Error-Sensitive Grading for Model Combination

Surendra K. Singhi and Huan Liu

Department of Computer Science and Engineering,
Arizona State University, Tempe, AZ 85287-8809, USA
surendra@asu.edu, hliu@asu.edu

Abstract. Ensemble learning is a powerful learning approach that combines multiple classifiers to improve prediction accuracy. An important decision while using an ensemble of classifiers is to decide upon a way of combining the prediction of its base classifiers. In this paper, we introduce a novel grading-based algorithm for model combination, which uses cost-sensitive learning in building a meta-learner. This method distinguishes between the grading error of classifying an incorrect prediction as correct, and the other-way-round, and tries to assign appropriate costs to the two types of error in order to improve performance. We study issues in error-sensitive grading, and then with extensive experiments show the empirically effectiveness of this new method in comparison with representative meta-classification techniques.

1 Introduction

The accessibility and abundance of data in today's information age and the advent of multimedia and Internet have made machine learning an indispensable tool for knowledge discovery. Ensemble learning is a powerful and widely used technique which combine the decision of a set of classifiers to make the final prediction, this not only help in reducing the variance of learning, but also facilitates learning concepts (or hypothesis) from training data which are difficult for a single classifier. In large datasets, where there may be multiple functions defining the relationship between the predictor and response variables, ensemble methods allow different classifiers to represent each function individually instead of using one single overly complex function to approximate all the functions.

Building a good quality ensemble is a two steps process. During the first step (model generation phase), the constituent (or base level) classifiers should be selected such that they make independent or uncorrelated errors, or in other words, ensemble should be as diverse as possible. One way of introducing diversity is by varying the bias of learning, i.e., by employing different learning algorithms (results in heterogeneous ensemble); another technique is to keep the learning algorithm same, but manipulate the training data, so that the classifiers learn different functions in the hypothesis space (results in homogeneous ensemble). After an ensemble of classifiers is obtained, the next important step is to construct a meta classifier, which combines the predictions of the base classifiers (or model combination phase). This is the main focus of this paper.

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 724–732, 2005.

Different model combination techniques, depending upon the methods used by them can be partitioned into three categories i.e., voting, stacking and grading. The nomenclature for these categories was decided based on the most basic methods which represent the underlying principle of the methods falling under that category.

Voting. The techniques in this category are very simple, and widely used with homogeneous ensembles. *Majority voting* is a naive voting technique, in which a simple summation of the output probabilities (or 0, 1 values) of base classifiers is done, and a normalized probability distribution is returned. *Weighted Voting*, is a variation in which, a reliability weight or confidence value inversely proportional to the validation-set error rate, is assigned to each classifier. The meta-classifier then does a weighted sum to arrive at the final class probabilities. In one possible variation, instead of assigning a single reliability weight to the base classifier, for each class a separate reliability weight can be assigned.

Stacking. The stacking techniques are based on the idea of stacked generalization [1]. The distinguishing feature of the stacking techniques is that, the meta-classifier tries to learn the pattern or relationship between the predictions of the base classifiers and the actual class. Stacking with *Multi-response Linear Regression (MLR)* [2], is a stacking technique in which the MLR algorithm is used as the meta-classifier algorithm. Based on probability estimates given by the base-classifiers, meta-training datasets are constructed for **each** class. Then from these meta-training datasets linear regression models are built, the number of linear regression models is same as the number of classes. Dzeroski [3] shows that using Model Tree instead of Multi-response Linear Regression may yield better result. *StackingC* [4] is a variation, in which while building the meta-training datasets, instead of using class probabilities given by the base classifiers for all the different classes; only class probabilities corresponding to the particular class for which regression model is being built, are used. This results in faster model building time for the meta-classifier and also has the added benefit of the giving more diverse models for each classifier.

Grading. The defining feature of methods in this category (also known as referee method [5,6]) is that, instead of directly finding the relationship between the predictions of the base classifier and the actual class (as in stacking); the meta-classifier grades the base-classifiers, and selects either a single or subset of base-classifier(s) which are likely to be correct for the given test instance. The intuition behind grading is that in large datasets where there may be multiple functions defining the relationship between predictor and response variables, it is important to choose the correct function for any given test instance. In stacking the meta-classifier uses the predictions of the base classifier to decide the way they (predictions) should be

Table 1. Grading meta-training dataset, for a dataset with m features and n instances

Attributes			*Graded*
A_1	....	A_m	Class
$x_{1,1}$	...	$x_{1,m}$	1
$x_{2,1}$	...	$x_{2,m}$	1
...	...	...	...
$x_{n,1}$	...	$x_{n,m}$	0

combined to make the final decision; but in grading the test instance is used to decide which all base-classifiers, and with what reliability weight should they be used to make the final prediction.

