Thresholding for Making Classifiers Cost-sensitive

Victor S. Sheng, Charles X. Ling

Department of Computer Science
The University of Western Ontario, London, Ontario N6A 5B7, Canada
{ssheng, cling}@csd.uwo.ca

Abstract

In this paper we propose a very simple, yet general and effective method to make any cost-insensitive classifiers (that can produce probability estimates) cost-sensitive. The method, called *Thresholding*, selects a proper threshold from training instances according to the misclassification cost. Similar to other cost-sensitive meta-learning methods, *Thresholding* can convert any existing (and future) cost-insensitive learning algorithms and techniques into cost-sensitive ones. However, by comparing with the existing cost sensitive meta-learning methods and the direct use of the theoretical threshold, *Thresholding* almost always produces the lowest misclassification cost. Experiments also show that *Thresholding* has the least sensitivity on the misclassification cost ratio. Thus, it is recommended to use when the difference on misclassification costs is large.

Introduction

Classification is a primary task of inductive learning in machine learning. Many effective inductive learning techniques have developed, such as naïve Bayes, decision trees, neural networks, and so on. However, most original classification algorithms ignore different misclassification errors; or they implicitly assume that all misclassification errors cost equally. In many real-world applications, this assumption is not true. For example, in medical diagnosis, missing a cancer diagnosis (false negative) is much more serious than the other way around (false positive); the patient could lose his/her life because of the delay in treatment. In many real-world applications, the differences between different misclassification errors can be quite large. Cost-sensitive learning (Turney, 1995, 2000; Elkan, 2001; Zadrozny and Elkan, 2001; Lizotte, 2003; Ting, 1998) has received much attention in recent years to deal with such an issue. Many works for dealing with different misclassification costs have been done, and they can be categorized into two groups. One is to design cost-sensitive learning algorithms directly (Turney, 1995; Drummond and Holte, 2000). The other is to design a wrapper that converts existing cost-insensitive base learning algorithms into cost-sensitive ones. The wrapper method is also called cost-sensitive meta-learning. Section 2 provides a more detailed review of cost-sensitive meta-learning approaches (such as relabeling (Domingos, 1999, Witten & Frank, 2005), sampling (Zadronzny et al., 2003), and weighting (Ting, 1998)).

Cost-sensitive meta-learning methods are useful because they allow us to reuse existing base learning algorithms and their related improvements. *Thresholding* is another cost-sensitive meta-learning method, and it is applicable to any classifiers that can produce probability estimates on training and test examples. Almost all classifiers (such as decision trees, naïve Bayes, and neural networks) can produce probability estimates on examples. *Thresholding* is very simple: it selects the probability that minimizes the total misclassification cost on the training instances as the threshold for predicting testing instances. However, we will show that *Thresholding* is highly effective. It outperforms previous meta-learning cost-sensitive methods, and even the theoretical threshold, on almost all datasets. It is also least sensitive when the difference in misclassification costs is high.

In the next section, we will give an overview of previous work on cost-sensitive meta-learning, particularly MetaCost (Domingos, 1999) and Weighting (Ting, 1998). Section 3 describes *Thresholding* that can convert any cost-insensitive classifiers into cost-sensitive ones. The empirical evaluation is presented in Section 4, which is followed by conclusions in the last section.

Review of Previous Work

Cost-sensitive meta-learning converts existing cost-insensitive base learning algorithms 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, for cost-insensitive learning algorithms.

Cost-sensitive meta-learning techniques can be classified into two main categories, *sampling* and *non-sampling*, in terms of whether the distribution of training data is modified or not according to the misclassification costs. Costing (Zadronzny et al., 2003) belongs to the sampling category. This paper focuses on the non-sampling cost-sensitive meta-learning approaches. The non-sampling approaches can be further classified into three subcategories: relabeling, weighting, and threshold adjusting, described below.

The first is *relabeling* the classes of instances, by applying the minimum expected cost criterion (Michie, Spiegelhalter, and Taylor, 1994). *Relabeling* can be further divided into two branches: relabeling the training instances and relabeling the test instances. MetaCost (Domingos, 1999) belongs to the former, and CostSensitiveClassifier (CSC) (Witten & Frank, 2005) belongs to the latter.

