
 y=0;
```

Table 2: Label generation functions. The function `rand()` generates a random number drawn uniformly from the interval $[0, 1]$.

Benefit Function 1

```
if (YearsOwned<20)
 equity = 0;
else
 equity = 0.1*YearsOwned
         - 2;

disposable = 2*Salary/3
       - Loan/5
       + 5000*Education
       + equity/5
       - 10000;

if (disposable>0)
 if (y = 0)
  b = randn(250,20);
 else
  b = randn(200,20);
else
 if (y = 0)
  b = randn(80,20);
 else
  b = randn(150,20);
```

Benefit Function 2

```
if (Age<40)
 if (Education∈ {0,1})
  if (y= 0)
   b = randn(100,20);
  else
   b = randn(80,20);
 else
  if (y= 0)
   b = randn(50,20);
  else
   b = randn(120,20);
else
if (40 ≤Age< 60)
 if (Education∈ {1,2,3})
  if (y= 0)
   b = randn(100,20);
  else
   b = randn(150,20);
 else
  if (y= 0)
   b = randn(120,20);
  else
   b = randn(140,20);
else
 if (Education∈ {2,3,4})
  if (y= 0)
   b = randn(90,20);
  else
   b = randn(70,20);
 else
  if (y= 0)
   b = randn(50,20);
  else
   b = randn(70,20);
```

Table 3: Benefit generation functions. The function `randn(`μ`,`σ`)` generates a random number drawn from a Gaussian with mean μ and standard deviation σ.

The advantage of using a synthetic data generator is that we can evaluate any classifier by generating the benefits for each possible action, which is not possible with real data. In the real-world, we cannot "reset" customers to the same state and mail a different catalog as if the customer had not received the first one, but we can do this with Quest.

We applied the One-Benefit reduction 1 to three training sets of 50000 examples generated using three settings of the label and benefit generation functions (Label1-Benefit1, Label1-Benefit2 and Label2-Benefit2).For obtaining the estimates of $P(y|x)$ we use naive Bayes followed by the PAV calibration algorithm [7]. For learning the main classifier, we use three methods:

- weighted Naive Bayes,
- costing with Naive Bayes as base learner,
- costing with C4.5 as base learner.

For evaluating the classifiers, we use the simulator to generate three test sets of 50000 examples. We evaluate the classifiers using three methods:

- **True:** use the generator to obtain benefit values for the two labels for each test example and average the benefits for the labels chosen by the classifier (unbiased but unrealistic in a data mining setting).
- **Biased:** select only the test examples that agree with the classifier and average the benefits for those examples.
- **Corrected:** select only the test examples that agree with the classifier and use the bias correction method proposed in Section 3 to calculate the expected benefit of the classifier (unbiased and realistic).

The probabilities $P(y|x)$ necessary for the bias correction method are obtained by applying the model learned on the training set to the test examples.

Table 4 summarizes the results obtained. For comparison purposes, it also includes the average benefit of the training labels, of the best possible labelling and of the worst possible labelling.

In all cases, the One-Benefit reduction improves upon the training labels. Furthermore, by comparing the two settings with the same benefit function and different training labelling, we see that the particular training labelling does not greatly influence the final result. The different learning algorithms (weighted NB, costing NB and costing C4.5) in general led to classifiers that are equally good, except that costing C4.5 resulted in a better classifier for the settings with Benefit2.

Whereas using only the selected examples to evaluate the classifier yields incorrect estimates of the value of the classifier, the evaluation using the bias correction method yields results that are very close to the true (but unrealistic) evaluation.

Labelling Function 1 - Benefit Function 1

	Evaluation Method		
Classifier	True	Biased	Corrected
worst possible	146.94	-	-
best possible	206.31	-	-
training labels	178.06	-	-
weighted NB	192.74	180.74	191.86
costing NB	192.30	180.80	191.80
costing C.45	190.94	180.23	190.78

Labelling Function 1 - Benefit Function 2

	Evaluation Method		
Classifier	True	Biased	Corrected
worst possible	73.87	-	-
best possible	116.01	-	-
training labels	102.99	-	-
weighted NB	107.21	115.65	107.85
costing NB	107.06	115.53	107.76
costing C.45	112.45	120.30	112.56

Labelling Function 2 - Benefit Function 2

	Evaluation Method		
Classifier	True	Biased	Corrected
worst possible	73.87	-	-
best possible	116.01	-	-
training labels	96.12	-	-
weighted NB	107.08	112.20	108.50
costing NB	107.09	112.34	108.56
costing C.45	112.67	116.41	112.52

Table 4: Experimental results.

5. CONCLUSIONS

We present here a new formulation for the cost-sensitive learning problem that we call the One-Benefit formulation. Instead of assuming that the benefits for each of the labels is known for each training example as in previous cost-sensitive formulations, this formulation only assumes that the benefit for one of the possible labels for each training example is known at training time. We argue that this is a realistic setup for some cost-sensitive domains such as direct marketing and medical treatment.

We show that it is possible to learn under this formulation by presenting a reduction from One-Benefit learning into standard classifier learning. The reduction requires that we first learn $P(y|x)$, that is, the conditional probability of the labels that appear in the training data, and then use the benefits and the conditional probabilities to correctly weigh the training data before applying the classifier learner. We also show how to correctly evaluate a classifier when only One-Benefit test examples are available, again by correctly weighting the examples.

We present some preliminary experimental results using a synthetic data generator. The advantage of using the generator is that we can evaluate the classifiers without resorting to the proposed evaluation method and, therefore, we can assess the accuracy of our evaluation method. Our results show that by using the One-Benefit reduction it is possible to learn a classifier that has greater expected benefit than the classifier used to label the training examples. Also, our proposed evaluation method succeeds in correctly measuring the expected benefit of the classifiers.

In future work, we would like to apply this method to real data sets from the direct marketing and medical treatment domains.

6. ACKNOWLEDGEMENTS

I thank Charles Elkan, John Langford and Roberto Oliveira for the many helpful conversations about the topic of this paper.

7. REFERENCES

[1] A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Error limiting reductions between classification tasks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005. To appear.

[2] P. Domingos. MetaCost: A general method for making classifiers cost sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[3] C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, Aug. 2001.

[4] D. Margineantu. Class probability estimation and cost-sensitive classification. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 270–281, 2002.

[5] R. Srikant. IBM Quest Synthetic Data Generation Code, 1999. Available at `http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html`.

[6] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 204–213, 2001.

[7] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699, 2002.

[8] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 435–442, 2003.