An extended tuning method for cost-sensitive regression and forecasting

Huimin Zhao a,⁎, Atish P. Sinha a, Gaurav Bansal b

a Sheldon B. Lubar School of Business, University of Wisconsin-Milwaukee, P. O. Box 742, Milwaukee, WI 53201–0742, United States
b University of Wisconsin–Green Bay, 2420 Nicolet Dr., Green Bay, WI 54311–7001, United States

1. Introduction

Real-world classification and regression problems [3,7,8,37] are often characterized by asymmetric costs (losses), which are associated with different types of prediction errors. For example, to a banking regulatory authority, incorrectly classifying a failing bank as a sound one has more serious consequences than raising a false alarm on a healthy bank [28,30]. To a real estate assessor, overestimating the value of a house incurs heavier costs to the assessor’s office than underestimating the house value by the same amount [34]. Such problems entail the use of cost-sensitive learning, which attempts to minimize the expected cost due to misprediction errors, rather than minimize simple measures such as error rate (in classification) and mean squared error (in regression).

Most research on cost-sensitive learning in the data mining literature has focused on classification problems [33]. There are in general two approaches to cost-sensitive classification [42]. One is to account for the asymmetric costs prior to or during model learning, by assigning appropriate weights on different training examples [15,24,32,38] or directly modifying the objective function or learning algorithm [16,18,20,41]. The other is to tune, in a post-hoc manner, the decision threshold of a model learned by a regular learning algorithm. This method can deal with general cost structures given the regular regression model, is found using a hill-climbing search algorithm. This method adjusts the prediction of a regular regression model by a certain amount. The amount of adjustment that minimizes average misprediction cost, under particular asymmetric loss functions, such as LinLin (asymmetric linear), QuadQuad (asymmetric quadratic), LinEx (approximately linear on one side and exponential on the other side), and SquarEx [31,39]. While some deliberately designed loss functions, such as LinEx and SquarEx, lead to closed-form solutions, the optimal predictor under general loss structures typically does not exist in closed form, thus entailing numeric solutions or approximations (in either the predictor or the loss function) [10,11].

Recently, inspired by the post-hoc tuning method for cost-sensitive classification, Bansal et al. [4] proposed a method for tuning a regular regression model so as to minimize average misprediction cost under an asymmetric cost structure. This method adjusts the prediction of a regular regression model by a certain amount. The amount of adjustment that minimizes average misprediction cost, given the regular regression model, is found using a hill-climbing search algorithm. This method can deal with general cost structures and does not require modifying the underlying learning method. This is a big advantage for organizations that operate in dynamic environments where the shape and parameters of the cost structure could change over time. Empirical evaluation in the domain of bank loan charge-off forecasting shows that this tuning method significantly reduces the average misprediction costs of models trained with several regression methods, including linear least squares, model tree, and backpropagation neural network.

In this paper, we propose an extended tuning method, which finds a polynomial function of the prediction of a regular regression model to make the final cost-sensitive prediction. We also propose an iterative hill-climbing algorithm for finding the optimal coefficients of polynomial tuning functions. The method of Bansal et al. [4] becomes a special case of the new method, where the tuning function is linear...
and has a fixed unit coefficient on the term of degree one. We have also empirically evaluated our proposed method in the domain of bank loan charge-off forecasting. The results demonstrate significant further performance improvement of the new method over its predecessor.

The rest of the paper is organized as follows. In the next section, we briefly review the related literature. We then briefly describe some necessary background and the tuning method proposed by Bansal et al. [4]. We next propose the extended tuning method and present the results of the empirical evaluation. Finally, we conclude the paper with a discussion of managerial implications and potential future research directions.

2. Literature review

In this section, we review the relevant literature on cost-sensitive learning for classification and regression problems. There is a large body of research on cost-sensitive learning for classification problems in the data mining field. Several cost-sensitive classification methods have been proposed. Some of them extend particular basic classification methods, which rely on symmetric performance measures. Fan et al. [18] proposed the AdaCost algorithm, which turns the AdaBoost algorithm into a cost-sensitive boosting method. Gama [20] proposed the Iterative Bayes method, which modifies Naive Bayes to accommodate asymmetric cost structures. Drummond and Holte [16] proposed methods for cost-sensitive decision tree learning. Zhao [41] proposed a method for simultaneously building a set of alternative decision trees, each of which excels under a particular cost setting.

Besides such methods for extending particular classification methods, there are two generic approaches for converting any classification method into a cost-sensitive one: instance weighting and post-hoc tuning. Such methods are called meta methods and have broader applicability than those extending particular regular classification methods. The instance weighting approach [15,24,32,38] weights different types of training instances based on the (asymmetric) cost structure prior to classifier training. This is similar to instance re-sampling (over-sampling or under-sampling) [9]. The classifier training method is not modified, but as fewer errors are made on the more costly type, the overall misclassification cost is lowered. The post-hoc tuning approach [17,29] does not modify the training data or the training method and only adjusts (tunes) the trained classifier. This approach follows two steps. It first learns a regular cost-neutral classifier without considering the actual asymmetric cost structure. It then relies on the posterior class probabilities predicted by the base classifier and finds decision thresholds that minimize the average misclassification cost.

Zhao [42] showed that the two approaches lead to similar results for some base classification methods, such as Naive Bayes, and quite different results for other methods, such as decision trees. The instance weighting approach is computationally more involved than the post-hoc tuning approach, especially in dynamic environments where the cost structure could change over time, because then instance weighting requires model retraining whereas post-hoc tuning does not.