Weighting (Ting, 1998) assigns a certain weight to each instance in terms of its class, according to the misclassification costs, such that the learning algorithm is in favor of the class with high weight/cost.

The third subcategory is *threshold adjusting*. *Thresholding* belongs to this category. It searches for the best probability as a threshold for future prediction. We provide a detailed description of it in Section 3. In Section 4, we compare it with the other non-sampling methods: relabling and weighting.

In (Elkan, 2001), the theoretical threshold for making an optimal decision on classifying instances into positive is obtained as:

$$T = \frac{C(1,0)}{C(1,0) + C(0,1)}, \quad (2)$$

where *C(j,i)* is the misclassification cost of classifying an instance belonging to class *j* into class *i*. In this paper, we assume that there is no cost for the true positive and the true negative, i.e., $C(0,0) = C(1,1) = 0$.

(Elkan, 2001) further discusses how to use this formula to rebalance training instances (e.g., via sampling) to turn cost-insensitive classifiers into cost-sensitive ones. In a later section, we will show *Thresholding*, which searches for the best threshold, surprisingly outperforms the direct use of the theoretical threshold defined in (2).

Thresholding

As we have discussed in Introduction, almost all classification methods can produce probability estimates on instances (both training instances and test instances). *Thresholding* simply finds the best probability from the training instances as the threshold, and use it to predict the class label of test instances: a test example with predicted probability above or equal to this threshold is predicted as positive; otherwise as negative. Thus, for a given threshold, the total misclassification cost for a set of examples can be calculated, and it (M_C) is a function of the threshold (T); that is, $M_C=f(T)$. The curve of this function can be obtained after computing misclassification costs for each possible threshold. In reality, we only need to calculate misclassification costs for each possible probability estimates on the training examples. With this curve, *Thresholding* can simply choose the best threshold that minimizes the total misclassification cost, with the following two improvements on tie breaking and overfitting avoidance.

There are in general three types of curves for the function $M_C=f(T)$, as shown in Figure 1. Figure 1(a) shows a curve of the total misclassification cost with one global minimum. This is the ideal case. However, in practice, there may exist local minima in the curve $M_C=f(T)$ as shown in Figures 1(b) and 1(c). Figure 1(b) shows a case with multiple local minima but one of them is smaller than all others. In both cases ((a) and (b)) it is straightforward for *Thresholding* to select the threshold with the minimal total cost. Figure 1(c) shows a case with two or more local minima with the same value. We have designed a heuristic to resolve the tie: we select the local minimum with hills that are less steep on average; in another word, we select the local minimum whose "valley" has a wider span. The rationale behind this heuristic for the tie breaking is that we prefer a local minimum that is less sensitive to small changes in the threshold selection. For the case shown in Figure 1(c), the span of the right "valley" is greater than the one of the left. Thus, T_2 is chosen as the best threshold.

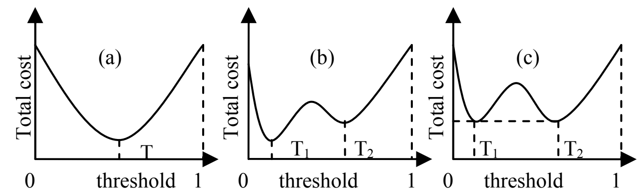

Figure 1. Typical curves for the total misclassification cost.

Another improvement is overfitting avoidance. Overfitting can occur if the threshold is obtained directly from the training instances: the best threshold obtained directly from the training instances may not generalize well for the test instances. To reduce overfitting, *Thresholding* searches for the best probability as threshold from the validation sets. More specifically, an *m-fold* cross-validation is applied, and the base learning algorithm predicts the probability estimates on the validation sets. After this, the probability estimate of each training example is obtained (as it was in the validation set). *Thresholding* then simply picks up the best threshold that yields the minimum total misclassification cost (with the tie breaking heuristic described earlier), and use it for the test instances. Note that the test instances are not used for searching the best threshold.

