To appear in *Encyclopedia of Machine Learning*. C. Sammut (Ed.). Springer. 2008

Cost-Sensitive Learning and the Class Imbalance Problem

Charles X. Ling, Victor S. Sheng
The University of Western Ontario, Canada

Synonyms
Learning with different classification costs, cost-sensitive classification

Definition

Cost-Sensitive Learning is a type of learning in data mining that takes the misclassification costs (and possibly other types of cost) into consideration. The goal of this type of learning is to minimize the total cost. The key difference between cost-sensitive learning and cost-insensitive learning is that cost-sensitive learning treats the different misclassifications differently. Cost-insensitive learning does not take the misclassification costs into consideration. The goal of this type of learning is to pursue a high accuracy of classifying examples into a set of known classes.

The class imbalanced datasets occurs in many real-world applications where the class distributions of data are highly imbalanced. Cost-sensitive learning is a common approach to solve this problem.

Motivation and Background

Classification is the most important task in inductive learning and machine learning. A classifier can be trained from a set of training examples with class labels, and can be used to predict the class labels of new examples. The class label is usually discrete and finite. Many effective classification algorithms have been developed, such as naïve Bayes, decision trees, neural networks, and so on. However, most original classification algorithms pursue to minimize the error rate: the percentage of the incorrect prediction of class labels. They ignore the difference between types of misclassification errors. In particular, they implicitly assume that all misclassification errors cost equally.

In many real-world applications, this assumption is not true. The differences between different misclassification errors can be quite large. For example, in medical diagnosis of a certain cancer, if the cancer is regarded as the positive class, and non-cancer (healthy) as negative, then missing a cancer (the patient is actually positive but is classified as negative; thus it is also called "false negative") is much more serious (thus expensive) than the false-positive error. The patient could lose his/her life because of the delay in the correct diagnosis and treatment. Similarly, if carrying a bomb is positive, then it is much more expensive to miss a terrorist who carries a bomb to a flight than searching an innocent person.

Cost-sensitive learning takes costs, such as the misclassification cost, into consideration. It is one of the most active and important research areas in machine learning, and it plays an important role in real-world data mining applications. (Turney, 2000) provides a comprehensive survey of a large variety of different types of costs in data mining and machine learning, including misclassification costs, data acquisition cost (instance costs and attribute costs), active learning costs, computation cost, human-computer interaction cost, and so on. The misclassification cost is singled out as the most important cost, and it has also been mostly studied in recent years.

Theory

We summarize the theory of cost-sensitive learning, published mostly in (Elkan, 2001; Zadrozny and Elkan, 2001). The theory describes how the misclassification cost plays its essential role in various cost-sensitive learning algorithms.

Without loss of generality, we assume binary classification (i.e., positive and negative class) in this paper. In cost-sensitive learning, the costs of false positive (actual negative but predicted as positive; denoted as *FP*), false negative (*FN*), true positive (*TP*) and true negative (*TN*) can be given in a cost matrix, as shown in Table 1. In the table, we also use the notation *C(i, j)* to represent the misclassification cost of classifying an instance from its actual class *j* into the predicted class *i*. (We use 1 for positive, and 0 for negative). These misclassification cost values can be given by domain experts, or learned via other approaches. In cost-sensitive learning, it is usually assume that such a cost matrix is given and known. For multiple classes, the cost matrix can be easily extended by adding more rows and more columns.

Table 1. An example of cost matrix for binary classification.

	Actual negative	Actual positive
Predict negative	C(0,0), or TN	C(0,1), or FN
Predict positive	C(1,0), or FP	C(1,1), or TP

Note that *C(i, i)* (*TP* and *TN*) is usually regarded as the "benefit" (i.e., negated cost) when an instance is predicted correctly. In addition, cost-sensitive learning is often used to deal with datasets with very imbalanced class distribution (Japkowicz and Stephen, 2002). Usually (and without loss of generality), the minority or rare class is regarded as the positive class, and it is often more expensive to misclassify an actual positive example into negative, than an actual negative example into positive. That is, the value of *FN* or *C(0,1)* is usually larger than that of *FP* or *C(1,0)*. This is true for the cancer example mentioned earlier (cancer patients are usually rare in the population, but predicting an actual cancer patient as negative is usually very costly) and the bomb example (terrorists are rare).