In Grading, the meta-classifier itself is an ensemble of grader classifiers. Corresponding to each base level classifier there is a grader classifier which tries to learn its area of expertise or high predictive accuracies. For training the grader, as shown in Table 1 the original attributes are also used as the attributes for the grading dataset, but instead of using the original class attribute, a new graded class attribute with two possible values 1 (correct prediction) or 0 (incorrect prediction) is used. While making predictions the grader classifier assigns a weight to the base classifier's likelihood of being correct. This weight can be either absolute 1 or 0 score or it can also be a probability values. Only predictions from base classifiers, with reliability weight above a certain threshold or which are more likely to be correct than incorrect are taken; and then these predictions are combined using weighted voting to make the final prediction.

2 Error Sensitive Grading

We propose a new approach to doing Grading which combines cost-sensitive learning in assigning different costs to meta-training instances, and tries to make the graders conservative in assigning prediction tasks to the base classifiers. The intuition behind this method is to increase the prediction accuracy of each base classifier by using it only for instances for which it is very likely to be correct, but an immediate side-effect of this is that the number of instances for which the base classifier is used to make prediction decreases. In this work we study various research issues related to error-sensitive grading, including a new tie-breaking scheme designed for grading.

2.1 Cost-Sensitive Learning

In many machine learning domains, different misclassification incur different penalties and hence misclassification costs are different, given a test instance, cost-sensitive learning aims to predict the class that will lead to the lowest expected cost, where the expectation is computed using the conditional probability of each class and the misclassification cost. The most common method of achieving this objective is by re-balancing the training set given to the learning algorithm, i.e., to change the proportion of positive and negative training examples in the training set by over-sampling or under-sampling. An alternative, if the learning algorithm can use weights on training examples, is to set the weight of each example depending upon the cost.

2.2 Type A vs. Type B Errors

While grading the base classifiers, there could be two types of mistakes: when a base classifier predicts correctly, the grader says it is wrong (**Type A**); or when

a base classifier predicts wrongly, the grader considers it right (**Type B**). The issue is that if there are enough good classifiers then it doesn't hurt to leave one out. Yet, including a classifier when it is bad can really hurt performance. So, it is important to differentiate the two types of errors: the cost of Type B errors should be far higher than that of Type A errors. In other words, the cost of classifying an incorrect prediction as a correct prediction should be higher than the cost of classifying a correct prediction as incorrect. A base classifier generally predicts a high percentage of validation instances as correct, and so the majority of instances in the meta-training dataset for the graders are correct, and as a result the grader assigns a lower cost to Type A errors compared to the Type B error. This can lead to poor performance by graders, and hence by the meta-classifier and the ensemble.

2.3 Error Sensitive Grading Algorithm

The balance between the different misclassification costs can be readjusted by explicitly using cost-sensitivity. We call this modified version of grading as Error Sensitive Grading (ESGrading). Assigning higher cost to wrong grading makes the graders conservative in their decision making, i.e., the grader will predict a base classifier to be correct only when it is extremely sure. But one immediate drawback of making the graders conservative is that none of the base classifiers may be selected to predict on a test instance, to avoid this limitation the ensemble should have a large pool of diverse classifiers, so that the graders choose at least one base classifier to make the prediction. While using error-sensitive grading an important parameter which has to be chosen is, the different misclassification costs, because it is this cost which determines how conservative the grader should be. As graders have to deal with binary classification problem, i.e., predict whether the base classifier is correct or incorrect, the misclassification cost of two types can be combined into a single cost-ratio, which we define below.