Empirical Evaluation

Table 1. Twelve Datasets used in the experiments, where Monks-P3 represents the dataset *Monks-Problems-3*.

	No. of Attributes	No. of Instances	Class dist. (*N/P*)	Cost ratio (*FP/FN*)
Breast-cancer	10	286	201/85	85/201
Breast-w	10	699	458/241	241/458
Car	7	1728	1210/518	518/1210
Credit-g	21	1000	700/300	300/700
Diabetes	9	768	500/268	268/500
Hepatitis	20	155	32/123	123/32
Kr-vs-kp	37	3196	1669/1527	1527/1669
Monks-P3	7	554	266/288	288/266
Sick	30	3772	3541/231	231/3541
Spect	23	267	55/212	212/55
Spectf	45	349	95/254	254/95
Tic-tac-toe	10	958	332/626	626/332

To compare *Thresholding* with other existing methods, we choose 11 real-world datasets and 1 artificial dataset (Monks-Problems-3), listed in Table 1, from the UCI Machine Learning Repository (Blake and Merz, 1998). These datasets are chosen because they are binary classes, have at least some discrete attributes, and have a good number of instances. In all experiments, we use *10-fold* cross validation in *Thresholding*.

Comparing with Other Meta-Learning Methods

We choose C4.5 (Quinlan, 1993) as the base learning algorithm. We first conduct experiments to compare the performance of *Thresholding* with existing meta-learning cost-sensitive methods: MetaCost, CSC and Weighting in CostSensitiveClassifier. Many researchers (Bauer and Kohavi, 1999; Domingos, 1999; Buhlmann and Yu, 2003; Zadrozny et al., 2003) have shown that Bagging (Breiman, 1996) can reliably improve base classifiers. As bagging has already been applied in MetaCost, we also apply bagging (with different numbers of bagging iterations) to *Thresholding* and CostSensitiveClassifier.

We implement *Thresholding* in the popular machine learning toolbox WEKA (Witten & Frank, 2005). As MetaCost and CostSensitiveClassifier are already implemented in WEKA, we directly use these implementations in our experiments.

As misclassification costs are not available for the datasets in the UCI Machine Learning Repository, we reasonably assign their values to be roughly the number of instances of the opposite class. This way, the rare class is more expensive if you predict it incorrectly. This is normally the case in the real-world applications. This setting can also reduce the potential effect of the class distribution, because the performance of MetaCost, CSC and Weighting in CostSensitiveClassifier may be affected by the base learners that implicitly make decisions based on the threshold 0.5. Later we will set misclassification costs to be independent of the number of examples.

The experimental results, shown in Figure 3, are presented in terms of the average total cost via 10 runs over ten-fold cross-validation applied to all the methods. This is the external cross-validation for *Thresholding*. Note that *Thresholding* has an internal cross-validation (i.e., the m-fold cross validation described in Section 3), which is only used to search the proper threshold from the training set in *Thresholding*. Figure 2 shows the experiment process for *Thresholding*.

1. Apply 10-fold cross-validation. That is, sample 90% data for training, and the rest (none-overlapping) is for testing
   a. Apply 10-fold cross-validation on the training data to find the proper threshold.