Given the cost matrix, an example should be classified into the class that has the minimum expected cost. This is the minimum expected cost principle. The expected cost *R(i|x)* of classifying an instance *x* into class *i* (by a classifier) can be expressed as:

$$R(i \mid x) = \sum_{j} P(j \mid x) C(i, j), \qquad (1)$$

where *P(j|x)* is the probability estimation of classifying an instance into class *j*. That is, the classifier will classify an instance *x* into positive class if and only if:

$P(0|x)C(1,0) + P(1|x)C(1,1) \leq P(0|x)C(0,0) + P(1|x)C(0,1)$

This is equivalent to:

$$P(0|x)(C(1,0) - C(0,0)) \le P(1|x)(C(0,1) - C(1,1))$$

Thus, the decision (of classifying an example into positive) will not be changed if a constant is added into a column of the original cost matrix. Thus, the original cost matrix can always be converted to a simpler one by subtracting $C(0,0)$ to the first column, and $C(1,1)$ to the second column. After such conversion, the simpler cost matrix is shown in Table 2. Thus, any given cost-matrix can be converted to one with $C(0,0) = C(1,1) = 0$. [1] In the rest of the paper, we will assume that $C(0,0) = C(1,1) = 0$. Under this assumption, the classifier will classify an instance x into positive class if and only if:

$$P(0|x)C(1,0) \le P(1|x)C(0,1)$$

Table 2. A simpler cost matrix with an equivalent optimal classification.

	True negative	True positive
Predict negative	0	$C(0,1) - C(1,1)$
Predict positive	$C(1,0) - C(0,0)$	0

As $P(0|x)=1 - P(1|x)$, we can obtain a threshold p^* for the classifier to classify an instance x into positive if $P(1|x) \ge p^*$, where

$$p^* = \frac{C(1,0)}{C(1,0)+C(0,1)} = \frac{FP}{FP+FN}. \qquad (2)$$

Thus, if a cost-insensitive classifier can produce a posterior probability estimation $p(1|x)$ for test examples x, we can make it cost-sensitive by simply choosing the classification threshold according to (2), and classify any example to be positive whenever $P(1|x) \ge p^*$. This is what several cost-sensitive meta-learning algorithms, such as *Relabeling*, are based on (see later for details). However, some cost-insensitive classifiers, such as C4.5, may not be able to produce accurate probability estimation; they are designed to predict the class correctly. *Empirical Thresholding* (Sheng and Ling, 2006) does not require accurate estimation of probabilities – an accurate ranking is sufficient. It simply uses cross-validation to search the best probability from the training instances as the threshold.

Traditional cost-insensitive classifiers are designed to predict the class in terms of a default, fixed threshold of 0.5. (Elkan, 2001) shows that we can "rebalance" the original training examples by sampling such that the classifiers with the 0.5 threshold is equivalent to the classifiers with the p* threshold as in (2), in order to achieve cost-sensitivity. The rebalance is done as follows. If we keep all positive examples (as they are assumed as the rare class), then the number of negative examples should be multiplied by $C(1,0)/C(0,1) = FP/FN$. Note that as usually $FP < FN$, the multiple is less than 1. This is thus often called "under-sampling the majority class". This is also equivalent to "proportional sampling", where positive and negative examples are sampled by the ratio of:

$$p(1)\ FN : p(0)\ FP \qquad (3)$$

where $p(1)$ and $p(0)$ are the prior probability of the positive and negative examples in the original training set. That is, the prior probabilities and the costs are interchangeable: doubling $p(1)$ has the same effect as doubling FN, or halving FP (Drummond and Holte, 2000). Most sampling

[1] Here we assume that the misclassification cost is the same for all examples. This property is stronger than the one discussed in (Elkan 2001).

meta-learning methods, such as Costing (Zadrozny et al., 2003), are based on (3) above (see later for details).

Almost all meta-learning approaches are either based on (2) or (3) for the thresholding- and sampling-based meta-learning methods respectively, to be discussed in the next section.

Structure of Learning System

Broadly speaking, cost-sensitive learning can be categorized into two categories. The first one is to design classifiers that are cost-sensitive in themselves. We call them the direct method. Examples of direct cost-sensitive learning are ICET (Turney, 1995) and cost-sensitive decision tree (Drummond and Holte, 2000; Ling et al, 2004). The other category is to design a "wrapper" that converts any existing cost-insensitive (or cost-blind) classifiers into cost-sensitive ones. The wrapper method is also called cost-sensitive meta-learning method, and it can be further categorized into thresholding and sampling. Here is a hierarchy of the cost-sensitive learning and some typical methods. This paper will focus on cost-sensitive meta-learning that considers the misclassification cost only.

Cost-sensitive learning
- Direct methods
  - ICET (Turney, 1995)
  - Cost-senstive decision trees (Drummond and Holte, 2000; Ling et al, 2004)
- Meta-learning
  - Thresholding
    - MetaCost (Domingos, 1999)
    - CostSensitiveClassifier (CSC in short) (Witten & Frank, 2005)
    - Cost-sensitive naïve Bayes (Chai et al., 2004)
    - Empirical Thresholding (ET in short) (Sheng and Ling, 2006)
  - Sampling
    - Costing (Zadrozny et al., 2003)
    - Weighting (Ting, 1998)

Direct Cost-sensitive Learning

The main idea of building a direct cost-sensitive learning algorithm is to directly introduce and utilize misclassification costs into the learning algorithms. There are several works on direct cost-sensitive learning algorithms, such as ICET (Turney, 1995) and cost-sensitive decision trees (Ling et al., 2004).