It has long been recognized that regression problems—like classification problems—can be characterized by asymmetric losses too (e.g., [2,5,19,22,34,40]) and it is inappropriate to “blindly trust in squared error loss” [39], although the main results of this literature are relatively recent [25]. Among others, Varian [34] presented one of the earliest studies of cost-sensitive regression. The particular problem studied was that of real estate assessment. Real estate assessors need to routinely estimate current market values of taxable properties based on the values of certain characteristics of the properties. Overestimates and underestimates incur different losses accruing to the assessor’s office. In case of an underestimate, the loss is equal to the amount of estimation error. However, in case of overestimates, the assessor’s office may be faced with complaints or appeals, which entail lengthy expensive procedures to resolve. Varian [34] suggested that the symmetric quadratic loss function of ordinary least squares procedures, which had been applied in practice, was inappropriate and proposed the LinEx loss function as an alternative. This loss function has approximately exponential losses for large overestimates and approximately linear losses for large underestimates. In a related but different context, that of property valuation by a real estate agent, Cain and Janssen [6] applied LinLin, QuadQuad, and LinEx loss functions. Zellner [39] derived optimal estimators and predictors relative to Varian’s LinEx loss function for a number of well-known models. Thompson and Basu [31] generalized Varian’s LinEx to SquarEx, which has approximately quadratic losses for large underestimates and approximately exponential losses for large overestimates and includes LinEx as a special case.

Although closed-form solutions have been found for a few special cases, such as LinEx and SquarEx, it has been proven that an optimal predictor under general loss structures does not exist in closed form [10,11]. Numeric or approximate solutions are therefore needed for general loss functions. Christoffersen and Diebold [11] proposed an approximately optimal predictor via series expansions. With a different and complementary approach, Christoffersen and Diebold [10] proposed a method for finding the exactly optimal predictor for an approximate loss function, instead of approximating the optimal predictor for the exact loss function. This method approximates a general loss function by a piecewise-linear loss function, which is constructed by concatenating linear segments.

In addition to the extensive research on cost-sensitive regression published in the statistics literature, a few publications (e.g., [4,12,13,33]) have appeared in the data mining literature. Crone [12] and Crone et al. [13] proposed a modified back-propagation neural network training method, which assumes a LinLin cost function rather than the squared error cost used in the standard method. Torgo and Ribeiro [33] proposed a case-specific, cost-sensitive performance measure, which is not just a function of the prediction error, but a function of the actual and predicted target values, and empirically compared a few standard regression methods using this measure. However, they did not suggest any method for optimizing or improving this performance measure.

While all previous methods for cost-sensitive regression try to directly learn a model under the actual asymmetric cost structure, the post-hoc tuning method recently proposed by Bansal et al. [4]—like the post-hoc tuning method for cost-sensitive classification problems—follows a two-step approach. It first learns a regular cost-neutral classifier without considering the actual asymmetric cost structure. It then uses the regular regression model as an aggregate, intermediate explanatory variable for making the ultimate cost-sensitive prediction, reducing the problem to a univariate one, analogous to how the tuning method for cost-sensitive classification works. Unlike other cost-sensitive regression methods, which extend particular regular regression methods under particular cost functions, this method is a meta method and has much broader applicability. Furthermore, as this method only tunes a regular regression model in a post-hoc manner, it does not require model retraining when the cost function changes. However, the particular tuning method (adding a certain amount) proposed by Bansal et al. [4] is rather restrictive since it only adjusts the prediction of a regular regression model by a certain amount. In this paper, we propose a more general tuning method, which extends the previous method by adjusting a regular regression model with a polynomial function. The previous method then becomes a special case of the proposed method.

3. Background

We adopt the performance measure for cost-sensitive regression used by Bansal et al. [4]. For a regression problem with a dependent
variable $y$ and a vector of independent variables $x$, a regression model is a mapping $f : x \to y$, learned from a training sample $S = \{x_i, y_i \mid i = 1, 2, ..., N\}$ by some learning method. If a prediction error $e$ incurs a cost $C(e)$, the average misprediction cost of model $f$, as estimated on $S$, is defined as

$$\theta = \frac{1}{N} \sum_{i=1}^{N} C(f(x_i) - y_i).$$

(1)

Regular regression methods, such as least-squares linear regression, model tree, and neural network, presume a symmetric cost function, i.e., $C(e) = C(-e)$ for any misprediction error $e$. For example, least-squares linear regression minimizes mean squared error, assuming a symmetric squared-error cost function. However, in many real-world regression problems, such as real estate assessment, system reliability estimation, and bank loan charge-off forecasting, the cost function is asymmetric since the consequences of under-prediction (negative) and over-prediction (positive) errors are different. In the case of $\theta$ is a convex function with regard to prediction error at a quadratic rate. In other more extreme cost functions, such as LinEx and SquarEx, even the shape of the cost function is different for the two types of errors. More generally, even the form of the cost function on the same side may change as the amount of error changes. For example, the unit cost (in the case of linear cost) associated with high overestimation (underestimation) may be different from that associated with moderate overestimation (underestimation).

The cost function is necessarily problem dependent. Defining an appropriate cost function is often an art and requires deep domain expertise in many applications. For example, Varian [34] adopted a LinEx cost function for the real estate assessment problem, while Thompson and Basu [31] suggested that SquarEx may be more appropriate than LinEx for the system reliability estimation problem. The use of such extremely asymmetric cost functions is necessarily dependent on a deep understanding of the application domain.

