Error Limiting Reductions Between Classification Tasks

Alina Beygelzimer BEYGEL@US.IBM.COM
IBM T. J. Watson Research Center, Hawthorne, NY 10532

Varsha Dani VARSHA@CS.UCHICAGO.EDU
Department of Computer Science, University of Chicago, Chicago, IL 60637

Tom Hayes HAYEST@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA 94720

John Langford JL@TTI-C.ORG
TTI-Chicago, Chicago, IL 60637

Bianca Zadrozny ZADROZNY@US.IBM.COM
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

Abstract

We introduce a reduction-based model for analyzing supervised learning tasks. We use this model to devise a new reduction from multi-class cost-sensitive classification to binary classification with the following guarantee: If the learned binary classifier has error rate at most ϵ then the cost-sensitive classifier has cost at most 2ϵ times the expected sum of costs of all possible lables. Since cost-sensitive classification can embed any bounded loss finite choice supervised learning task, this result shows that *any* such task can be solved using a binary classification oracle. Finally, we present experimental results showing that our new reduction outperforms existing algorithms for multi-class cost-sensitive learning.

1 Introduction

Supervised learning involves learning a function from supplied examples. There are many natural supervised learning tasks; classification alone comes in a number of different forms (e.g., binary, multi-class, cost-sensitive, importance weighted). Generally, there are two ways to approach a new task. The first is to solve it independently from other tasks. The second is to reduce it to a task that has already been thoroughly analyzed, automatically transferring existing theory and algorithms from well-understood tasks to new tasks. We investigate (and advocate) the latter approach.

Motivation. Informally, a reduction is a learning machine that solves one task using oracle (i.e., black-box) access to a solver for another task. Reductions present a powerful tool with a number of desirable properties:

1. If A can be reduced to B, then techniques for solving B can be used to solve A, transferring results from one learning task to another. This saves both research and development effort.
2. Reductions allow one to study an entire class of tasks by focusing on a single primitive (they also help to identify such primitives). Any progress on the primitive immediately implies progress for every task in the class.
3. Viewed as negative statements, reductions imply relative lower bounds stating that progress on one task cannot be achieved without making progress on some primitive.

There is an essential discrepancy between theory and practice of supervised learning. On one hand, it is impossible to prove, without making any assumptions, that we can learn to predict well. (For example, in PAC-learning we might assume that all examples are sampled independently, and the problem is predicting an unknown decision list.) On the other hand, learning algorithms are often successfully used in practice.

Reductions give us a way to cope with this discrepancy by making *relative* statements of the form "Good per-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

formance of the oracle on subproblems generated by the reduction translates into good performance on the original problem". Such statements can be made without any assumptions that cannot be verified or do not generally hold in practice. Thus reductions give us a tool for showing *relative* ability to learn (relative to basic, better understood primitives).

We propose an analysis framework allowing us to prove such error transformation guarantees. Attempts to understand this framework in terms of sample complexity based models of learning (Vapnik & Chevonenkis, 1971; Valiant, 1984) will fail; it is important to understand that this is a different model of analysis.

To test the model, we consider learning machines which have oracle access to a binary classification learner, and ask what can be done with such a machine. Our main technical contribution is Section 3 which presents a new reduction from cost-sensitive classification to binary classification and quantifies the way in which errors by the binary classification oracle induce errors in the original problem. Since cost-sensitive classification can (in some precise sense) express any bounded-loss finite-choice supervised learning task, the reduction shows that *any* such task can be solved using a binary classification oracle.

To give further evidence that the proposed framework allows for a tractable analysis, we show how to analyze several existing algorithms in this model (Section 6).

In addition to the above theoretical motivations, there is significant empirical evidence (see (Zadrozny et al., 2003), (Freund & Schapire, 1997), (Dietterich & Bakiri, 1995)) that this style of analysis often produces learning algorithms that perform well in practice. Section 3.2 presents experimental results showing that our new reduction outperforms existing algorithms for multi-class cost-sensitive learning.

2 Basic Definitions

This section defines the basic concepts.

Definition 1. *A* supervised learning task *is a tuple* (K, Y, ℓ)*, where* K *is the space of supervised advice available at training time,* Y *is a prediction space, and* $\ell : K \times Y \rightarrow [0, \infty)$ *is a loss function.*

We assume that ℓ is nonconstant in both arguments, which is satisfied by any reasonable loss function. To test the definition, we instantiate it for common classification tasks. For *binary classification*, $K = \{0, 1\}$ (the space of labels given for training examples), $Y = \{0, 1\}$, and $\ell(k, y) = I(k \neq y)$, where $I(\cdot)$ is the indicator function which is 1 when the argument is true and 0 otherwise. A simple generalization to more than two labels gives *multiclass classification* with $K = Y = \{1, \ldots, r\}$ and $\ell(k, y) = I(k \neq y)$.