Definition 1 (Cost-ratio). *It is the ratio of cost of Type A error over cost of Type B error.*

A lower value of the cost-ratio means that there is a heavy penalty for predicting an incorrect base classifier as correct. A value of cost-ratio equal to 1 means that the cost of grader misclassifying a base-classifier, whether the base-classifier is correct or incorrect is equal (the method is then equivalent to normal Grading). A value of cost-ratio equal to 0 on the other hand, indicates that the base classifier should never misclassify an incorrect base-classifier as correct, that is the base-classifier should never be trusted. To prove our hypothesis we did an experiment of varying the cost-ratio and observing the error rate of Grading method (Figure 1). In this experiment we used 10 bagged decision trees (Weka J4.8, a JAVA port to C4.5 Release 8 [7]) classifiers at the base level and decision trees again as the grader. The figure confirms to intuition and shows how when

Algorithm 1. Error-Sensitive Grading algorithm

```
procedure ESGrading(baseClassifiers,validSet)
    costRatio ← FindCostRatio(baseClassifiers, validSet)
    for all classifier ∈ baseClassifiers do
        Grader ← BuildGrader(classifier, costRatio, validSet)
        Add Grader to the meta-classifiers
    end for
end procedure

procedure FindCost(baseClassifiers,validSet)
    for costRatio ← 0.0 to 1.0 Step δ do
        Find cross validation error on validation set using costRatio and the base classifiers
    end for
    return costRatio with minimum cross validation error
end procedure
```

the cost is close to 0, the graders will be conservative and none of the base-classifiers will be predicted to be correct, and this results in a higher error, as the default majority-voting tie-breaking will be used. As the cost is increased the error rate decreases till it reaches a low-point after which it again begins to increase as the graders becomes lax in their grading and more base-classifiers are predicted to be correct.

One way to determine this cost-ratio is using cross-validation. It helps dynamically adjust the cost depending upon the number and diversity of base classifiers in an ensemble. When there are a large number of diverse base classifiers, then the graders can afford to be more conservative in picking base classifiers for making prediction, as the probability that at least one correct base classifier will be picked is high. On the other hand, when the number of base classifiers is small, the graders should be comparatively lenient to avoid the possible scenarios in which no base classifiers is picked for model combination. Using cross validation to determine the cost helps in striking the balance between making the graders conservative or lenient.

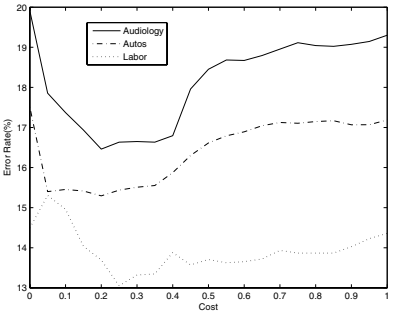

Fig. 1. Error rate of Error Sensitive Grading vs. the cost ratio

Algorithm 1 shows how to create an Error Sensitive Grading meta-classifier. The first step is to call the procedure *FindCostRatio* to decide the cost-ratio which should be used. The variable *validSet* is used to denote the validation set for building the meta-learner. The procedure *FindCostRatio* evaluates various cost-ratios and then uses cross validation to build graders via cost-sensitive learning and finally determines the error rate. The cost-ratio associated with the least error rate is returned and then used to build the final graders with the entire validation dataset.

2.4 Tie Breaking for Grading

In the Grading method as proposed by [6], when there is a tie in the likelihood of an instance belonging to different classes, the meta-classifier checks which of the class has higher prior probability and accordingly makes a decision. We suggest an alternative scheme in which instead of completely ignoring the predictions of some of the base classifiers (the classifiers with higher probability of being wrong than correct), the grader should assign a delta (close to zero) probability to all such classifiers being correct. This will ensure that under normal circumstances the meta-classifier only uses the prediction of the graders which are correct, but when there is a tie, majority voting is used.

In our experiments (not shown here due to space limitation), we observed that in general the ties are so rare that this tie-breaking scheme does not make any difference for the normal Grading algorithm. But in Error Sensitive Grading when the number of base classifiers is small, and the cost-ratio is close to 0, the graders may assign higher probability to all the base classifiers of being wrong than correct, and then none of the base classifiers will be selected to make the final prediction. In such, a scenario the above tie-breaking scheme is a better alternative (than the current prior probability method of breaking ties), because it makes use of majority voting to break the ties.

2.5 Time Complexity of Error-Sensitive Grading

The time complexity of Error-Sensitive Grading algorithm depends upon the method used for setting the cost-ratio. When c-fold cross-validation is used to determine cost-ratio, and t different cost-ratios are tried, then the time complexity is $O(c * t * G)$ where G is the time complexity of the Grading method. Grading is a time-consuming algorithm, but because learning is done offline, generally time is not a big issue for building classifiers.