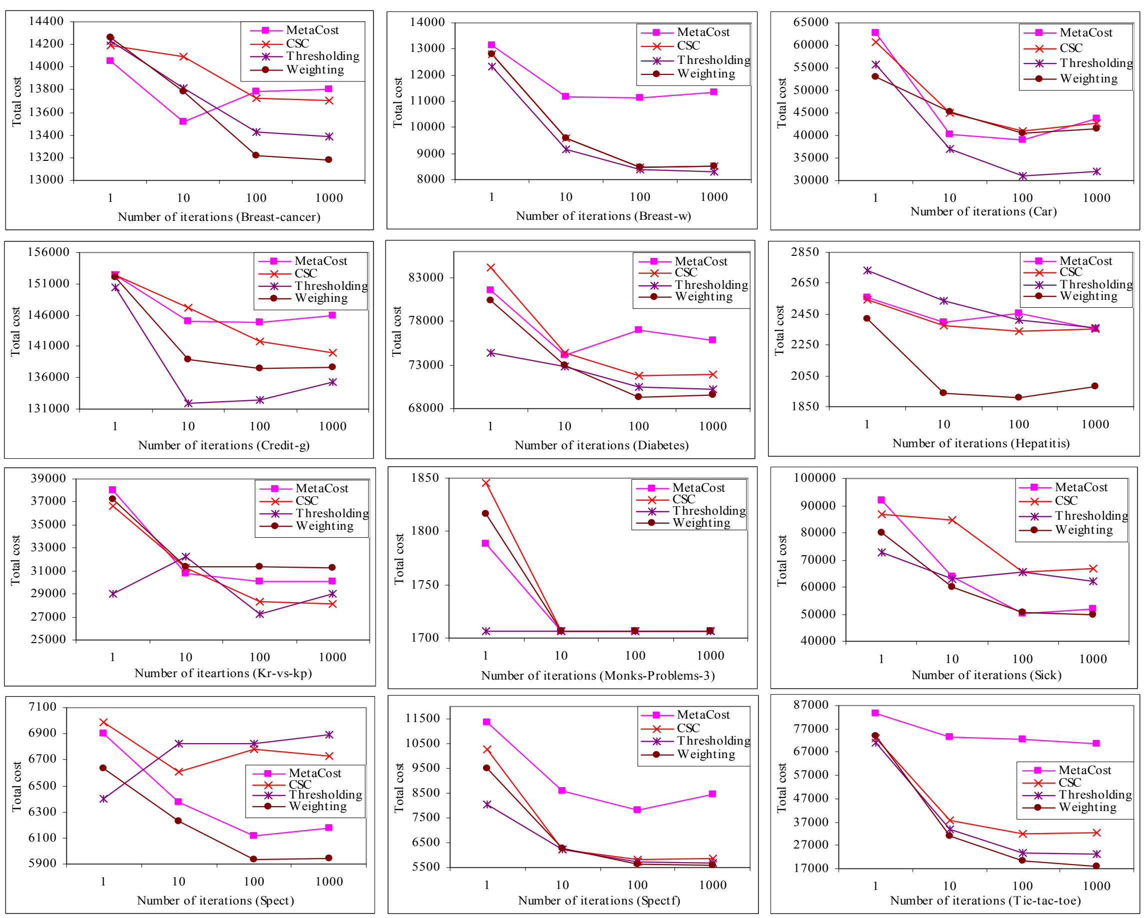

Figure 3. Comparing *Thresholding* with other meta-learning approaches. The lower the total cost, the better.

i. Apply the base learner on the internal training set
ii. Predict probability estimates on the validation set

b. Find the best threshold based on the predicted probabilities
c. Classify the examples in the test set with the threshold obtained in step 1(a)

2. Obtain the average total cost

Figure 2. The experiment process of *Thresholding*.

In Figure 3, the vertical axis represents the total misclassification cost, and the horizontal axis represents the number of iterations in bagging. We summarize the experimental results in Table 2.

Table 2. Summary of the experimental results. An entry *w/t/l* means that the approach at the corresponding row wins in *w* datasets, ties in *t* datasets, and loses in *l* datasets, compared to the approach at the corresponding column[1].

	MetaCost	CSC	Weighting
CSC	7/1/4		
Weighting	9/0/3	10/1/1	
Thresholding	9/1/2	9/1/2	6/1/5