Apparently, models learned by regular regression methods, which are designed to minimize symmetric costs, will not be optimal when the cost function is actually asymmetric. Given an asymmetric cost function $C$, Bansal et al. [4] adjusted the prediction of a regular regression model $f$ by an amount of $\delta$ so as to minimize the average misprediction cost $\theta$, resulting in an adjusted model $f' = f + \delta$. The original model $f$ minimizes a symmetric cost measure, whereas the adjusted model $f'$ minimizes the asymmetric cost measure given the original model. As $\theta$ is a convex function with regard to $\delta$ when the cost function $C$ is convex, an efficient hill climbing algorithm was employed to search for the optimal $\delta$.

4. Extended tuning method

We observe that the post-hoc tuning method of Bansal et al. [4] can be generalized into a two-step scheme. First, a regression model $f$ is learned as usual without considering the cost asymmetry. The prediction error of this regression model on instance $i$ is $f(x_i) - y_i$. Next, $g$, a function of $f$, is found so as to minimize the average misprediction cost $\theta$. The final prediction for the dependent variable $y$ is given by an adjusted regression model $f' (x) = g(f(x))$. The prediction error of the adjusted regression model on instance $i$ is $f'(x_i) - y_i = g(f(x_i)) - y_i$. The tuning method of Bansal et al. [4] is a special case where the tuning function is in the form of $g(f) = f + \delta$. At the tuning step, the base regression model $f$ is essentially treated as an aggregate, intermediate explanatory variable for making the ultimate cost-sensitive prediction, reducing the tuning function $g$ to a univariate one (i.e., $g$ is a function of $f$). This contrasts to methods that try to directly learn a regression model under the actual asymmetric cost structure without using the regular model $f$ as an intermediate variable.

Furthermore, under some conditions, the optimal tuning function $g$ for a given base regression model $f$ may be found efficiently—i.e., in a reasonable time—using a heuristic approach such as hill climbing. For example, it can be shown (in Proposition 1 below) that, under the following assumptions, the average misprediction cost $\theta$ is convex with regard to the parameters of $g$ and therefore any local minimum of $\theta$ is also a global minimum:

(a) The cost function, $C(e)$, is convex with regard to prediction error, $e$.

(b) The tuning function $g$ is a polynomial of $f$ of degree $m$, i.e., $g(f) = \sum_{j=0}^{m} \beta_j f^j$, $f^0 = 1$. The prediction error of the adjusted regression model on instance $i$ is $\sum_{j=0}^{m} \beta_j f(x_i) - y_i$. The average misprediction cost of the adjusted regression model can be expressed as the following function of the vector of coefficients $B = (\beta_0, \beta_1, ..., \beta_m)^T$.

$$\theta(B) = \frac{1}{N} \sum_{i=1}^{N} C(g(f(x_i)) - y_i) = \frac{1}{N} \sum_{i=1}^{N} C \left( \sum_{j=0}^{m} \beta_j f(x_i) - y_i \right).$$

(3)

Fig. 1. Cost function examples.
While these assumptions put restrictions on the cost function and tuning function theoretically, we believe that the proposed method has wide applicability. All cost functions that have been studied in the literature (e.g., LinLin, QuadQuad, LinEx, and SquarEx) are convex. A large range of possible tuning functions can be closely approximated by a polynomial function with a sufficient degree.

**Proposition 1.** Under the above-mentioned assumptions, the average misprediction cost of the adjusted regression model, $\hat{\theta}(\beta)$, is convex with regard to $\beta$.

**Proof.** The proof of proposition 1 is trivial. Since the adjustment, characterized by $\beta$, is applied on a regression model $f$ after the model has been learned, the prediction of $f$ on a given problem instance, $f_i(x_i)$, $i = 1, 2, ..., N$, is constant with respect to $\beta$. Since the cost function $C(e)$ is convex with regard to $e$ and convexity is invariant under affine maps ($x \mapsto Ax + b$), the misprediction cost of the adjusted model on a problem instance, $C(\sum_{j=0}^{m} \beta_j f_j(x_i) - y_i) = C(f^T \beta - y_i)$, $i = 1, 2, ..., N$, where $f = [f(x_1), f^2(x_1), ..., f^m(x_1)]^T$, is convex with regard to $\beta$. The summation $\sum_{i=1}^{N} C\left(\sum_{j=0}^{m} \beta_j f_j(x_i) - y_i\right)$ is then convex with regard to $\beta$.

The average misprediction cost of the adjusted model, $\hat{\theta}(\beta) = \frac{1}{N} \sum_{i=1}^{N} C\left(\sum_{j=0}^{m} \beta_j f_j(x_i) - y_i\right)$, is therefore convex with regard to $\beta$. □

When the average misprediction cost $\theta$ is convex with regard to the parameters of the tuning function and therefore any local minimum of $\theta$ is also a global minimum, the hill climbing algorithm of Bansal et al. [4] can be extended to repeatedly and alternately search on the parameters $\beta_j$, $j = 0, 1, ..., m$, until no further improvement is possible. Fig. 2 outlines the overall algorithm.