In general, the space K of advice provided at training time need not equal the prediction space Y. We sometimes need this additional flexibility to provide some information to the learning algorithm about the loss of different choices. For example, in *importance weighted classification*, we want to specify that predicting some examples correctly is more important than predicting others, which is done by letting $K = \{1, \ldots, r\} \times [0, \infty)$ and $\ell(\langle k_1, k_2 \rangle, y) = k_2 I(k_1 \neq y)$. In *cost-sensitive classification*, K is used to associate a cost to each prediction $y \in Y$ by setting $K = [0, \infty)^r$ and $\ell(\langle k_1, \ldots, k_r \rangle, y) = k_y$. In both examples above, the prediction space Y is $\{1, \ldots, r\}$. Cost-sensitive classification is often useful in practice since each prediction may generally have a different associated loss, and the goal is to minimize the expected loss (instead of the number of misclassifications).

A task is (implicitly) a set of learning problems, each of which requires more information to be fully specified.

Definition 2. *A* supervised learning problem *is a tuple* $(D, X, \mathcal{T})$*, where* $\mathcal{T} = (K, Y, \ell)$ *is a supervised learning task,* X *is an arbitrary feature space, and* D *is a distribution over* $X \times K$*.*

The goal in solving a supervised learning problem is to find a hypothesis $h : X \rightarrow Y$ minimizing the expected loss $E_{(x,k)\sim D}\ell(k, h(x))$. Finding h is difficult because D is generally not known at training time. For any particular hypothesis h, the loss rate $h_D = E_{(x,k)\sim D}\ell(k, h(x))$ is a fundamental quantity of interest.

Definition 3. *A* supervised learning algorithm for task (K, Y, ℓ) *is a procedure mapping any finite set of examples in* $(X \times K)^*$ *to a hypothesis* $h : X \rightarrow Y$*.*

We use these definitions in Section 4 to define an error-limiting reductions model. Ultimately, the real test of a learning model is whether it allows for tractable analysis leading to useful learning algorithms. To motivate the formulations of the model, we first show an example of what can be done with this model. In particular, we present a new reduction from cost-sensitive classification to binary classification and prove its error-transformation bounds. To give further evidence, Section 6 shows that the model captures several existing machine learning algorithms.

3 Reduction from Cost Sensitive Classification to Binary Classification

We present a new reduction, Weighted All-Pairs (WAP), from r-class cost sensitive classification to importance weighted binary classification. It can be composed with the Costing reduction (Zadrozny et al., 2003) to yield a reduction to binary classification. Since cost-sensitive classification can express any supervised learning task we consider (using the mapping $K' = \ell(K, Y)$ for the task (K, Y, ℓ)), this implies that *any* such task can be solved using binary classification.

3.1 The Algorithm

WAP is a weighted version of the All-Pairs reduction (Hastie & Tibshirani, 1997) from multi-class to binary classification. The original reduction works by learning a classifier trained to predict, given a pair of classes (i, j) as a part of the feature space, "Is label i more probable than label j?". To make a multiclass prediction, the All-Pairs reduction runs the classifier for every pair of labels and outputs the label i maximizing the number of "yes" answers, with ties broken arbitrarily.

To solve cost-sensitive classification, WAP needs to account for the costs. The algorithms specifying the reduction are given below.

1 WAP-Train (Set of r-class cost sensitive examples S, importance weighed binary classifier learning algorithm B)

Set $S' = \emptyset$.
for all examples $(x, k_1, \ldots, k_r)$ in S **do**
  for all pairs (i, j) with $1 \leq i < j \leq r$ **do**
    Add an importance weighted example
    $(\langle x, i, j\rangle, I(k_i < k_j), |v_j - v_i|)$ to S'.
  end for
end for
Return $h = B(S')$.