We can draw the following interesting conclusion from the results shown in Figure 3 and Table 2. First of all, MetaCost almost performs worse than other meta-learning algorithms. MetaCost may overfit the data as it uses the same learning algorithm to build the model as the one to relabel the training examples. Bagging does improve its performance in all datasets tested, particularly in first 10 iterations. But the improvements are not as significant as Bagging applied in other algorithms, particularly after 10 iterations. Second, CSC performs better than MetaCost in seven out of twelve datasets. In other datasets, it is similar or worse. Third, overall, Weighting performs much better than MetaCost and CSC. Weighting performs worse than MetaCost only in three datasets (*Car, Kr-vs-kp, and Monks-Problems-3*). In others, it outperforms MetaCost significantly. Comparing with CSC, Weighting performs better in ten out of twelve datasets. In the other datasets, it is the same (*Breast-w*) or worse (*Kr-vs-kp*). Fourth, *Thresholding* outperforms MetaCost and CSC in nine out of twelve datasets respectively, and outperforms Weighting in six datasets. In the others, it is similar or worse. Similar to MetaCost, Bagging does improve the performance of *Thresholding*, but not significant. Without bagging (i.e., the number of iteration is 1), *Thresholding* performs the best in nine out of twelve datasets. In all, we can conclude that *Thresholding* is the best, followed by Weighting, followed by CSC.

Both MetaCost and CSC belong to the relabeling category: the former relabels the training instances and the latter relabels the test instances. This leads us to believe that the relabeling approach is less satisfactory, and *Thresholding* and Weighting seem to be better meta-learning approaches. The experimental results in this section show *Thresholding* is the best.

Sensitivity to Cost Ratios

In the last section, we compare the performance of meta-learning methods under some specific misclassification costs. In this section, we evaluate the sensitivity of these meta-learning methods in terms of different cost ratios of 2:1, 5:1, 10:1, and 20:1 between false positive and false negative. These cost ratios are independent to the number of positive and negative instances.

Bagging (with 10 iterations) is still applied in all methods. The results, shown in Figure 4, are presented in terms of the average total cost (in units; we set the false negative misclassification cost as one unit) over ten-fold cross-validation. The vertical axis represents the total cost, and the horizontal axis represents the cost ratios. We summarize the results in Table 3.

Table 3. Summary of the experimental results (Figure 4). The definition of the entry *w/t/l* is the same as Table 2.

	MetaCost	CSC	Weighting
CSC	2/0/10		
Weighting	5/3/4	7/2/3	
Thresholding	6/3/3	9/2/1	7/2/3

From the results in Figure 4 and Table 3, we can draw the following conclusions. First, the relative relationship for the performance of the meta-learning methods remains the same: *Thresholding* is the best, followed by Weighting. However, MetaCost is much better than CSC. This shows that the post-relabeling (CSC) becomes worse when the cost ratios increase. *Thresholding* outperforms all other methods for most cost ratios in seven out of twelve datasets tested. Second, overall the total misclassification cost increases with increasing values of the cost ratios. This is expected as the sum of false positive and false negative increases when the value of the cost ratio increases.

Another interesting conclusion is that each method has a different sensitivity to the cost ratio increment. The sensitivity can be reflected by how quickly the total misclassification cost increases when the cost ratio increases. The less quickly it increases, the better. We can see from Figure 4 that the increment of the total cost of CSC is always almost the greatest. It is followed by MetaCost. MetaCost is similar as Weighting only in three datasets (*Breast-w, Credit-g*, and *Spect*). However, Weighting outperforms MetaCost in five of the rest datasets. *Thresholding* is again the best (i.e., the slowest increment) in six out of twelve datasets tested. It performs better than MetaCost in six datasets, better than CSC in nine datasets, and better than Weighting in seven datasets. Except two datasets (*Credit-g and Diabetes*),

[1] As there are four points in each curve, we define that curve *A* wins curve *B* if *A* has more than three points, including three points, lower than their corresponding points in *B*. We also define that *A* ties with *B* if *A* has two points lower and the other two points higher than their corresponding points in *B*. For the rest cases, curve A loses to curve B.

