An Empirical Study of the Noise Impact on Cost-Sensitive Learning *

Xingquan Zhu[1, 3], Xindong Wu[2], Taghi M. Khoshgoftaar[1], Yong Shi[3]
[1]Dept. of Computer Science & Eng., Florida Atlantic University, Boca Raton, FL 33431, USA
[2]Department of Computer Science, University of Vermont, Burlington, VT 05405, USA
[3]Graduate University, Chinese Academy of Sciences, Beijing, 100080, China
{xqzhu, taghi}@cse.fau.edu; xwu@cems.uvm.edu; yshi@gucas.ac.cn

Abstract

In this paper, we perform an empirical study of the impact of noise on cost-sensitive (CS) learning, through observations on how a CS learner reacts to the mislabeled training examples in terms of misclassification cost and classification accuracy. Our empirical results and theoretical analysis indicate that mislabeled training examples can raise serious concerns for cost-sensitive classification, especially when misclassifying some classes becomes extremely expensive. Compared to general inductive learning, the problem of noise handling and data cleansing is more crucial, and should be carefully investigated to ensure the success of CS learning.

1 Introduction

Recently, a body of work has been attempted to address cost-related inductive learning issues, with techniques known as cost-sensitive learning (Tan 93, Turney 95, Pazzani et al. 94), where the "cost" could be interpreted as misclassification cost, training cost, and test cost (Turney). Among all different types of costs, the misclassification cost is the most popular one. In general, misclassification cost is described by a cost matrix *C*, with *C*(*i*, *j*) indicating the cost of predicting that an example belongs to class *i* when in fact it belongs to class *j*. With this type of cost, the objective of a *CS* learner is to form a generalization such that the average cost on previously unobserved instances is minimized. Obviously, this minimal cost is determined by two most important factors: (1) the inductive bias of the underlying *CS* learner; and (2) the quality of the training data. Existing research has made significant progress in exploring efficient *CS* learning algorithms (Bradford et al. 98, Zuberk & Dietterich 02, Tan 93, Geibel & Wysotzki 03, Brefeld et. al. 03, Domingos 99, Chan & Stolfo 98, Zadrozny 03, Abe & Zadrozny 04), with assumptions that the input data are noise-free or noise in the datasets is not significant. In data-driven application domains, many potential problems, such as unreliable data acquisition sources, faulty sensors, and data collection errors, will make data vulnerable to errors. It is therefore important to understand how a CS learning algorithm behaves in noisy environments and how to handle data errors in supporting effective CS learning. In this paper, we report our empirical study on the impact of noise on *CS* learning. It is hoped that our observations will be beneficial to any real-world applications with cost concerns.

2 Related Work

The problem of learning from noisy data has been a focus of much attention in data mining (Quinlan 86b), and most algorithms have a mechanism to handle noise in the training data. For example, pruning on a decision tree is designed to reduce the chance that the tree is overfitting to noise in the training data (Quinlan 86). In real-world applications, many learning algorithms rely on a data cleaning model for data enhancement (Brodley & Friedl 99). Although the problem of error handling in supervised learning has been well studied, research in the area has been mainly focused on general inductive learning for the minimization of zero-one loss or error rate. In reality, many applications are not only characterized by error rate, but also by various types of costs (Turney), such as misclassification cost and test cost.

Among all different types of costs, the misclassification cost is the most popular one. Assume that examples of a dataset are drawn independently from a distribution, *d,* with domain $X \times Y \times C$, where *X* is the input space to a classifier, *Y* is an output space, and $C \subset [0, \infty)$ is the importance (misclassification cost) associated with misclassifying that example. The goal of cost-sensitive (*CS*) learning, from the misclassification cost perspective, is to learn a classifier $h: X \rightarrow Y$ which minimizes the expected cost (Zadrozny 03)

$$E_{d(x,y,c)}[c \cdot I(h(x) \neq y)] \qquad (1)$$

Two important issues determine the minimization of the expected cost: the total number of misclassifications and the cost of each single misclassification. To minimize overall costs, a compromise is often made by sacrificing cheap examples and enhancing the accuracy on classes containing expensive instances. This is distinct from general inductive learning, because the latter is biased towards larger classes (likely the cheap examples in a *CS* scenario). To enhance a CS learner trained from noisy data environments, Zhu and Wu (04) have proposed a Classification Filter for data cleansing. But the study of *CS* learners in noisy environments still lacks in-depth empirical and theoretical analysis.

* The support of the National Science Foundation of China under Grant No.60674109 is acknowledged.