ICET (Turney, 1995) incorporates misclassification costs in the fitness function of genetic algorithms. On the other hand, cost-sensitive decision tree (Ling et al., 2004), called CSTree here, uses the misclassification costs directly in its tree building process. That is, instead of minimizing entropy in attribute selection as in C4.5, CSTree selects the best attribute by the expected total cost reduction. That is, an attribute is selected as a root of the (sub)tree if it minimizes the total misclassification cost.

Note that as both ICET and CSTree directly take costs into model building, they can also take easily attribute costs (and perhaps other costs) directly into consideration, while meta cost-sensitive learning algorithms generally cannot.

(Drummond and Holte, 2000) investigates the cost-sensitivity of the four commonly used attribute selection criteria of decision tree learning: accuracy, Gini, entropy, and DKM. They claim that the sensitivity of cost is highest with the accuracy, followed by Gini, entropy, and DKM.

Cost-Sensitive Meta-Learning

Cost-sensitive meta-learning converts existing cost-insensitive classifiers into cost-sensitive ones without modifying them. Thus, it can be regarded as a middleware component that pre-processes the training data, or post-processes the output, from the cost-insensitive learning algorithms.

Cost-sensitive meta-learning can be further classified into two main categories: *thresholding* and *sampling*, based on (2) and (3) respectively, as discussed in the Theory section.

Thresholding uses (2) as a threshold to classify examples into positive or negative if the cost-insensitive classifiers can produce probability estimations. *MetaCost* (Domingos, 1999) is a *thresholding* method. It first uses bagging on decision trees to obtain reliable probability estimations of training examples, relabels the classes of training examples according to (2), and then uses the relabeled training instances to build a cost-insensitive classifier. *CSC* (Witten & Frank, 2005) also uses (2) to predict the class of test instances. More specifically, *CSC* uses a cost-insensitive algorithm to obtain the probability estimations $P(j|x)$ of each test instance[2]. Then it uses (2) to predict the class label of the test examples. Cost-sensitive naïve Bayes (Chai et al., 2004) uses (2) to classify test examples based on the posterior probability produced by the naïve Bayes.

As we have seen, all *thresholding*-based meta-learning methods replies on accurate probability estimations of $p(1|x)$ for the test example x. To achieve this, (Zadrozny and Elkan, 2001) propose several methods to improve the calibration of probability estimates. *ET* (Empirical Thresholding) (Sheng and Ling, 2006) is a thresholding-based meta-learning method. It does not require accurate estimation of probabilities – an accurate ranking is sufficient. *ET* simply uses cross-validation to search the best probability from the training instances as the threshold, and uses the searched threshold to predict the class label of test instances.

On the other hand, *sampling* first modifies the class distribution of training data according to (3), and then applies cost-insensitive classifiers on the sampled data directly. There is no need for the classifiers to produce probability estimations, as long as it can classify positive or negative examples accurately. (Zadrozny et al., 2003) show that proportional sampling with replacement produces duplicated cases in the training, which in turn produces overfitting in model building.

[2] CSC is a meta-learning method and can be applied on any classifiers.

However, it is unclear if proper overfitting avoidance (without overlapping between the training and pruning sets) would work well (future work). Instead, (Zadrozny et al., 2003) proposes to use "rejection sampling" to avoid duplication. More specifically, each instance in the original training set is drawn once, and accepted into the sample with the accepting probability $C(j,i)/Z$, where $C(j,i)$ is the misclassification cost of class i, and Z is an arbitrary constant such that $Z \geq max\ C(j,i)$. When $Z = max\ C(j,i)$, this is equivalent to keeping all examples of the rare class, and sampling the majority class without replacement according to $C(1,0)/C(0,1)$ – in accordance with (3). Theorems on performance guarantee have been proved. Bagging is applied after rejection sampling to improve the results further. The resulting method is called *Costing*.

Weighting (Ting, 1998) can also be viewed as a sampling method. It assigns a normalized weight to each instance according to the misclassification costs specified in (3). That is, examples of the rare class (which carries a higher misclassification cost) are assigned proportionally high weights. Examples with high weights can be viewed as example duplication – thus sampling. *Weighting* then induces cost-sensitivity by integrating the instances' weights directly into C4.5, as C4.5 can take example weights directly in the entropy calculation. It works whenever the original cost-insensitive classifiers can accept example weights directly[3]. In addition, *Weighting* does not rely on bagging as *Costing* does, as it "utilizes" all examples in the training set.