The algorithm (named CostSensitiveRegression) can work with any base regression method and any convex cost function. It takes a base regression method, a training sample, a convex cost function, the maximum degree of the polynomial tuning function to be found, and a given precision for adjusting the coefficients of the tuning function as

**CostSensitiveRegression ($\Gamma$, $S$, $C$, $m$, $p$)**

1. $f := \Gamma(S)$. /* Learn a regression model $f$ using $\Gamma$ based on $S$. */
2. $\beta := (\beta_0, \beta_1, ..., \beta_m)^T := (0, 1, 0, ..., 0)^T$.
3. DO
   3.1 $\beta_{old} := \beta$.
   3.2 FOR $d$ FROM 1 TO $m$ $\cdots$
   3.2.1 $\text{HillClimbing}(S, C, m, p, f, \beta, d)$.
   WHILE $\beta \neq \beta_{old}$ /* End of 3. */
4. RETURN $\sum_{j=0}^{m} \beta_j f_j$. /* Return an adjusted regression model. */

**HillClimbing($S$, $C$, $m$, $p$, $f$, $\beta$, $d$).**

1. $\bar{\beta'} := \beta'$ := $\beta$, $\beta_{d}' := \beta_{d} + p$, $\beta_{d} := \beta_{d} - p$.
2. IF $\hat{\theta}(\bar{\beta'}) < \hat{\theta}(\beta)$, /* $\hat{\theta}$ is defined in equation (2). */
   2.1 $\delta := 1$. /* Set the direction of adjustment. */
   3. ELSE IF $\hat{\theta}(\bar{\beta'}) < \hat{\theta}(\beta)$,
   3.1 $\delta := -1$.
   4. ELSE
   4.1 RETURN.
5. $\beta := \beta'$.
6. DO
   6.1 $s := 1$. /* Set the initial hill climbing stride. */
   6.2 $\beta := \beta'$.
   6.3 DO
   6.3.1 $\beta := \beta' := \beta'$.
   6.3.2 $\beta_{d} := \beta_{d} + s \times p \times \delta$.
   6.3.3 $s := s \times 2$. /* Double the stride after each step. */
   6.3.4 $\beta_{d} := \beta_{d} + s \times p \times \delta$.
   WHILE $\hat{\theta}(\bar{\beta'}) < \hat{\theta}(\beta)$. /* End of 6.3. */
   WHILE $s > 2$. /* End of 6. */

Fig. 2. An extended tuning algorithm for cost-sensitive regression.
inputs and returns an adjusted regression model as the output. First, a regular regression model is trained using the base regression method (step 1). Initially, the tuning function is simply the identity function (i.e., \( g(f) = f \)) (step 2). Then, a loop is used to repeatedly search for the optimal coefficients of the terms in the tuning function until no further improvement is possible (step 3). During each iteration, a nested loop is used to alternately search for the optimal coefficient of each term in the tuning function, given the current coefficients of the other terms, by calling a hill climbing procedure (step 3.2). Finally, an adjusted regression model that minimizes the average misprediction error is returned (step 4).

The hill climbing procedure (named HillClimbing) searches for the optimal coefficient of a particular term in the polynomial tuning function given the current coefficients of the other terms. First, the direction of climbing, which leads to lower average misprediction cost, is determined via trials (steps 1 to 4). Starting with the current coefficients (step 5), several iterations of hill climbing in the determined direction are then carried out to approach the optimal coefficient of the term under search until the number of climbing steps during an iteration falls below two and thus no further climbing is promising (loop at step 6). During each iteration of hill climbing, several climbing steps are attempted until the performance does not improve any more (nested loop at step 6.3). To speed up the procedure, the climbing stride starts from the given precision of adjustment (step 6.1) and is doubled after every climbing step (step 6.3.3).

Note that a linear tuning function, including the special case of Bansal et al. [4] where the coefficient of the linear term is fixed at 1, does not change the model complexity. Applying a linear tuning function on a linear regression model, for example, the resulting adjusted regression model is still a linear one. A high-degree tuning function, however, will increase the model complexity. Applying a quadratic tuning function on a linear regression model, for example, will result in a quadratic adjusted regression model.

In general, as we increase the degree of the tuning function (\( m \)) and thus the resulting model complexity, we are able to lower the average misprediction cost on the training sample. However, the and thus the resulting model complexity, we are able to lower the quadratic tuning function on a linear regression model, for example, the resulting model complexity. Applying a linear tuning function on a linear regression model, for example, the resulting does not change the model complexity. Applying a linear tuning function on a linear regression model, for example, the resulting does not change any more or the testing performance starts to decrease.

The precision of adjustment (\( p \)) controls how closely the program can approach the optimal solution. If \( p \) is set too large, the found solution may not be satisfactorily close to the optimal; if it is too small, too much computation time is incurred. The user may experiment with different values until the solution does not change much or the computation time is not affordable any more. A straightforward enhancement in a practical tool is to allow the user to reduce \( p \) and continue with finer tuning on the basis of the current solution at any point without starting from scratch.

5. Empirical evaluation

We have implemented the extended tuning method and empirically compared it with the one presented by Bansal et al. [4]. In this section, we report on the implementation and empirical evaluation.

5.1. Implementation and experimental environment

We implemented the CostSensitiveRegression algorithm (Fig. 2) in Java by extending the Classifier class in the Weka machine learning toolkit [36] (http://www.cs.waikato.ac.nz/ml/weka/). The program, including source code, is available from the authors. We ran the experiment on a Dell Optiplex/GX620 workstation, configured with a 3 GHz Pentium D CPU and 1 GB of RAM, running the Windows XP operating system.