3 Experiment Setting

An empirical study needs a full control over noise levels in the data, so that we can observe the behaviors of the underlying CS learners. For this purpose, we implement two manual corruption mechanisms, *Total Random Corruption* (*TRC*) and *Proportional Random Corruption* (*PRC*). With *TRC*, when the users specify an intended corruption level $x \cdot 100\%$, we will randomly introduce noise to all classes. I.e., an instance with its label i has an $x \cdot 100\%$ chance to be mislabeled as another random class (excluding class i). *TRC*, however, changes the class distribution in the dataset after noise corruption, which raises a big concern in *CS* Learning, because modifying the class distribution may change the average cost considerably. We then propose *PRC*, which keeps the class distribution constant during the noise corruption. Given a dataset with Y classes, assume the original class distribution is $\pi_1, \pi_2,..,\pi_Y$, with $\sum_i^Y \pi_i = 1$, where π_1 and π_Y are the percentage of the most and least common classes respectively. For any user specified corruption level, say $x \cdot 100\%$, we proportionally introduce random noise to different classes with instances in class i having a $(\pi_Y / \pi_i) \cdot x \cdot 100\%$ chance of being corrupted as another random class. That is, we use the least common class as the baseline, and proportionally introduce noise to different classes. It is obvious that the actual noise level in *PRC* is less (or even much less) than the intended corruption level.

For each experiment, we perform 10 times 3-fold cross-validation. In each run, the dataset is divided into a training set E and a test set T. We introduce a certain level of noise to E and generate a noisy dataset E'. Meanwhile, assuming a noise cleansing technique is able to remove all noisy instances from E', we can therefore build a cleansed dataset E''. We observe *CS* learners trained from E, E', and E'' to study noise impact (assessed by using T as the test set). The empirical study is made by the observation on three benchmark two-class datasets, Credit Screen Dataset (Credit), Wisconsin Breast Cancer Dataset (WDBC), and Thyroid disease dataset (Sick) (Blake & Merz 98), with their class distributions varying from almost even to extremely biased (as shown in Table 1). We use two-class datasets as our test bed, because a two-class problem can explicitly reveal the impacts of noise and cost-ratio on CS learning, whereas for multiple-class problems, issues such as costs among classes, class distributions, and errors in each class often make it difficult to draw comprehensive conclusions. We use the C5.0 CS classification tree (Quinlan 97) in all experiments.

To assign misclassification cost matrix values, $C(i, j)$, $i \neq j$, we adopt a Proportional Cost (*PC*) mechanism: for any two classes i and j ($i \neq j$), we first check their class distribution π_i and π_j. If $\pi_i \geq \pi_j$, which means that class j is relatively rarer than i, then $C(j, i)=100$ and $C(i, j)$ equals to $100 \cdot r$; otherwise $C(i, j)=100$ and $C(j, i)= 100 \cdot r$, where r is the cost-ratio, as defined by Eq. (2).

$$r = C(i, j) \, / \, C(j, i) \qquad (2)$$

Table 1. Class distributions of the benchmark datasets

Dataset	***Class Distribution***	***Separability(c4.5)***
Credit Screening (Credit)	0.56:0.44	82.8%
Wisconsin Diagnostic Breast Cancer (WDBC)	0.63:0.37	93.3%
Thyroid disease (Sick)	0.94:0.06	98.7%

Table 2. Major symbols used in the paper

Symbol	***Description***
E, T, E', E''	E: Training set, T: Test set, E': Noise corrupted training set, E'': Noise cleansed training set
CS_X	The average misclassification cost from dataset X
Ac_Cs_X Ac_Nm_X	The average classification accuracy of a CS and a normal classifier on dataset X respectively
$h(t)$ vs $H(t)$	A CS classifier vs a non-CS classifier
x, y, c	x: an input instance, y: class label of x, c: the cost associated to mislabeling x
r	The cost ratio between the minor class and the major class
κ	The number of instances in the test set T
$C(i, j)$	The cost of predicting that an example belongs to class i when in fact it belongs to class j.
α_1, α_2	The distribution of the major class vs the minor class
$\varepsilon, \varepsilon_1, \varepsilon_2$	The classification error rate on the whole test set, the major class examples and the minor class examples respectively
CS_{min}	The cost of classifying a major class example into the minor class
CS_{Avg}	The average misclassification cost on the test set T
CS_{Upper_bound}	The upper bound of a CS classifier
$d(x,y,c)$	The distribution of the noisy training set E'
$d'(x,y,c)$	The distribution of a new set constructed by sampling E'
$E_{d(x,y,c)}[c \cdot I(h(x) \neq y)]$	The expected average misclassification cost of the instances drawn from the distribution d

