Y. Weiss, & L. Bottou, (Eds.), *Advances in neural information processing systems 17* (pp. 905–912). Cambridge, MA: MIT Press.

Meilă, M. (2003). Comparing clusterings by the variation of information. In *Proceedings of Sixteenth Conference on Learning Theory* (pp. 173–187).

Shamir, R., Sharan, R., & Tsur, D. (2004). Cluster graph modification problems. *Discrete Applied Mathematics, 144*, 173–182.

Swamy, C. (2004). Correlation Clustering: Maximizing agreements via semidefinite programming. In *Proceedings of Fifteenth ACM-SIAM Symposium on Discrete Algorithms* (pp. 519–520).

Tan, J. (2007). A Note on the inapproximability of correlation clustering. Technical Report 0704.2092, eprint arXiv, 2007.

Correlation-Based Learning

►Biological Learning: Synaptic Plasticity, Hebb Rule and Spike Timing Dependent Plasticity

Cost

In ►*Markov decision processes*, negative *rewards* are often expressed as *costs*. A reward of $-x$ is expressed as a cost of x. In ►supervised learning, *cost* is used as a synonym for ►loss.

Cross References

►Loss

Cost Function

►Loss Function

Cost-Sensitive Classification

►Cost-Sensitive Learning

Cost-Sensitive Learning

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

Synonyms

Cost-sensitive classification; Learning with different classification costs

Definition

Cost-Sensitive Learning is a type of learning 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 different misclassifications differently. That is, the cost for labeling a positive example as negative can be different from the cost for labeling a negative example as positive. Cost-insensitive learning does not take misclassification costs into consideration.

Motivation and Background

Classification is an important task in inductive learning and machine learning. A classifier, trained from a set of training examples with class labels, can then 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 ►support vector machines. However, most classification algorithms seek 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 have equal cost.

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 (where having cancer is regarded as the positive class, and non-cancer (healthy) as negative), misdiagnosing a cancer patient as healthy (the patient is actually positive but is classified as negative; thus it is also called "false negative") is much more serious (thus expensive) than a false-positive error. The patient could lose his/her life because of a delay in 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 onto a flight than searching an innocent person.

Cost-sensitive learning takes costs, such as the misclassification cost, into consideration. 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 received the most attention in recent years.

Theory

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

Without loss of generality, binary classification is assumed (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, the notation $C(i,j)$ is also used to represent the misclassification cost of classifying an instance from its actual class j into the predicted class i (1 is used 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 assumed 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.

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 distributions (see ▶Class Imbalance Problem) (Japkowicz & 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 $FN = C(0,1)$ is usually larger than that of $FP = C(1,0)$. 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:

$$R(i|x) = \sum_j P(j|x)\, C(j,i), \tag{1}$$

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:

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

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 $C(0,0) = C(1,1) = 0$. (Here it is assumed that the misclassification cost is the same for

Cost-Sensitive Learning. Table 1 An Example of Cost Matrix for Binary Classification

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

Cost-Sensitive Learning. 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

all examples. This property is a special case of the one discussed in Elkan (2001).) In the rest of the paper, it will be assumed 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:

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

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

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

Thus, if a cost-insensitive classifier can produce a posterior probability estimation $p(1|x)$ for each test example x, one can make the classifier 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 return a class label without a probability estimate. *Empirical Thresholding* (Sheng & Ling, 2006) does not require accurate estimation of probabilities – an accurate ranking is sufficient. It simply uses ►cross-validation to search for the best probability value p^* to use as a 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 one 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 all positive examples (as they are assumed as the rare class) are kept, 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:

$$p(1)\,FN : p(0)\,FP \quad (3)$$

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 & Holte, 2000). Most sampling meta-learning methods, such as costing (Zadrozny, Langford, & Abe, 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.They are called the direct method. Examples of direct cost-sensitive learning are ICET (Turney, 1995) and cost-sensitive decision tree (Drummond & Holte, 2000; Ling, Yang, Wang, & Zhang, 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-sensitive decision trees (Drummond & Holte, 2000; Ling et al., 2004)
- Meta-learning
  - Thresholding
    - MetaCost (Domingos, 1999)
    - CostSensitiveClassifier (CSC in short) (Witten & Frank, 2005)
    - Cost-sensitive naïve Bayes (Chai, Deng, Yang, & Ling, 2004)
    - Empirical Thresholding (ET in short) (Sheng & 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) investigate 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 preprocesses 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 $P(j \mid x)$ of each test instance. (CSC is a meta-learning method and can be applied to any classifiers.) 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 seen, all *thresholding*-based meta-learning methods rely on accurate probability estimations of $p(1 \mid x)$ 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 the 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 they 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. 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 $Z = max_{ij}C(j,i)$, 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). 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 oversampling. *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. (Thus, it can be said that *Weighting* is a semi meta-learning method.) In addition, *Weighting* does not rely on bagging as *Costing* does, as it "utilizes" all examples in the training set.

Recommended Reading

Chai, X., Deng, L., Yang, Q., & Ling, C. X. (2004). Test-cost sensitive naïve Bayesian classification. In *Proceedings of the fourth IEEE international conference on data mining*. Brighton: 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, San Diego* (pp. 155–164). New York: ACM.

Drummond, C., & Holte, R. (2000). Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the 17th international conference on machine learning* (pp. 239–246).

Elkan, C. (2001). The foundations of cost-sensitive learning. *In Proceedings of the 17th international joint conference of artificial intelligence* (pp. 973–978). Seattle: Morgan Kaufmann.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5), 429–450.

Ling, C. X., Yang, Q., Wang, J., & Zhang, S. (2004). Decision trees with minimal costs. In*Proceedings of 2004 international conference on machine learning (ICML'2004)*.

Sheng, V. S., & Ling, C. X. (2006). Thresholding for making classifiers cost-sensitive. In *Proceedings of the 21st national conference on artificial intelligence* (pp. 476–481), 16–20 July 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* (pp. 23–26). Heidelberg: Springer.

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 17th international conference on machine learning, Stanford University, California*.

Witten, I. H., & Frank, E. (2005). *Data mining – Practical machine learning tools and techniques with Java implementations*. San Francisco: Morgan Kaufmann.

Zadrozny, B., & 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* (pp. 204–213).

Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive learning by cost-proportionate instance weighting. In *Proceedings of the third International conference on data mining.*

Cost-to-Go Function Approximation

►Value Function Approximation

Covariance Matrix

Xinhua Zhang
Australian National University,
Canberra, Australia

Definition

It is convenient to define a covariance matrix by using multi-variate random variables (*mrv*): $\mathbf{X} = (X_1, \ldots, X_d)^\top$. For univariate random variables X_i and X_j, their covariance is defined as:

$$\mathrm{Cov}(X_i, X_j) = \mathbb{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right],$$

where μ_i is the mean of X_i : $\mu_i = \mathbb{E}[X_i]$. As a special case, when $i = j$, then we get the variance of X_i, $\mathrm{Var}(X_i) = \mathrm{Cov}(X_i, X_i)$. Now in the setting of *mrv*, assuming that each component random variable X_i has finite variance under its marginal distribution, the covariance matrix $\mathrm{Cov}(\mathbf{X}, \mathbf{X})$ can be defined as a d-by-d matrix whose (i,j)-th entry is the covariance:

$$\left(\mathrm{Cov}(\mathbf{X}, \mathbf{X})\right)_{ij} = \mathrm{Cov}(X_i, X_j) = \mathbb{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right].$$

And its inverse is also called *precision matrix*.

Motivation and Background

The covariance between two univariate random variables measures how much they change together, and as a special case, the covariance of a random variable with itself is exactly its variance. It is important to note that covariance is an unnormalized measure of the correlation between the random variables.

As a generalization to multi-variate random variables $\mathbf{X} = (X_1, \ldots, X_d)^\top$, the covariance matrix is a