The values v_i used by the reduction are defined as follows: For a given cost-sensitive example $(x, k_1, \ldots, k_r)$, let $L(t)$ be the function $L(t) = |\{j \mid k_j \leq t\}|$, for $t \in [0, \infty)$. By shifting, we may assume that the minimum cost is 0, so that $t \geq 0$ implies $L(t) \geq 1$. Optimizing the loss of the shifted problem is equivalent to optimizing the loss of the original problem. The values v_i are defined as $v_i = \int_0^{k_i} 1/L(t)dt$. Note that the values are order-preserving: $k_i < k_j$ iff $v_i < v_j$ for all i and j.

We say that label i *beats* label j for input x if either $i < j$ and $h(\langle x, i, j\rangle) = 1$, or $i > j$ and $h(\langle x, j, i\rangle) = 0$.

Note that if h makes no errors and $k_i \neq k_j$, then label i beats label j exactly when $k_i < k_j$. **WAP-Test** outputs the label which beats the maximum number of other labels, with ties broken arbitrarily.

2 WAP-Test (classifier h, example x)

for all pairs (i, j) with $1 \leq i < j \leq r$ **do**
  Evaluate $h(\langle x, i, j\rangle)$
end for
Output $\text{argmax}_i |\{j \mid i \text{ beats } j\}|$.

Before stating the error transform, we must first define the distribution **WAP-Train**(D) induced by the reduction on the importance-weighted binary classifier (to which the reduction makes a single call). To draw a sample from this distribution, we first draw a cost sensitive sample $(x, k_1, \ldots, k_r)$ from the input distribution D and then apply **WAP-Train** to the singleton set $\{(x, k_1, \ldots, k_r)\}$ to get a sequence of $\binom{r}{2}$ examples for the binary classifier. Now we just sample uniformly from this set. Recall that the loss rate of classifier h on distribution D is denoted by h_D.

Theorem 1. (WAP error efficiency) *For all importance weighted binary learning algorithms B and cost-sensitive datasets S, let h =* **WAP-Train**(S, B)*. Then for all cost-sensitive test distributions D,*

$$\textbf{WAP-Test}(h)_D \leq 2h_{\textbf{WAP-Train}(D)}.$$

This theorem states that the cost sensitive loss is bounded by twice the importance weighted loss on the induced importance weighted learning problem.

Theorem 1 and the Costing reduction (Theorem 4 in Section 6.1) imply the following efficiency guarantees:

Theorem 2. *For all cost sensitive test distributions D, let D' be the binary classification problem induced by* **WAP-Train** *and Costing, and h be the learned binary classifier. Then*

$$\textbf{WAP-Test}(h)_D \leq 2h_{D'} E_{\langle x,k_1,\ldots,k_r\rangle \sim D} \sum_{i=1}^{r} k_i.$$

For notational simplicity, assume that $k_1 \leq \cdots \leq k_r$. Note that no generality is lost since the algorithm does not distinguish between the labels. The following insight is the key to the proof.

Lemma 1. *Suppose label i is the winner. Then, for every $j \in \{1, \ldots, i-1\}$, there must be at least $\lceil j/2 \rceil$ pairs (a, b), where $a \leq j < b$, and b beats a.*

Proof. Consider the restricted tournament on $\{1, \ldots, j\}$.

Case 1: Suppose that some w beats at least $\lceil j/2 \rceil$ of the others. If no label $b > j$ beats any label $a \leq j$, then w would beat at least $\lceil j/2 \rceil + 1$ more labels than any $b > j$; in particular, w would beat at least $\lceil j/2 \rceil + 1$ more labels than i. Thus, in order to have label i beat as many labels as w, at least $\lceil j/2 \rceil$ edges of the form $(w, b), b > j$ or $(a, i), a \leq j$ must be reversed.

Case 2: There is no label $w \in \{1, \ldots, j\}$ beating $\lceil j/2 \rceil$ of the rest of $\{1, \ldots, j\}$. This can only happen if j is odd and there is a j-way tie with $(j-1)/2$ losses per label in $\{1, \ldots, j\}$. In this case, although every label beats $(j+1)/2$ more labels than any $b > j$, in particular i, it is still necessary to reverse at least $(j+1)/2 \geq \lceil j/2 \rceil$ edges, in order to ensure that $i > j$ beats as many labels as each of $\{1, \ldots, j\}$. □

Proof. (Theorem 1) Suppose that our algorithm chooses the wrong label i for a particular example $(x, k_1, \ldots, k_r)$. We show that this requires the adversary to incur a comparable loss.