4 Noise Impact

4.1 Misclassification Costs

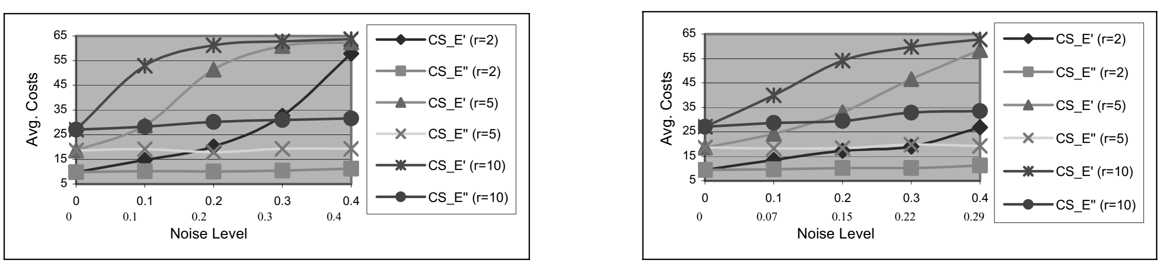
In Figs. 1 to 2, we report the noise impact on the cost of *CS* learners trained from the benchmark datasets, where Figs. 1(a) and 1(b), and Figs. 2(a) and 2(b) represent the results from different noise corruption models of the WDBC and Sick dataset respectively. Fig. 2(c) reports the results from the Credit dataset (because the class distributions of the Credit dataset are almost even, we only report its results on the *TRC* model). In all figures, the first and the second rows of the x-axis represent the intended and actual noise levels in the dataset. The y-axis indicates the average cost of *CS* classifiers. CS_E' and CS_E'' represent the average cost of a *CS* classifier trained from E' and E'' (before and after noise cleaning).

As we can see from Figs. 1 to 2, for both noise corruption models, the average cost of a *CS* classifier proportionally increases with the value of r, even if the dataset is noise-free. This does not surprise us, because we fix the cost of $C(i, j)$ to 100 and set the cost of $C(j, i)$ to $100 \cdot r$. So raising the value of r will surely increase the average cost. When noise is introduced to the dataset, the average cost will inevitably increase, regardless of the noise corruption model and the value of r. On the other hand, removing noisy instances can keep average costs almost the same as that of the noise-free dataset (*E*). This indicates that data cleansing is an effective way to reduce the impact of noise on CS learning.

Although noise increases the cost of a *CS* classifier regardless of the cost-ratio (*r*), when comparing the results from three *r* values, we can find that their patterns of increase are different. As shown in Fig. 1(a), Fig. 2(a), and Fig. 2(c), when *r* is small, increasing noise levels will gradually raise the average cost, and the impacts of noise are less significant for low noise level data. On the other hand, for large *r* values, introducing a small portion of noise will elevate the average cost greatly, and increasing the noise level further does not show significant extra impacts. However, *TRC* in Fig. 1(a), Fig. 2(a), and Fig. 2(c) have already changed the class distribution of the dataset, it is not clear that given the same amount of noise in the dataset which one is responsible for the increase of the cost: the change of the class distribution or the large cost-ratio *r*. We then turn to Fig. 1(b) and Fig. 2(b) for answers, where the class distribution is constant during noise corruption.

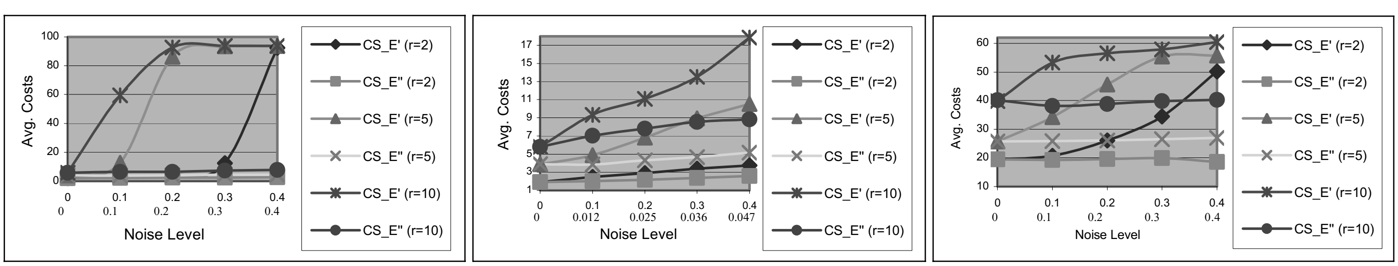
As shown in Figs. 1(b) and 2(b), depending on the bias of the class distribution of the dataset, the actual noise level in the dataset is lower (or much lower) than the intended corruption level. For example, in Fig. 2(b), when the intended noise level is 0.1, the actual noise level in the database is just about 0.012 (please refer to Section 3 and Tab. 1 for the reason). At this noise level, the cost increase for *r*=2 is 0.56 (from 1.91 to 2.47), but for *r*=10 the increase is about 3.5 (from 5.8 to 9.31). One may argue that this increase may be incurred by increasing the class-ratio *r,* because raising *r* will make the minority class more expensive. Nevertheless, even if we assume that the classification accuracy remains the same, we can still see that the average increase from *r*=10 (3.5/10) is significantly larger than the increase from *r*=2 (0.56/2). It is obvious that given the same amount of noise in the dataset, even if noise does not change the class distribution, a dataset with a large cost-ratio tends to be more error prone and therefore may receive a large misclassification cost. These observations indicate that when the dataset has a large cost-ratio value, the existence of noise becomes fatal, and a small portion of noise can corrupt the results significantly. Because of its sensitivity to minor data errors, a dataset with a large cost-ratio will experience more difficulties in data cleansing, if the overall goal is to minimize the misclassification cost.