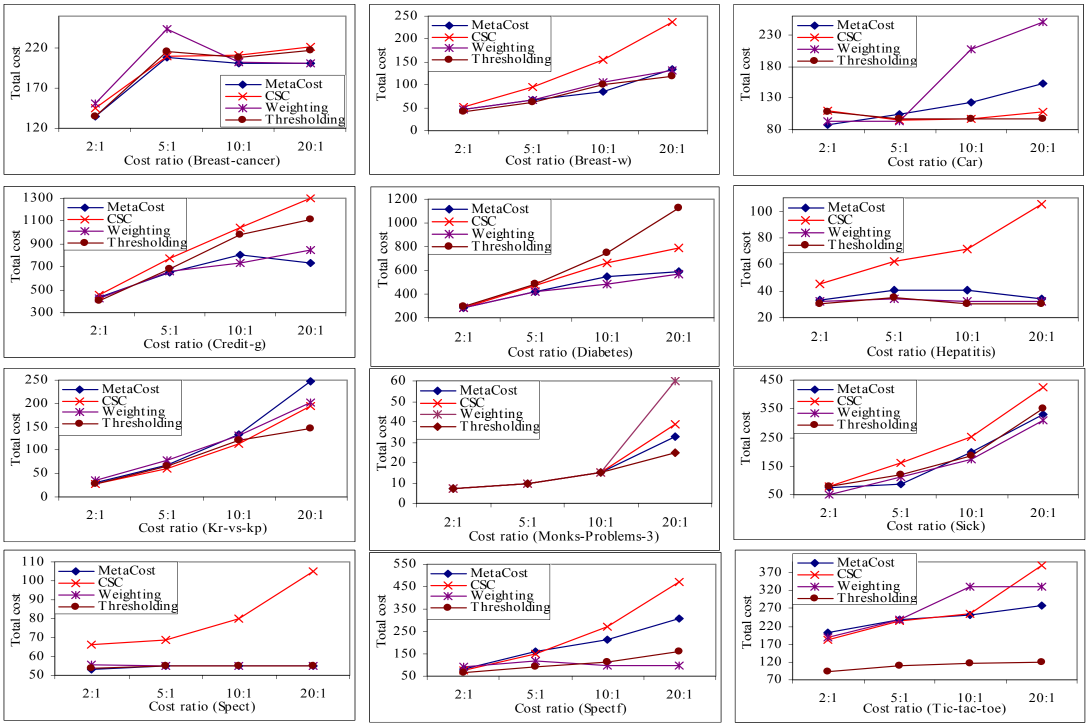

Figure 4. Total cost under different cost ratios.

Thresholding is one of the best methods for the rest of the datasets. In all, we can conclude that CSC is most sensitive to the increment of the cost ratios, followed by MetaCost, and followed by Weighting. Thresholding is the most resistant (the best) to the cost ratios. Thus, when the cost ratio is large, it is recommended over other methods.

Theoretical Threshold

Thresholding searches for the best threshold from the training instances; however, it is time consuming to search for the best threshold via cross-validation. How does it compare with the theoretical threshold reviewed earlier? In this section, we compare *Thresholding* with the direct use of the theoretical threshold. We conduct the same experiments as the last subsection. The results are presented in Figure 5. We can see that *Threholding* clearly outperforms the theoretical threshold in nine out of twelve datasets. They are exact same in the dataset *Monks-Problems-3*. In addition, *Thresholding* has a better resistance (insensitivity) to large cost ratios, particularly in datasets *Breast-w*, *Car, Credit-g*, *Hepatitis*, *Kr-vs-kp, Spectf*, *Spect*, and *Tic-tac-toe*. We can thus conclude that it is worth spending time to search for the best threshold in *Thresholding*.

Table 4. Comparing Theoretical with other approaches.

	MetaCost	CSC	Weighting
Theoretical	6/0/6	9/2/1	5/0/7

We summarize the comparisons between Figure 4 and Figure 5 in Table 4. We can see that the use of the theoretical threshold performs much better than CSC, although it ties to MetaCost and worse than Weighting.

Conclusions and Future Work

Thresholding is a general method to make any cost-insensitive learning algorithms cost-sensitive. It is a simple yet direct approach as it learns the best threshold from the training instances. Thus, the best threshold chosen reflects not only different misclassification costs but also the data distribution. We were surprised by its good performance. However our repeated experiments show that *Thresholding* outperforms other existing cost-sensitive meta-learning methods, such as MetaCost, CSC, Weighting, and the direct use of the theoretical threshold. *Threholding* also has the best resistance (insensitivity) to large misclassification cost ratios. Thus, it is recommended to use especially when the difference in misclassification costs is large.