Lemma 1 and the definition of v_i imply that the penalty incurred to make label i win is at least

$$\int_0^{k_i} \frac{\lceil L(t)/2 \rceil}{L(t)} \mathrm{d}t \geq \int_0^{k_i} \frac{1}{2} \mathrm{d}t = \frac{k_i}{2}.$$

On the other hand, the total importance assigned to queries for this instance equals

$$\sum_{i<j} v_j - v_i = \sum_{i<j} \int_{k_i}^{k_j} \frac{1}{L(t)} \mathrm{d}t = \int_0^{k_r} \frac{L(t)R(t)}{L(t)} \mathrm{d}t$$
$$= \int_0^{k_r} R(t) \mathrm{d}t = \sum_{i=1}^{r} \int_0^{k_i} \mathrm{d}t = \sum_{i=1}^{r} k_i,$$

where $R(t) = r - L(t)$ is the number of labels whose value is greater than t and the second equality follows from switching the order of summation and counting the number of times a pair (i, j) satisfies $i < t < j$. The second to last equality follows by writing $R(t)$ as a sum of the r indicator functions for the events $\{k_j > t\}$, and then switching the order of summation.

Consequently, for every example $(x, k_1, \ldots, k_r)$, the total importance assigned to queries for x equals $\sum_i k_i$, and the cost incurred by our algorithm on instance x is at most twice the importance of errors made by the binary classifier on instance x. Averaging over the choice of x shows that the cost is at most 2. □

This method of assigning importances is provably near-optimal.

Theorem 3. (Lower bound) *For any other assignments of importances $w_{i,j}$ to the points (x, i, j) in the above algorithm, there exists a distribution with expected cost $\epsilon/2$.*

Proof. Consider examples $\left(x, 0, \frac{1}{r-1}, \ldots, \frac{1}{r-1}\right)$. Suppose that we run our algorithm using some $w_{i,j}$ as the importance for the query (x, i, j). Any classifier which errs on $(x, 1, i)$ and $(x, 1, j)$, where $i \neq j$, causes our algorithm to choose label 2 as the winner, thereby giving a cost of $1/(r-1)$, out of the total cost of 1. The importance of these two errors is $w_{1,i} + w_{1,j}$, out of the total importance of $\sum_{i,j} w_{i,j}$. Choosing i and j so that $w_{1,i} + w_{1,j}$ is minimal, the adversary's penalty is at most $2 \sum_{i=2}^{r} w_{1,i}/(r-1)$, and hence less than $2/(r-1)$ times the total importance for x. This shows that the cost of the reduction cannot be reduced below 1/2 merely by improving the choice of weights. □

3.2 Experiments

We conducted experiments comparing WAP to the original all-pairs reduction (Hastie & Tibshirani, 1997) and to state-of-the-art multi-class cost-sensitive learning algorithms. Using the C4.5 decision tree learner (Quinlan, 1993) as oracle classifier learner, we compared our reduction to the cost-sensitive learning algorithm MetaCost (Domingos, 1999). Using Boosted Naive Bayes (Elkan, 1997), we compared our reduction to the minimum expected cost approach after calibration of the probability estimates produced by the learner, as done by Zadrozny and Elkan (Zadrozny & Elkan, 2002). We applied the methods to seven UCI multiclass datasets. Since these datasets do not have costs associated with them, we generated artificial costs in the same manner that was done in the MetaCost paper (except that we fixed the cost of classifying correctly at zero). We repeated the experiments for 20 different settings of the costs. The average and standard error of the test set costs obtained with each method are shown in Tables 1 and 2.

We present results for two versions of this reduction. One is the exact version used in the proof above. In the other version, we learn one classifier per pair as is done in the original all-pairs reduction. More specifically, for each i and j (with $i < j$) we learn a classifier using the following mapping to generate training examples: $(x, k_1, \ldots, k_r) \mapsto (x, I(k_i < k_j), |v_i - v_j|)$.

In the results, we see that the first version appears to work well for some problems and badly for others, while the second version is more stable. Perhaps this is because the training examples given to the learner in the first version are not drawn i.i.d. from a fixed distribution as learning algorithms often expect.