Another interesting finding in Figs. 1 to 2 is that when applying *CS* learners to a noisy dataset, it seems that the trained *CS* classifiers have a *saturation point* in reacting to the actual noise in the dataset, regardless of whether errors change the class distribution or not. When the noise level is below this point, it will continuously impact the *CS* classifier. But once the noise level is beyond this point, the classifier becomes insensitive to noise. The following analysis will reveal that this *saturation point* is determined by two most important factors: the class distribution and the cost-ratio *r*. Given a two-class dataset, assuming its class distribution is α_1: α_2, with α_1 and α_2 denoting the distribution of the major class and the minor class respectively. The cost of predicting a minor class instance as the major class is $CS_{min} \cdot r$, with CS_{min} indicating the opposite cost (from the major class to the minor class). Assuming that we have built a CS classifier from the dataset and, for a test set *T* with κ examples, the classification error rates of this classifier on the whole test set and the major class are ε and ε_1 respectively. The error rate for the minor class ε_2 is the number of incorrectly classified minor class examples



(a) WDBC dataset, total random noise (*TRC*)   (b) WDBC dataset, proportional random noise (*PRC*)

Fig. 1. Impacts of class noise on the average cost of a *CS* classification algorithm



(a) Sick dataset, total random noise (*TRC*)   (b) Sick dataset, proportional random noise (*PRC*)   (c) Credit dataset, total random noise (*TRC*)

Fig. 2. Impacts of class noise on the average cost of a *CS* classification algorithm

divided by the total number of minor class instances, as shown in Eq. (3). Then, the average misclassification cost in T, $CS_{Avg,}$ is denoted by Eq. (4).

$$\varepsilon_2 = (\varepsilon\cdot\kappa - \frac{\kappa\cdot\alpha_1\cdot\varepsilon_1}{\alpha_1+\alpha_2}) \Big/ (\frac{\kappa\cdot\alpha_2}{\alpha_1+\alpha_1}) = \frac{(\alpha_1+\alpha_2)\cdot\varepsilon - \alpha_1\cdot\varepsilon_1}{\alpha_2} \quad (3)$$

$$\begin{aligned} CS_{Avg} &= (\frac{\varepsilon_1\cdot\alpha_1\cdot\kappa}{\alpha_1+\alpha_2} CS_{min} + \frac{\varepsilon_2\cdot\alpha_2\cdot\kappa}{\alpha_1+\alpha_2} r\cdot CS_{min}) \Big/ \kappa \\ &= \frac{\varepsilon_1\cdot\alpha_1\cdot CS_{\min} + (\varepsilon\cdot\alpha_1+\varepsilon\cdot\alpha_2-\varepsilon_1\cdot\alpha_1)\cdot r\cdot CS_{\min}}{\alpha_1+\alpha_2} \end{aligned} \quad (4)$$

When the average error rate ε increases, it normally raises the average misclassification cost CS_{Avg}, with its value bounded by an upper bound given by Eq. (5).

$$CS_{Upper\_bound} = \begin{cases} \frac{\alpha_1}{\alpha_1+\alpha_2}\cdot CS_{\min} & if \quad r \geq \frac{\alpha_1}{\alpha_2} \\ \frac{\alpha_2}{\alpha_1+\alpha_2}\cdot r\cdot CS_{\min} & Otherwise \end{cases} \quad (5)$$