Applications

The class imbalanced datasets occurs in many real-world applications where the class distributions of data are highly imbalanced. Again, without loss of generality, we assume that the minority or rare class is the positive class, and the majority class is the negative class. Often the minority class is very small, such as 1% of the dataset. If we apply most traditional (cost-insensitive) classifiers on the dataset, they will likely to predict everything as negative (the majority class). This was often regarded as a problem in learning from highly imbalanced datasets.

However, as pointed out by (Provost, 2000), two fundamental assumptions are often made in the traditional cost-insensitive classifiers. The first is that the goal of the classifiers is to maximize the accuracy (or minimize the error rate); the second is that the class distribution of the training and test datasets is the same. Under these two assumptions, predicting everything as negative for a highly imbalanced dataset *is often the right thing to do*. (Drummond and Holte, 2005) show that it is usually very difficult to outperform this simple classifier in this situation.

Thus, the imbalanced class problem becomes meaningful only if one or both of the two assumptions above are not true; that is, if the cost of different types of error (false positive and false negative in the binary classification) is not the same, or if the class distribution in the test data is different from that of the training data. The first case can be dealt with effectively using methods in cost-sensitive meta-learning.

In the case when the misclassification cost is not equal, it is usually more expensive to misclassify a minority (positive) example into the majority (negative) class, than a majority example into the minority class (otherwise it is more plausible to predict everything as negative).

[3] Thus, we can say that *Weighting* is a semi meta-learning method.

That is, *FN* > *FP*. Thus, given the values of *FN* and *FP*, a variety of cost-sensitive meta-learning methods can be, and have been, used to solve the class imbalance problem (Ling and Li, 1998; Japkowicz and Stephen, 2002). If the values of *FN* and *FP* are not unknown explicitly, *FN* and *FP* can be assigned to be proportional to *p(-):p(+)* (Japkowicz and Stephen, 2002).

In case the class distributions of training and test datasets are different (for example, if the training data is highly imbalanced but the test data is more balanced), an obvious approach is to sample the training data such that its class distribution is the same as the test data (by oversampling the minority class and/or undersampling the majority class) (Provost, 2000).

Note that sometimes the number of examples of the minority class is too small for classifiers to learn adequately. This is the problem of insufficient (small) training data, different from that of the imbalanced datasets.

References and Recommended Reading

Chai, X., Deng, L., Yang, Q., and Ling,C.X.. 2004. Test-Cost Sensitive Naïve Bayesian Classification. *In Proceedings of the Fourth IEEE International Conference on Data Mining*. Brighton, UK : IEEE Computer Society Press.

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

Drummond, C., and Holte, R. 2005. Severe Class Imbalance: Why Better Algorithms Aren't the Answer. *In Proceedings of the Sixteenth European Conference of Machine Learning, LNAI 3720, 539-546.*

Drummond, C., and Holte, R. 2000. Exploiting the cost (in)sensitivity of decision tree splitting criteria. *In Proceedings of the 17th International Conference on Machine Learning*, 239-246.

Elkan, C. 2001. The Foundations of Cost-Sensitive Learning. *In Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence*, 973-978. Seattle, Washington: Morgan Kaufmann.

Japkowicz, N., and Stephen, S., 2002. The Class Imbalance Problem: A Systematic Study. Intelligent Data Analysis, 6(5): 429-450.

Ling, C.X., and Li, C., 1998. Data Mining for Direct Marketing – Specific Problems and Solutions. Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), pages 73 - 79.

Ling, C.X., Yang, Q., Wang, J., and Zhang, S. 2004. Decision Trees with Minimal Costs. *In Proceedings of 2004 International Conference on Machine Learning (ICML'2004)*.

Provost, F. 2000. Machine learning from imbalanced data sets 101. In Proceedings of the AAAI'2000 Workshop on Imbalanced Data.

Sheng, V.S. and Ling, C.X. 2006. Thresholding for Making Classifiers Cost-sensitive. *In Proceedings of the 21st National Conference on Artificial Intelligence, 476-481.* July 16–20, 2006, Boston, Massachusetts.

Ting, K.M. 1998. Inducing Cost-Sensitive Trees via Instance Weighting. *In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery,* 23-26. Springer-Verlag.

Turney, P.D. 1995. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* 2:369-409.

Turney, P.D. 2000. Types of cost in inductive concept learning. *In Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, Stanford University, California.

Witten, I.H., and Frank, E. 2005. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann Publishers.

Zadrozny, B. and Elkan, C. 2001. Learning and Making Decisions When Costs and Probabilities are Both Unknown. *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, 204-213.

Zadrozny, B., Langford, J., and Abe, N. 2003. Cost-sensitive learning by Cost-Proportionate instance Weighting. In *Proceedings of the 3th International Conference on Data Mining*.