The extended tuning algorithm takes five inputs: a base regression method, a training sample, a convex cost function, the maximum degree of the polynomial tuning function to be found, and a given precision for adjusting the coefficients of the tuning function. The inputs are passed to the CostSensitiveRegression program through command options in the fashion of Weka. The program can work with any base regression method. In the experiment, we used three base regression methods available in Weka: the standard least-squares linear regression (LR), M5 model tree [26], and backpropagation neural network (NN) [27]. We retained the default parameter settings for these methods in Weka. The training sample was presented in the ARFF data file format of Weka.

The program can also work with any convex cost function. In the experiment, we implemented and evaluated the LinLin and QuadQuad cost functions. We evaluated linear, quadratic, cubic, and quartic tuning functions (i.e., polynomial tuning functions with a maximum degree of one, two, three, and four, respectively). The precision for adjusting the coefficients of the tuning functions we used in the experiment was 0.00001.

5.2. Problem domain

Following Bansal et al. [4], we conducted the evaluation in the domain of loan charge-off forecasting, which is characterized by asymmetric costs on misprediction errors. For a bank, under-predicting its future loan charge-off by a certain amount is riskier than over-prediction by the same amount. Taking this into account, Bansal et al. [4] decided to penalize under-predictions more heavily than over-predictions while tuning regression models.

Banks determine their loan loss reserves based on their predictions of future loan charge-off amounts. If a bank over-predicts its future loan charge-off, it will need to maintain extra funds in the loan loss reserves and thus suffer reduced earnings—the reserves are deducted from earnings—and possibly receive a lower credit score from financial analysts. Under-predicting the loan charge-off, however, has more serious consequences as it presents a rosier picture of an otherwise worse scenario. Besides the bank management, outside bodies such as investors, accountants, and regulators are also interested in knowing whether a bank is adequately prepared for its future loan losses. If a bank does not have sufficient loan loss reserves, the consequences could be dire. Financial analysts who are responsible for a bank’s credit rating also consider the bank’s reserve adequacy in the rating. If a bank under-predicts its future loan charge-off and thus does not prepare sufficient provision for its loan losses, it will not only face the wrath of investors, accountants, and regulators but also experience an even greater downturn in its credit rating. Hence, under-prediction is much costlier than over-prediction in the domain of loan charge-off forecasting.

5.3. Data and cost functions

The loan charge-off forecasting data we used came from Wharton Research Data Services (WRDS, https://wrds.wharton.upenn.edu/). We used the most recent data (of 20 quarters between 2004 and 2008) of commercial banks available in the Bank Regulatory database of WRDS. The database contains data of all banks filing the Report of Condition and Income (known as the "Call Report") regulated by the Federal Reserve System, Federal Deposit Insurance Corporation, and the Comptroller of the Currency. We used the same 14 variables (listed in Table 1 with the variable codes, names, and definitions from WRDS) as Bansal et al. [4], who used the first 13 variables in a
particular quarter as the independent variables and the loan charge-off in the same quarter as the dependent variable in regression models. Such modeling may be appropriate for studying the effects of the independent variables on loan charge-off, but application of an asymmetric loss function for a cost-sensitive decision problem lends itself to a forecasting scenario. As Whittle [35, p. 106] note:

“In most cases the predicted value, once obtained, is used to initiate or modify a course of action... In this larger context the problem of predicting is more familiar to the practitioner, who is usually more concerned with how well he has done with the task he is assigned than with the theoretical elegance of his model.”

Thus, there were 18 training datasets and 18 testing datasets. Observations with missing values were discarded. The remaining datasets contained an average of 7684 (a minimum of 7351 and a maximum of 8042) observations. As the distributions of the variables were largely skewed, (natural) logarithm transformations were applied to reduce the extent of skewness.

As discussed earlier, under-prediction is considered much costlier than over-prediction in this domain. Following Bansal et al. [4], we used a LinLin cost function with a steeper cost slope for under-prediction as compared to over-prediction. We fixed the unit cost for over-prediction at 1 and examined the following cost ratios (under-prediction to over-prediction): 100:1, 50:1, 20:1, and 10:1. To evaluate the generalizability of the proposed method, we also used a QuadQuad cost function. Again, we fixed the unit cost for over-prediction at 1 and examined the same four cost ratios.

5.4. Results

Table 2 summarizes the results of conventional performance measures, including Pearson correlation coefficient, mean absolute error, and mean squared error. Each reported quantity is an average over the 18 training or testing datasets. We include these measures here for the sake of completeness. However, absolute error and squared error are symmetric (i.e., cost ratio between under-prediction and over-prediction is 1:1) special cases of LinLin and QuadQuad cost functions, respectively, and are not suitable for the loan charge-off forecasting problem where the cost structure is clearly asymmetric. Linear tuning functions, including the special case of Bansal et al. [4], do not change the Pearson correlation coefficient, which remains invariant under linear transformations. High-degree tuning functions will increase the model complexity and Pearson correlation coefficient is not appropriate for measuring nonlinear dependence.
Tables 3 and 4 summarize the results of the average misprediction cost under LinLin cost functions on the training and testing data, respectively. Each reported quantity is an average over the 18 training or testing datasets. “None” refers to the original regression models without tuning. “BSZ” refers to the tuning method of Bansal, Sinha, and Zhao [4]. Figs. 3 and 4 contrast the performance of the extended tuning method and that of the BSZ method on the training and testing data, respectively.

The linear tuning function outperformed the BSZ tuning function on the training data for every base regression method under every cost ratio. This performance improvement (from the BSZ tuning function to the linear tuning function) continued to hold on the testing data. A repeated-measures, factorial ANOVA under each cost ratio, with tuning function (BSZ and linear) and base regression method (LR, M5, and NN) as factors, showed that the performance difference between the two tuning functions was statistically significant ($F(1, 17) > 14.1$, $p < 0.01$). Intuitively, the linear tuning function, as compared to the BSZ tuning function, increases the flexibility of the tuned model in fitting the training data without increasing the model complexity, hence leading to significant performance improvement.