Actually, the upper bound in Eq. (5) is the average cost of predicting all instances belonging to the minor class when $r \geq \alpha_1/\alpha_2$, or predicting all instances belonging to the major class when $r < \alpha_1/\alpha_2$ (which is the optimal solution for a *CS* classifier with low confidence, because any cost higher than this upper bound is not necessary). It is understandable that $CS_{Avg} \leq CS_{Upper\_bound}$ holds most of the time, which eventually produces the inequality denoted by Eq. (6). Since

$$CS_{Avg} \leq CS_{Upperbound} \Rightarrow \begin{cases} \varepsilon_1\cdot\alpha_1 + (\varepsilon\cdot\alpha_1+\varepsilon\cdot\alpha_2-\varepsilon_1\cdot\alpha_1)\cdot r \leq \alpha_1 & if \quad r\geq\alpha_1/\alpha_2 \\ \varepsilon_1\cdot\alpha_1 + (\varepsilon\cdot\alpha_1+\varepsilon\cdot\alpha_2-\varepsilon_1\cdot\alpha_1)\cdot r \leq r\cdot\alpha_2 & Otherwise \end{cases}$$

so

$$\varepsilon \leq \begin{cases} \frac{(1-\varepsilon_1)\cdot\alpha_1 + \varepsilon_1\cdot\alpha_1\cdot r}{r\cdot(\alpha_1+\alpha_2)} & if \quad r \geq \alpha_1/\alpha_2 \\ \frac{(\alpha_2+\varepsilon_1\cdot\alpha_1)\cdot r - \varepsilon_1\cdot\alpha_1}{r\cdot(\alpha_1+\alpha_2)} & Otherwise \end{cases} \quad (6)$$

For two class problems $\alpha_1+\alpha_2=1$, so Eq. (6) becomes

$$\varepsilon \quad \leq \quad \varepsilon^* = \begin{cases} \varepsilon_1\cdot\alpha_1 + \{(1-\varepsilon_1)\cdot\alpha_1\}/r & if \quad r\geq\alpha_1/\alpha_2 \\ \alpha_2 + (1-1/r)\cdot\varepsilon_1\cdot\alpha_1 \approx \alpha_2 & Otherwise \end{cases} \quad (7)$$

With the inequality in (7), we can finally analyze the relationship between the class distribution α_1, α_2, the cost ratio r, and the error rate ε, and reveal the following facts:

1. Given a test set T, the average misclassification error rate ε of a *CS* classifier, which is trained from a training set equivalent to T, is bounded by a critical value ε^* defined by Eq. (7). When $r \geq \alpha_1/\alpha_2$, the higher the value of the cost-ratio r, the lower the ε^*, and the more likely the classifier tends to approach the *saturation point*.
2. In theory, when $r < \alpha_1/\alpha_2$, ε^* becomes $\alpha_2+(1-1/r)\varepsilon_1\alpha_1$, which can be approximated as α_2. This means that the larger the number of minor class instances (a larger α_2), the higher the ε^*, and the less likely the classifier tends to approach the *saturation point*. However, this conclusion crucially relies on the fact that the underlying *CS* learner is "optimal" and indeed knows that sticking to the major class will lead to the lowest cost, which is not always the case in reality (as we will analyze next).

From Section 3 and Tab. 1, we know that α_1: α_2=0.627:0.373 ($r > \alpha_1/\alpha_2$ for any r=2, 5, 10) and CS_{min}=100 for the WDBC dataset. So according to Eq. (5) CS_{Upper_bound} is 62.7, which is consistent with the results in Fig. 1(a). Now the interesting results come from the results in Fig. 2(a), which also clearly poses a *saturation point* of the misclassification cost. According to Table 1 and Eq. (5), we know that for the Sick dataset α_1: α_2=0.94:0.06, so the ratio between α_1/α_2 is far larger than the value of r. As a result, the CS_{Upper_bound} should be $\alpha_2\cdot r\cdot CS_{\min}/(\alpha_1+\alpha_2)=6\cdot r$ (corresponding to the second item in Eq. (5)). Unfortunately, the results in Fig. 2 (a) indicate that the real CS_{Upper_bound} we got is about 94, which actually equals the value of having all instances classified into the minor class (the first item in Eq. (5)). In reality, it is understandable that the underlying *CS* learners are reluctant to stick to the major class for minimal misclassification cost, especially when noise has changed the *apriori* class distributions significantly. Instead, the CS learners will tend to stick to the minor classes in high uncertainty environments. As a result, the results from this seriously biased dataset are consistent with the conclusion drawn from the first item in Eq. (5).