In our future work, we plan to apply *Thresholding* on datasets with multiple classes.

Acknowledgements

We thank anonymous reviewers for the useful comments and suggestions. The authors thank NSERC for the support of their research.

References

Bauer, E., and Kohavi, R. 1999. An empirical comparison of voting classification algorithms: bagging, boosting and variants, *Machine Learning*, 36(1/2):105-139.

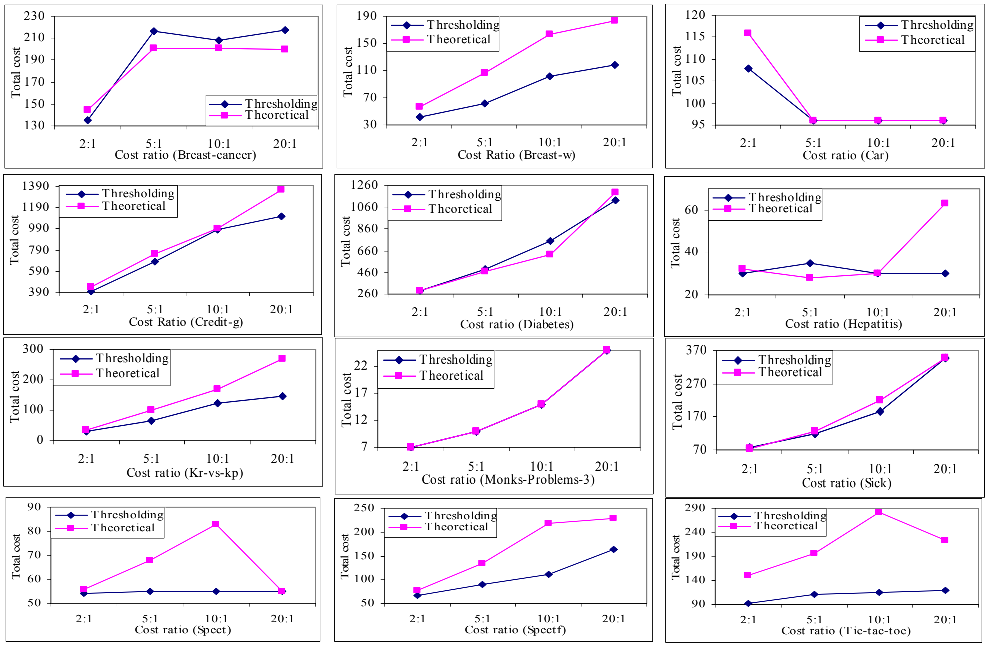

Figure 5. Comparing *Thresholding* with the theoretical threshold.

Blake, C.L., and Merz, C.J. 1998. *UCI Repository of machine learning databases (website)*. Irvine, CA: University of California, Department of Information and Computer Science.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. 1984. Classification and regression trees. Wadsworth, Belmont, CA.

Brieman, L. 1996. Bagging predictors. *Machine Learning*, 24:123-140.

Buhlmann, P., and Yu, B. 2003. Analyzing bagging. *Annals of Statistics*.

Chawla, N.V., Japkowicz, N., and Kolcz, A. eds. 2004. *Special Issue on Learning from Imbalanced Datasets. SIGKDD*, 6(1): ACM 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. 2000. Exploiting the cost (in)sensitivity of decision tree splitting criteria. *In Proceedings of the 17th International Conference on Machine Learning*, 239-246.

Drummond, C., and Holte, R.C. 2003. C4.5, Class Imbalance, and Cost Sensitivity: Why under-sampling beats over-sampling. *Workshop on Learning from Imbalanced Datasets II.*

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.

Lizotte, D., Madani, O., and Greiner R. 2003. Budgeted Learning of Naïve-Bayes Classifiers. *In Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. Acapulco, Mexico: Morgan Kaufmann.

Michie, D., Spiegelhalter, D.J., and Taylor, C.C. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited.

Quinlan, J.R. eds. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

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.

Weiss, G., and Provost, F. 2003. Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research* 19: 315-354.

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

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.*

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.