As the degree of the polynomial tuning function increased, the average misprediction cost of every base regression method on the training data continued to decrease, indicating that the tuned models could fit the training data better, but the performance gradually flattened out. However, the performance improvement did not always generalize to the testing data. The average misprediction cost of some base regression methods on the testing data started to increase at some points, indicating that overfitting might have occurred due to the increased model complexity. Intuitively, the high-degree (quadratic, cubic, and quartic) tuning functions did not work well because such transformations increased the complexity of the base regression models and, possibly, invalidated some assumptions behind the base regression models. For example, the quadratic tuning function not only changes a linear regression model into a quadratic one but also introduces interactions among independent variables. The increased model flexibility hardly compensated for the increase in model complexity. It turned out that none of the high-degree tuning functions improved upon the linear tuning function significantly.

<p>| Table 3 | Average misprediction costs under LinLin cost functions on the training data. |
|---------|-----------------------|----------------------|-----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Cost ratio</th>
<th>Tuning method</th>
<th>None</th>
<th>BSZ</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>10</td>
<td>6.150</td>
<td>2.632</td>
<td>2.614</td>
<td>2.613</td>
<td>2.611</td>
<td>2.611</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>11.740</td>
<td>3.122</td>
<td>3.077</td>
<td>3.076</td>
<td>3.074</td>
<td>3.073</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>10</td>
<td>6.024</td>
<td>2.620</td>
<td>2.589</td>
<td>2.585</td>
<td>2.579</td>
<td>2.579</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>27.932</td>
<td>3.741</td>
<td>3.620</td>
<td>3.619</td>
<td>3.613</td>
<td>3.612</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>10</td>
<td>6.287</td>
<td>2.596</td>
<td>2.563</td>
<td>2.555</td>
<td>2.547</td>
<td>2.547</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>11.520</td>
<td>3.072</td>
<td>3.010</td>
<td>3.005</td>
<td>3.001</td>
<td>3.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>56.984</td>
<td>4.160</td>
<td>3.992</td>
<td>3.979</td>
<td>3.969</td>
<td>3.967</td>
<td></td>
</tr>
</tbody>
</table>

Table 5 and 6 summarize the results of the average misprediction cost under QuadQuad cost functions on the training and testing data, respectively. Again, each reported value is an average over the 18 training or testing datasets. Figs. 5 and 6 contrast the performance of the extended tuning method and that of the BSZ method on the training and testing data, respectively.

The findings from the QuadQuad cost function were similar to those from the LinLin cost function. The linear tuning function again outperformed the BSZ tuning function on both the training and testing data for every base regression method under every cost ratio. A repeated-measures, factorial ANOVA under each cost ratio, with tuning function (BSZ and linear) and base regression method (LR, M5, and NN) as factors, showed that the performance difference between the two tuning functions was statistically significant ($F(1, 17) > 19.9$, $p < 0.001$).

The cost reduction from BSZ to linear appears to be small, in the range of 0.7% to 10%. However, the cost values are on a natural log scale as the variables, including loan charge-off, have gone through natural logarithmic transformations. The cost reduction on the original scale is in the range of 1.9% to 25%, which would be considered practically useful for the bank loan charge-off forecasting problem.

As the degree of the polynomial tuning function further increased, the performance of every base regression method on the training data continued to improve, but the performance gradually flattened out. However, the average misprediction cost of some base regression methods on the testing data started to increase at some points, indicating that overfitting might have occurred due to the increased model complexity. Especially for M5, when the degree of the polynomial tuning function increased from quadratic to cubic and quartic, the performance dramatically deteriorated, showing the possibility of severe overfitting.

There are also some interesting findings about the relative performance of the three base regression methods. The method with the least model complexity, LR, achieved the worst performance on the training data, as expected. However, it often outperformed the other two methods on the testing data, indicating that the other methods might have overfitted the training data. This shows again the importance of selecting an appropriate level of model complexity for a particular problem [23].

For completeness, we also experimented with situations where over-prediction is costlier than under-prediction. We fixed the unit cost for under-prediction at 1 and examined the following cost ratios (under-prediction to over-prediction): 1:100, 1:50, 1:20, and 1:10. Note, however, such cost ratios are clearly unrealistic for loan charge-off forecasting and were used for testing the method only. The findings are similar to those when the cost ratios are greater than 1.

Table 7 summarizes the model training and tuning times. The time needed by the linear tuning function was less than two seconds. As
the degree of the tuning function increased, the tuning time increased too. The quartic tuning function took less than a minute.