Based on the above observations, we know that given datasets with the same noise level but a different cost-ratio value r, the larger the cost ratio r, the smaller the value of ε^*, then the dataset is more likely to approach to a *saturation point*, where the misclassification cost appears to reach the maximum. Meanwhile, class noise does not continuously make an impact on the CS classifier all the time, and after noise reaches a certain level, the CS classifier may simply stick to the minor class. As a result, the impact of the noise in the system becomes constant. Given a dataset with $r \geq \alpha_1/\alpha_2$ (which is pretty common in reality), the higher the cost-ratio, the more likely the classifier tends to do so. For situations $r < \alpha_1/\alpha_2$, the conclusions will also likely hold.

4.2 Classification Accuracies

The above observations conclude that a dataset with a higher cost-ratio is sensitive to data errors, and learners built from such data are most likely cost intensive. This conclusion does not, however, solve the concerns like: (1) why is a learner built from noisy data with large cost-ratio cost intensive? and (2) given any noisy dataset *D,* if users were able to build both a normal classifier or a *CS* classifier, which one of them is more trustworthy in identifying noisy examples? Now, let's refer to the theoretical proof and empirical comparisons of the classification accuracies between a normal classifier and a *CS* classifier for answers.

The following analyses are based on an existing ***Folk Theorem**** (Zadrozny *et. al.* 03), which states that if we have examples drawn from the distribution:

* It was called "folk theorem" in the literature, because the authors concluded that the result appears to be known, although it has not been published yet.

$$d'(x,y,c) \equiv \frac{c}{E_{d(x,y,c)}[c]} d(x,y,c) \qquad (8)$$

where $d(x,y,c)$ represents the distribution of the original dataset *E,* then the optimal error rate classifiers built for newly constructed distribution d' are the optimal cost minimizers for data drawn from *d*. This theorem states that if we sample from the original distribution, *d,* of dataset *E* and constructing another distribution d', the efforts which try to learn an optimal error rate classifier on d' actually leads to a classifier which optimizes the misclassification cost for *d*.

THEOREM 1: For a dataset *E* with a distribution *d*, assuming we are able to use an optimal learning theory to build a *CS* ($h(t)$), and a normal classifier ($H(t)$) respectively. Then *h(t)* is inferior to *H(t)* in terms of the average classification accuracy. That is, $\mathrm{Err}_d(H) \le \mathrm{Err}_d(h)$, where $\mathrm{Err}_X(\cdot)$ represents the classification error rate of the classifier on *X*.

Proof:

Assume a *CS* classifier $h(t)$ was built from *d*. Then with the CS learning formula in Eq. (1), the expected cost of $h(t)$ on *d* can be denoted by Eq. (9).

$$E_{d(x,y,c)}[c \cdot I(h(x) \neq y))] = \sum_{x,y,c} d(x,y,c) \cdot c \cdot I(h(x) \neq y) \qquad (9)$$

With the folk theorem in Eq. (8), we know that $d'(x,y,c) = \frac{c}{E_{d(x,y,c)}[c]} d(x,y,c)$, where d' is the sampled distribution from *d*. Then Eq. (9) becomes.

$$\begin{aligned} E_{d(x,y,c)}[c \cdot I(h(x) \neq y))] &= E_{d(x,y,c)}[c] \cdot \sum_{x,y,c} d'(x,y,c) I(h(x) \neq y) \qquad (10) \\ &= E_{d(x,y,c)}[c] \cdot E_{d'(x,y,c)}[I(h(x) \neq y)] \end{aligned}$$

We can transform Eq. (10) as follows.

$$\begin{aligned} E_{d'(x,y,c)}[I(h(x) \neq y)] &= \frac{1}{E_{d(x,y,c)}[c]} E_{d(x,y,c)}[c \cdot I(h(x) \neq y)] \qquad (11) \\ &\le \frac{c}{E_{d(x,y,c)}[c]} E_{d(x,y,c)}[I(h(x) \neq y)] \end{aligned}$$

Because $E_{d(x,y,c)}[c] = \arg\max_c \{(x,y,c) \in d\}$, Eq. (11) becomes

$$E_{d'(x,y,c)}[I(h(x) \neq y)] \quad \le \quad E_{d(x,y,c)}[I(h(x) \neq y)] \qquad (12)$$

That is $$Err_{d'}(h(t)) \quad \le \quad Err_d(h(t)) \qquad (13)$$

Because $h(t)$ was built on a biased distribution d' sampled from *d*, therefore $Err_d(H(t)) \le Err_{d'}(h(t))$; otherwise, there is no need for $H(t)$ to exist (we can directly use $h(t)$ on *d* for normal classification). Therefore, we have Eq. (14) and the statement of Theorem 1 is true.