3 Experiments and Discussion

For empirical evaluation we chose nineteen datasets from the UCI Machine Learning Repository [8]. Following the research done in [9], for all the experiments the reported results are obtained by ten ten-fold stratified cross-validations and t-test is done with calibrated degrees of freedom equal to 10. The reported estimates are the average of the 100 runs and the values after the $\pm$ sign is the average standard deviation. Superscripts denote significance levels for the difference in accuracy between the Error Sensitive Grading and the corresponding algorithm, using a one-tailed paired t test: 1 is 0.01, 2 is 0.025, 3 is 0.05, 4 is 0.1 and 5 is above 0.1.

We decided to use bagging [10] to study the effect of error sensitive grading as it is a widely used ensemble method and easily allows us to adjust the number of base classifiers. We implemented the Error Sensitive Grading method within WEKA [11]. All other algorithms are available within WEKA, but we

Table 2. Error rate for Bagging with different model combination techniques

Dataset	*Maj. Voting*	*Stacking MT*	*StackingC*	*Grading*	*E. S. Grading*
AUDIOLOGY	19.59 ± 6.73[1]	20.40 ± 6.91[1]	17.53 ± 5.92[4]	19.36 ± 6.66[2]	18.36 ± 6.26
AUTOS	16.97 ± 5.81[2]	17.17±6.03[4]	15.95±5.68[5]	17.41±5.90[1]	16.14±5.51
CREDIT-A	13.78± 4.62[5]	14.85 ± 5.10 [3]	14.75 ± 5.00[2]	13.69 ± 4.59[2]	13.91 ± 4.66
CREDIT-G	26.73 ± 8.95[5]	28.86 ± 9.65[1]	28.06 ±9.41[1]	26.84 ± 8.99[5]	26.74±8.93
GLASS	26.32 ± 8.94[3]	28.23 ±9.84[4]	27.08 ± 9.23[5]	26.61 ± 9.09[4]	26.88 ± 9.22
HEART-C	21.22 ± 7.17[2]	22.58 ± 7.79[5]	22.42±7.61[5]	21.76 ± 7.40[3]	22.08 ± 7.51
HEPATITIS	18.25 ±6.13[5]	21.02 ±7.15[1]	18.80±6.38[5]	19.43±6.67[5]	18.97±6.63
HYPO	0.45 ± 0.16[5]	7.64 ±2.56 [1]	0.43 ± 0.15[5]	0.46 ±0.16[2]	0.43 ± 0.15
KR	0.63±0.22[5]	0.56 ± 0.22[5]	0.57 ± 0.23[5]	0.64 ± 0.23[4]	0.60 ± 0.22
LABOR	16.03 ± 5.95 [3]	15.77± 6.00[4]	14.23 ± 5.11[5]	13.8 ± 4.99[5]	14.2 ± 5.26
PRIMARY-TUMOR	56.49 ± 18.90[1]	58.59 ± 19.57[2]	58.94 ± 19.68[1]	57.58 ± 19.24 [5]	57.55 ± 19.24
SEGMENT	2.55± 0.88[3]	2.80±0.99[1]	2.44±0.87[5]	2.51±0.84[4]	2.45±0.83
SICK	1.16 ±0.39[1]	1.07±0.39[5]	1.07±0.38[5]	1.11 ± 0.38[2]	1.09 ±0.38
SONAR	21.78 ± 7.50[5]	25.76 ± 9.09[1]	22.43 ±7.92[2]	21.30 ± 7.37[5]	21.24 ± 7.33
SOYBEAN	7.44 ± 2.56[5]	7.01 ±2.43[5]	6.87±2.45[5]	7.61±2.64[2]	7.19±2.48
SPLICE	5.68 ±1.90[5]	6.27 ±2.10[1]	5.79±1.93[3]	5.87 ±1.96[1]	5.69 ±1.90
VEHICLE	25.59± 8.57 [5]	25.48 ± 8.58[5]	25.30 ± 8.47 [5]	25.48±8.55 [5]	25.36±8.50
VOTE	3.56±1.22[4]	3.67±1.51[5]	3.81±1.37[5]	3.63 ± 1.2[5]	3.67 ±1.29
VOWEL	10.06 ± 3.39[1]	13.90 ± 4.72[1]	10.15±3.43[1]	10.47 ± 3.60[1]	9.25 ± 3.17
Loss/Tie/Win	3/10/6	0/10/9	0/13/6	2/10/7	-

adapted them to be used with bagging. During the experiments, it was ensured that the meta-classifiers, do not rebuild the base level decision trees built during model generation phase, as it goes against the spirit of meta-classifier as a model-combination method. Moreover, this ensures that the comparison of strengths of different model combination techniques can be done with a lower Type I error, as all the techniques are being used to combine the same set of classifiers.