5.5. Illustrative example

The linear regression model trained on the 2008 Quarter 1 dataset was

\[ y = f(x) = -0.511x_1 + 0.200x_2 + 0.064x_3 - 0.128x_5 + 0.011x_6 \\
+ 0.168x_8 + 0.569x_9 + 0.202x_{10} + 0.351x_{13} + 0.417x_{14} - 4.582. \]

The average misprediction cost of this basic model under the LinLin cost function with a cost ratio of 1:10 was 6.6 on the training dataset. The adjusted model after applying the BSZ tuning function was

\[ y = f(x) + 2.000 = -0.511x_1 + 0.200x_2 + 0.064x_3 - 0.128x_5 \\
+ 0.011x_6 + 0.168x_8 + 0.569x_9 + 0.202x_{10} + 0.351x_{13} + 0.417x_{14} - 2.582. \]

The average misprediction cost on the training dataset reduced to 2.84. The adjusted model after applying the linear tuning function was

\[ y = 0.892f(x) + 2.383 = -0.456x_1 + 0.178x_2 + 0.057x_3 - 0.114x_5 \\
+ 0.010x_6 + 0.150x_8 + 0.508x_9 + 0.180x_{10} + 0.313x_{13} \\
+ 0.372x_{14} - 1.704. \]

The average misprediction cost on the training dataset further reduced to 2.815. The adjusted model after applying the quadratic tuning function was

\[ y = -0.005f^2(x) + 0.953f(x) + 2.249. \]

Note that the adjusted model became quadratic. The average misprediction cost on the training dataset further reduced to 2.813. The average misprediction cost of the original model, the model tuned with the BSZ function, the model tuned with the linear function, and the model tuned with the quadratic function on the testing dataset, the 2008 Quarter 2 dataset, was 5.534, 2.923, 2.879, and 2.878, respectively.

Note that the models were built for prediction and forecasting purposes, rather than for explaining the impacts of the independent variables on loan charge-off. The loan charge-off amount of the current quarter was included in the models to predict the loan charge-off amount of the next quarter. We therefore do not attempt to interpret the coefficients of the independent variables in the models.

5.6. Experiments with other datasets

We experimented with four additional regression datasets: Abalone, Bank (8FM), House (8L), and Puma (8NH), which are available on the Weka Web site [36]. We used the same base regression methods (LR, M5, and NN), cost functions (LinLin and QuadQuad), cost ratios (1:100,
We randomly split each dataset into a training set with two thirds of the instances and a test set with the remaining instances. We conducted such splitting 20 times for each dataset and used the average performance over the 20 runs as an estimate of the true performance. The findings are similar in general although the best degree of tuning varies (the results are available from the authors.) As the degree of tuning increased, the performance on the training data tended to improve, but the performance improvement gradually flattened out. However, the performance on the testing data did not always continue to improve and even started to degenerate at some points. The degree of tuning that led to the best performance on the testing data varies across datasets and cost settings. While the linear tuning function appeared to be the best for the House dataset under most cost settings, performance improvement was observed when the degree of tuning increased to quadratic, cubic, and even quartic on other datasets. An appropriate degree of tuning needs to be empirically determined for a given dataset and cost setting.

![Fig. 4. Average misprediction costs of different tuning methods under LinLin cost functions on the testing data.](image)

![Table 5](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Cost ratio</th>
<th>Tuning method</th>
<th>None</th>
<th>BSZ</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>10</td>
<td>11.401</td>
<td>5.756</td>
<td>5.716</td>
<td>5.713</td>
<td>5.709</td>
<td>5.709</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>52.676</td>
<td>10.077</td>
<td>9.829</td>
<td>9.806</td>
<td>9.792</td>
<td>9.780</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>104.268</td>
<td>12.415</td>
<td>11.988</td>
<td>11.938</td>
<td>11.915</td>
<td>11.899</td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>10</td>
<td>11.141</td>
<td>5.630</td>
<td>5.574</td>
<td>5.567</td>
<td>5.547</td>
<td>5.547</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>21.278</td>
<td>7.204</td>
<td>7.162</td>
<td>7.156</td>
<td>7.125</td>
<td>7.125</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>102.374</td>
<td>12.290</td>
<td>11.754</td>
<td>11.744</td>
<td>11.695</td>
<td>11.677</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>10</td>
<td>11.997</td>
<td>5.632</td>
<td>5.519</td>
<td>5.491</td>
<td>5.473</td>
<td>5.473</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>22.646</td>
<td>7.248</td>
<td>7.060</td>
<td>7.031</td>
<td>7.004</td>
<td>7.003</td>
<td></td>
</tr>
</tbody>
</table>

![Table 6](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Cost ratio</th>
<th>Tuning method</th>
<th>None</th>
<th>BSZ</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>65.771</td>
<td>12.668</td>
<td>12.091</td>
<td>12.005</td>
<td>11.984</td>
<td>11.978</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>31.549</td>
<td>12.599</td>
<td>11.801</td>
<td>11.952</td>
<td>17.944</td>
<td>17.946</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>151.415</td>
<td>23.841</td>
<td>20.853</td>
<td>20.916</td>
<td>38.354</td>
<td>39.024</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>10</td>
<td>15.977</td>
<td>8.191</td>
<td>7.937</td>
<td>7.944</td>
<td>8.058</td>
<td>8.079</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>142.024</td>
<td>18.187</td>
<td>17.218</td>
<td>17.191</td>
<td>17.496</td>
<td>17.491</td>
<td></td>
</tr>
</tbody>
</table>

---

Practical forecasting problems often exhibit asymmetry in their cost structures. Classical cost functions, such as mean squared error, although analytically convenient and widely applied, are not appropriate for use in many practical applications. More realistic cost structures should be accounted for, either during model building or in a post-hoc model tuning step.