4 Error-Limiting Reductions

Informally, a reduction R from task $\mathcal{T} = (K, Y, l)$ to another task $\mathcal{T}' = (K', Y', l')$ is a procedure that uses oracle access to a learning algorithm for $\mathcal{T}'$ to solve $\mathcal{T}$, and guarantees that good performance of the oracle on subproblems generated by the reduction translates

Dataset	All-Pairs	MinExpCost	WAP v.1	WAP v.2
anneal	310 ± 205	94.6 ± 43	29.9 ± 3.0	164 ± 43
ecoli	538 ± 126	1013 ± 202	52.4 ± 6.3	183 ± 56
glass	838 ± 72	763 ± 99	245 ± 21	363±21
satellite	137 ± 33	107 ± 6.6	428 ± 21	148 ± 8.5
splice	59.8 ± 24	36.2 ± 2.4	197 ± 23	46.5 ± 3.5
solar	3989 ± 1415	36.4 ± 9.0	24.7 ± 2.0	36.8 ± 11
yeast	931±46	148±31.8	55.0±4.7	58.3±7.2

Table 1. Experimental results with Boosted Naive Bayes.

Dataset	All-Pairs	MetaCost	WAP v.1	WAP v.2
anneal	807 ± 105	15.3 ± 2.0	94.9 ± 7.6	25.4 ± 2.8
ecoli	588 ± 102	137 ± 7.2	614 ± 43	88.1 ± 5.9
glass	656 ± 53	197 ± 20	722 ± 58	157±8.8
satellite	196 ± 9.8	138 ± 7.2	614 ± 43	88.1 ± 5.9
splice	51.3 ± 2.9	49.7 ± 3.7	181 ± 51	46.3 ± 4.2
solar	5036 ± 246	13.2 ± 0.76	13.2 ± 0.79	12.5 ± 0.66
yeast	981±62	35.6±3.1	30.3±1.4	29.7±1.3

Table 2. Experimental results with C4.5.

into good performance on the original problem.

To formalize this statement, we need to define a mapping from a measure $\mathcal{D}$ over prediction problems in $(X \times K)^*$ to a measure $\mathcal{D}'$ over generated subproblems in $(X' \times K')^*$. In order to call the oracle, the reduction must take as input some training set S from $(X \times K)^*$ and create some S' from $(X' \times K')^*$. We can think of this process as inducing a mapping from $\mathcal{D}$ to $\mathcal{D}'$ according to $\mathcal{D}'(S') = E_{S\sim\mathcal{D}}\Pr_R(S')$, where the probability of generating S' is over the random bits used in the reduction on input S.

This rule defines $\mathcal{D}'$ for a single invocation of the oracle. If the reduction makes several sequential calls, we need to define $\mathcal{D}'$ for each invocation. Suppose that the first invocation produces h'. We replace this invocation with the oracle that always returns h', and use the above definition to find $\mathcal{D}'$ for the next invocation.

Now, suppose that we have a test example (x, k) drawn from some arbitrary distribution D over $X \times K$. (We can think of D as being a distribution over $(X \times K)^*$ having all probability mass on one-element sequences of examples.) The reduction maps this to a new measure D' using the argument above.

We can now state the formal definition.

Definition 4. *An* error-limiting reduction $R(S, A)$ from task (K, Y, ℓ) to task (K', Y', ℓ') *takes as input a finite set of examples $S \in (X \times K)^*$ and a learning algorithm A for task (K', Y', ℓ'), and outputs a hypothesis $h : X \to Y$ that uses a collection of sub-hypotheses returned by A. For every such sub-hypothesis h', and every measure D over $X \times K$, the reduction induces a measure D' over the inputs to h', as described above. For all $X \times K$ and D, the reduction must satisfy*

$$h_D \le g(\max_{h',D'} h'_{D'})$$

for some continuous monotone nondecreasing function $g : [0, \infty) \to [0, \infty)$ with $g(0) = 0$. We call g the error-limitation function *of the reduction.*

For this definition, we use the convention that $\max_\emptyset = 0$, where $\max_\emptyset$ is the maximum over the empty set. Note again that none of the above definitions require that examples be drawn i.i.d. Analysis in this model is therefore directly applicable to all real-world problems which satisfy the type constraint.

Notice that we do not stipulate that the training set and the test example (x, k) come from the same distribution. In fact, the definition is only dependent on the training distribution through the set of training examples S used to construct examples for the oracle.

Perspective It is useful to consider an analogy with Turing reductions in complexity theory. In both cases, a reduction from task A to task B is a procedure that solves A using a solver for B as a black box. There are several differences, the most important being that complexity-theoretic reductions are often viewed as negative statements providing evidence that a problem is intractable. Here we view reductions as positive statements allowing us to solve new problems.

Metrics of success We consider several ways to measure reductions. The metric essential to the definition of an error-limiting reduction is the *error efficiency* of the reduction *for a given example*, defined as the maximum ratio of the loss of the hypothesis output by the reduction on this example and the total loss of the oracle hypotheses on the examples generated by the reduction.