3.1 Against Different Model Combination Methods

In this experiment we compared Error-Sensitive Grading against different model-combination techniques such as Majority Voting, Stacking with Model Trees, StackingC with MLR, and Grading on the chosen datasets. We used 10 bagged decision trees as the base-classifiers, and for the two grading methods we used decision tree as the grader too. Table 2 shows the results of the experiment. The loss/tie/win row in the table summarizes the number of datasets in which error sensitive grading performed worse, at par, or better than other methods. This loss, tie and win was determined using significance levels of 1, 2, and 3. We can clearly observe from this comparison that Error-Sensitive Grading performs well in comparison with representative methods of model combination.

The reason for the robustness of Error-Sensitive Grading is that it uses cross validation to determine the cost such that in the worst case it will perform similar to Grading or Majority Voting. Depending upon the base classifiers cross validation attempts to adjust the cost such that the graders are selective in picking the base classifiers for predicting the class for a new instance. The reader might observe that Error Sensitive Grading has two losses compared to Grading,

the reason for this is that as the same data is used during model generation and then during the meta-classifier creation process, the resultant model sometimes over fits the data and hence does worse.

3.2 Performance with Different Base Classifiers

In this experiment (Table 3) we decided to observe the benefit of Error Sensitive Grading against traditional Grading and Majority Voting when different algorithms (decision tree, naive bayes and support vector machines) are used for building the base classifiers. The Grading meta-learner used for all these experiments was the same i.e., decision tree. Again, the loss/tie/win was determined by using significance levels of 1, 2, and 3 with t-test using the method described earlier. The results for decision tree are from the Table 2. From the table it is quite clear that Error-Sensitive Grading outperforms both Majority Voting, and Grading, across all kinds of base learners. This table shows the robustness and stability of the Error Sensitive Grading method when different types of base classifiers are used.

Table 3. Loss/Tie/Win for ES-Grading vs. Maj. Voting and normal Grading when different algorithms used for base learner.

Algorithm	*Maj. Voting*	*Grading*
Decision Tree	3/10/6	2/10/7
Naive Bayes	0/8/11	0/16/3
Support Vector	1/10/8	0/15/4

4 Conclusion and Further Work

In this paper, we proposed a new grading-based method for model combination called Error-Sensitive Grading which applies cost-sensitive learning to grading base classifiers, such that the grader classifiers are made conservative in selective base classifiers for making predictions. We also studied issues about Error-Sensitive Grading such as cost assignment via cross validation, and introduced a new tie-breaking scheme for grading. The experimental results show that Error-Sensitive Grading is very competitive against all types of model combination methods. Using cross-validation to determine the cost is just one possible way, in our future research we plan to study other alternatives to determine the cost.

References

1. Wolpert, D.H.: Stacked generalization. Neural Networks **5** (1992) 241–259
2. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research **10** (1999) 271–289
3. Dzeroski, S., Zenko, B.: Is combining classifiers better than selecting the best one? Machine Learning **54** (2004) 255–273
4. Seewald, A.K.: How to make stacking better and faster while taking care of an unknown weakness. In: Ninteenth International Conference on Machine Learning. (2002) 554–561
5. Ortega, J., Koppel, M., Argamon, S.: Arbitrating among competing classifiers using learned refrees. In: Knowledge and Information Systems. Volume 3. (2001)

6. Seewald, A.K., Furnkranz, J.: An evaluation of grading classifiers. In: Advances in Intelligent Data Analysis: 4th International Symposium, Springer-Verlag (2001)
7. Quinlan, J.: C4.5: Programs for machine learning. Morgan Kaufmann (1993)
8. Merz, C.J., Murphy, P.M.: UCI repository of machine learning databases (1998) http://www.ics.uci.edu/m̃learn/MLRepository.html.
9. Bouckaert, R.R.: Choosing between two learning algorithms based on calibrated tests. In: Twentieth International Conference on Machine Learning. (2003)
10. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
11. Witten, I., Frank, E.: Data Mining: Practical Machine Learning tools and techniques. 2nd edn. Morgan Kauffmann (2005)