Following Bansal et al. [4], we have proposed an extended tuning method for cost-sensitive regression. The BSZ method becomes a special case of the proposed method, where the tuning function is linear and has a fixed unit coefficient on the term of degree one. The proposed method can better fit the training data by allowing finer tuning. In particular, the linear tuning function is more flexible than the BSZ method and does not increase the model complexity. Higher degree of tuning may also be found beneficial depending on the problem and cost setting. As this approach does not require modifying the underlying model learning method, it is applicable to a wide range of cost functions and can be easily implemented to work with models generated by commercial software packages. It can therefore be applied in practical applications, which may prescribe special cost functions and software packages. Furthermore, when a decision maker cannot precisely pin down the cost function and needs to evaluate a set of possible cost functions, this approach provides an additional advantage over methods that require model rebuilding whenever the cost function changes.

We evaluated the cost-sensitive regression models in the domain of loan charge-off forecasting using recent real-world banking data. The impressive evaluation results of our method suggest that it could be of major help to forecasters trying to fine-tune their forecasting process. It has implications specifically in the domain of loan charge-off prediction. Recently, the nation's largest banks were put under stress-test to determine which ones were fit to receive federal funds [21]. Among the factors the test examined was the amount of money that would be left in the bank's reserves when its loans default. A bank that can predict its loan charge-offs effectively would be in a good position to figure out how much money to put in its loan loss reserves. A classic example is JPMorgan, which managed to remain financially healthy when several other big banks started falling one by one, primarily because it had kept $23 billion in its rainy-day fund [21]. Amid the current financial crisis, it is even more critical for banks to have systems in place for forecasting loan losses and have adequate provisions as a safeguard against those losses. As the events during the last few months indicate, if a bank does not have sufficient loan loss reserves, the consequences could be dire. In the second quarter of 2009 alone, 45 banks failed, adding to a total of 81 for the first two quarters, thereby exerting enormous pressure on the government's deposit insurance fund [14].

Our work also opens up several avenues for future research. First, the approach can be applied in other forecasting problems. Second, other cost functions and base regression methods can be evaluated. Third, the consequences of relaxing some of the assumptions made in

6. Conclusions

Practical forecasting problems often exhibit asymmetry in their cost structures. Classical cost functions, such as mean squared error, although analytically convenient and widely applied, are not appropriate for use in many practical applications. More realistic cost structures should be accounted for, either during model building or in a post-hoc model tuning step.

Following Bansal et al. [4], we have proposed an extended tuning method for cost-sensitive regression. The BSZ method becomes a special case of the proposed method, where the tuning function is linear and has a fixed unit coefficient on the term of degree one. The proposed method can better fit the training data by allowing finer tuning. In particular, the linear tuning function is more flexible than the BSZ method and does not increase the model complexity. Higher degree of tuning may also be found beneficial depending on the problem and cost setting. As this approach does not require modifying the underlying model learning method, it is applicable to a wide range of cost functions and can be easily implemented to work with models generated by commercial software packages. It can therefore be applied in practical applications, which may prescribe special cost functions and software packages. Furthermore, when a decision maker cannot precisely pin down the cost function and needs to evaluate a set of possible cost functions, this approach provides an additional advantage over methods that require model rebuilding whenever the cost function changes.

We evaluated the cost-sensitive regression models in the domain of loan charge-off forecasting using recent real-world banking data. The impressive evaluation results of our method suggest that it could be of major help to forecasters trying to fine-tune their forecasting process. It has implications specifically in the domain of loan charge-off prediction. Recently, the nation's largest banks were put under stress-test to determine which ones were fit to receive federal funds [21]. Among the factors the test examined was the amount of money that would be left in the bank's reserves when its loans default. A bank that can predict its loan charge-offs effectively would be in a good position to figure out how much money to put in its loan loss reserves. A classic example is JPMorgan, which managed to remain financially healthy when several other big banks started falling one by one, primarily because it had kept $23 billion in its rainy-day fund [21]. Amid the current financial crisis, it is even more critical for banks to have systems in place for forecasting loan losses and have adequate provisions as a safeguard against those losses. As the events during the last few months indicate, if a bank does not have sufficient loan loss reserves, the consequences could be dire. In the second quarter of 2009 alone, 45 banks failed, adding to a total of 81 for the first two quarters, thereby exerting enormous pressure on the government's deposit insurance fund [14].

Our work also opens up several avenues for future research. First, the approach can be applied in other forecasting problems. Second, other cost functions and base regression methods can be evaluated. Third, the consequences of relaxing some of the assumptions made in
this paper can be studied. When the optimal tuning function of a given form under a particular cost function cannot be found by an efficient algorithm due to the existence of multiple local optima, other heuristic methods, such as evolutionary computation, tabu search, and simulated annealing, may be applied. Finally, the proposed post-hoc tuning approach could be compared, both analytically and empirically, with methods that explicitly incorporate cost functions during model building. In this paper, we have focused on extending the BSZ method, which is the only post-hoc tuning method for cost-sensitive regression available in the literature. We have demonstrated that the extended tuning method outperforms the BSZ method.

Comparing the two general approaches for cost-sensitive regression, i.e., post-hoc tuning and direct cost-sensitive model building, demands extensive research, which we leave for the future.

### Tables

#### Table 7

<table>
<thead>
<tr>
<th>Cost Function</th>
<th>Method</th>
<th>Training</th>
<th>Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinLin</td>
<td>LR</td>
<td>0.49</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>M5</td>
<td>16.56</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>45.48</td>
<td>0.20</td>
</tr>
<tr>
<td>QuadQuad</td>
<td>LR</td>
<td>0.16</td>
<td>1.49</td>
</tr>
<tr>
<td></td>
<td>M5</td>
<td>0.20</td>
<td>1.68</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>0.20</td>
<td>1.70</td>
</tr>
</tbody>
</table>

#### References