$$Err_d(H(t)) \quad \le \quad Err_d(h(t)) \qquad (14)$$

Now let's turn to the experimental results in Fig. 4 to evaluate whether the empirical results indeed support our theoretical analysis (the meaning of each curve in Fig. 4 is denoted in Fig. 3). In Fig. 4, *Ac_Cs_E′* and *Ac_Nm_E′* indicate the classification accuracies of a normal and a *CS* classifier trained from *E′* respectively. As shown in Fig. 4, when the cost-ratio is small (*r*=2), the differences between the accuracies of the normal and *CS* classifiers are trivial, with the accuracy of the *CS* classifier slightly worse. This is understandable, because a *CS* classifier likely sacrifices the accuracy for minimal costs. When the cost-ratio gets larger (*r*=5 or higher), the accuracy of the *CS* classifier becomes significantly worse than the normal classifier. If we compare Fig. 4(a) with Fig. 1(a) and Fig. 4(c) with Fig. 2(a), we can find that this is actually the reason a dataset with a large cost-ratio is more willing to approach to the *saturation point*. In this situation, a small portion of noise will make the *CS* classifier ignore the classification accuracy and stick to the "optimal" class.

Theorem 1 together with the results in Fig. 4 clearly indicates that in general situations, a *CS* classifier is inferior to a normal classifier, in terms of classification accuracy. Because removing mislabeled training examples was shown to lead to a better CS learner (as shown in Figs. 1 to 2), class noise handing for effective CS learning will crucially depend on accurate noise identification mechanisms. As a result, in noisy environments, if we want to adopt a noise identification mechanism for data cleansing, we shall trust a normal classifier rather than a CS classifier, because the former always has a higher accuracy in determining the right class of an instance.

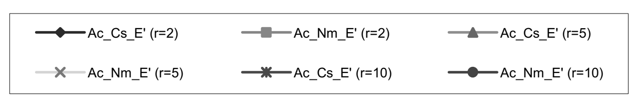
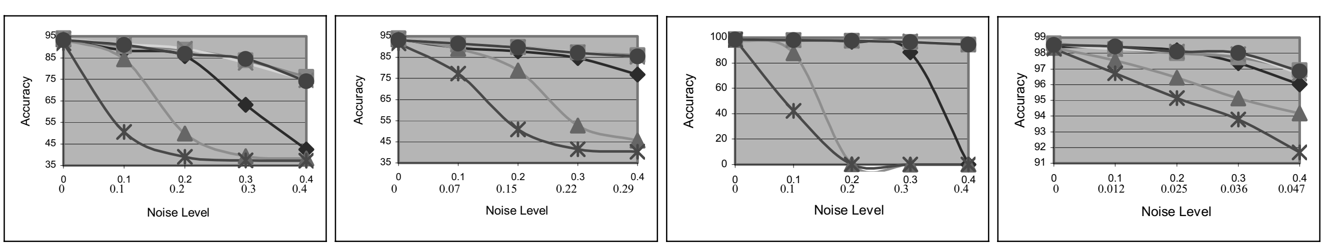


Fig. 3. The meaning of each curve in Fig. 4



(a) WDBC Dataset (*TRC*) (b) WDBC Dataset (*PRC*) (c) Sick Dataset (*TRC*) (d) Sick Dataset (*PRC*)

Fig. 4. Impacts of class noise on the classification accuracy of normal and *CS* classification algorithms

4.3 Misclassification Costs and Accuracies

For a given example I_k, assume we know the probability of each class j, $P(j|I_k)$, the Bayes optimal prediction for I_k is the class i that minimizes the conditional risk (Domingos 99, Duda & Hart 73), as defined by Eq. (15). The conditional risk $P(i|I_k)$ is the conditional probability of predicting instance I_k belongs to class i. According to the Bayesian formula, we can transform Eq. (15) to Eq. (17). The Bayes optimal prediction is guaranteed to achieve the lowest possible overall cost (*i.e.*, the lowest expected cost over all possible examples I_k, weighted by their probabilities $P(I_k)$).

With Eq. (17), it is clear that $C(i, j)$, $P(I_k \mid j)$, and α_j, together with the rule above imply a partition of the example space with j regions, such that class j is the optimal (least-cost) prediction in region j. The goal of cost-sensitive classification is to find the frontiers between these regions, explicitly or implicitly. However, this is complicated by two factors: (1) their dependence on the cost matrix C; and (2) the changes of both likelihood $P(I_k \mid j)$ and prior probability α_j of different classes j, caused by the existence of noise. In general, as misclassifying examples of class j becomes more expensive relative to misclassifying others, the region where j should be predicted will expand at the expense of the regions of other classes, even if the class probabilities $P(j|I_k)$ remain unchanged. In noisy environments, the probability of each class j, $P(j|I_k)$, becomes less reliable, due to the reason that noise modifies the likelihood $P(I_k|j)$ and the prior probability α_j. With Eq. (15), it is understandable that any error introduced to the likelihood $P(I_k|j)$ is going to be magnified by the cost matrix $C(i, j)$ (the *CS* classifier tends to favor "expensive" classes), even if noise does not change the *apriori* class distribution α_j. As a result, the higher the cost-ratio in the dataset, the more sensitive the *CS* classifier behaves to the data errors.

$$\arg\min_i P(i \mid I_k) = \arg\min_i \sum_j P(j \mid I_k) \cdot C(i, j) \qquad (15)$$

According to Bayesian formula, we have

$$P(j \mid I_k) = \frac{P(I_k \mid j) \cdot P(j)}{\sum_{i \in Y} P(I_k \mid i) \cdot P(i)} = \frac{P(I_k \mid j) \cdot P(j)}{P} \qquad (16)$$

where $P(I_k \mid j)$ is the likelihood of instance I_k, given a particular class j, and P is the sum of all the likelihoods. Assume α_j=$P(j)$ is the prior probability of class j. Then, Eq. (15) can be transformed as follows:

$$\arg\min_i P(i \mid I_k) = \arg\min_i \sum_j \frac{P(I_k \mid j) \cdot C(i, j) \cdot \alpha_j}{P} \qquad (17)$$

5 Conclusions

This paper has empirically and theoretically studied the impacts of mislabeled training examples (commonly referred to as class noise) on cost-sensitive learning. We observed behaviors of cost-sensitive learners trained from different noisy environments and assessed their performances in terms of average misclassification cost and classification accuracy. Our quantitative study concludes that the existence of class noise can bring serious troubles (higher misclassification costs and lower classification accuracy) for cost-sensitive learning, especially when misclassifying some classes becomes extremely expensive. Removing noisy instances will significantly improve the performance of a cost-sensitive classifier learned from noisy environments. Our observations suggest that cost-sensitive learning should be carefully conducted in noisy environments. Comparing to general inductive learning, the success of the cost-sensitive learning crucially depends on the quality of the underlying data, where system performances can deteriorate dramatically in the presence of a small portion of data errors.

References

Abe N. & Zadrozny B. (04), An iterative method for multi-class cost-sensitive learning. *Proc. of the 10th KDD Conference*, 2004.

Blake C. & Merz C. (98), UCI Data Repository, 1998.

Bradford, J. Kunz C., Kohavi R., Brunk C., & Brodley C. (98), Pruning decision trees with misclassification costs. *Proc. of the ECML Conf.*, 1998.

Brodley C. & Friedl M. (99), Identifying mislabeled training data. *Journal of AI Research*, 11:131-167, 1999.

Brefeld U., Geibel P. & Wysotzki F. (03), Support vector machines with example dependent costs. *Proc. of the ECML Conf.*, 2003.

Chan P. & Stolfo S. (98), Toward scalable learning with non-uniform class and cost distributions. *Proc. of the KDD Conf.*, 1998.

Domingos P. (99), MetaCost: a general method for making classifiers costsensitive. *Proc. of the 5th KDD Conference*, 1999.

Duda R. & Hart P. (73), *Pattern classification and scene analysis*, Wiley, New York, 1973.

Gamberger D., Lavrac N., & Groselj C. (99), Experiments with noise filtering in a medical domain. *Proc. of the ICML Conf.*, 1999.

Geibel P. & Wysotzki F. (03), Perceptron based learning with example dependent and noisy costs. *Proc. of the ICML Conf.*, 2003.

Pazzani M., Merz C., Murphy P., Ali K., Hume T. & Brunk C. (94), Reducing misclassification costs. *Proc. of the 11th ICML Conference*, 1994.

Quinlan J. (86), Induction of decision trees. *Machine Learning*, 1(1): 81-106, 1986.

Quinlan, J. (86b), The effect of noise on concept learning*, Machine Learning, Morgan Kaufmann*, 1986.

Quinlan J. (97), http://rulequest.com/see5-info.html, 1997.

Tan M. (93), Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning*, 13:7-33, 1993.

Turney P., Cost-sensitive learning bibliography, http://purl.org/peter.turney/bibliographies/cost-sensitive.html.

Zadrozny B., Langford J. & Abe N. (03), Cost-sensitive learning by cost-proportionate example weighting. *Proc. of the 3rd ICDM Conference*, 2003.

Zhu, X.& Wu, X., (04), Cost-guided Class Noise Handling for Effective Cost-sensitive Learning. *Proc. of the ICDM Conf.*, 2004.

Zubek V. & Dietterich T. (02), Pruning improves heuristic search for cost-sensitive learning. *Proc. of the ICML Conference*, 2000.