We also want the transformation to be computationally efficient assuming that all oracle calls take unit time. Calls can be either adaptive (i.e., depend on the answers to previous queries) or parallel (i.e., all queries can be asked before any of them are answered). It is easy to see that there is a general transformation turning parallel calls into a single call to the oracle learning algorithm. To make one call, we can augment the feature space with the name of the call and then learn a classifier on the union of all training data. We used this trick in the WAP reduction in Section 3.

5 The Structure of Error Limiting Reductions

Non-triviality of the definition The proposition below states that every reduction must make a nontrivial use of its oracle.

Proposition 5.1. *For every error-limiting reduction R, every training set S, and test example (x, k), there exists an oracle A such that*

$$\mathbf{E}_R[h(x)] = \arg\min_{y \in Y} \ell(k, y) = 0,$$

where h is the hypothesis output by $R(S, A)$, and the expectation is over the randomization used by R in forming h.

Proof. Recall that the error limitation property holds for all D, in particular for the deterministic D with all probability mass on (x, k). Thus if R just ignored its oracle we would have $\ell(k, h(x)) = g(\max_{\emptyset}) = g(0) = 0$. The fact that $\ell(k, y)$ is nonconstant (in both arguments) implies that there exists (x, k) such that any h must satisfy $\ell(k, h(x)) > 0$, a contradiction. Consequently, at least one call to the oracle must be made.

Next note that for all R, S, and (x, k), there exists an h' (and thus an oracle A outputting h') consistent with all examples induced by R on (x, k). If R is randomized, there exists a deterministic reduction R' taking one additional argument (a randomly chosen string) with the same output behavior as R. Consequently, there exists an oracle A dependent on this random string which achieves error rate 0. Since $g(0) = 0$, this must imply that $h(x) = \arg\min_{y \in Y} \ell(k, y) = 0$. □

Note, however, that while the notion of error-limiting reductions is non-trivial, it does not prevent a reduction from creating hard problems for the oracle.

Composition Since our motivation for studying reductions is to reduce the number of distinct problems which must be considered, it is natural to want to compose two or more reductions to obtain new ones. It is easy to see that reductions defined above compose.

Proposition 5.2. *If R_{12} is a reduction from task $\mathcal{T}_1$ to $\mathcal{T}_2$ with error limitation g_{12} and R_{23} is a reduction from task $\mathcal{T}_2$ to $\mathcal{T}_3$ with error limitation g_{23}, then the composition $R_{12} \circ R_{23}$ is a reduction from $\mathcal{T}_1$ to $\mathcal{T}_3$ with error limitation $g_{12} \circ g_{23}$.*

Proof. Given oracle access to any learning algorithm A for $\mathcal{T}_3$, reduction R_{23} yields a learning algorithm for $\mathcal{T}_2$ such that for any X_2 and D_2, it produces a hypothesis with loss rate $h'_{D_2} \leq g_{23}(h''_{D_3})$, where h''_{D_3} is the maximum loss rate of a hypothesis h'' output by A (over all invocations). We also know that R_{12} yields a learning algorithm for $\mathcal{T}_1$ such that for all X_1 and D_1 we have $h_{D_1} \leq g_{12}(h'_{D_2}) \leq g_{12}(g_{23}(h''_{D_3}))$, where h'_{D_2} is the maximum loss rate of a hypothesis h' output by $R_{23}(A)$. Finally, notice that $g_{12} \circ g_{23}$ is continuous, monotone nondecreasing, and $g_{12}(g_{23}(0)) = 0$. □

6 Examples of Reductions

Here, we show that the notion of an error-limiting reduction is both tractable for analysis and describes several standard machine learning algorithms.

6.1 Reduction from Importance Weighted Classification to Classification

In importance weighted classification, every example comes with an extra "importance" value which measures how (relatively) important a decision might be. This extra flexibility turns out to be useful in several applications, and as an intermediate step in several reductions.

The "Costing" (Zadrozny et al., 2003) algorithm is a (very simple) reduction from importance weighted classification to classification based on rejection sampling. Given an importance weighted dataset $S = (x, k_1, k_2)$ an unweighted dataset is produced using rejection sampling:

$$\{(x, k_1) : (x, k_1, k_2) \in S \text{ and } k_2 < t \sim U(0, w)\}$$

where $t \sim U(0, w)$ is a uniform random draw from the interval $[0, w]$ and w is greater than k_2 for all examples. The classifier learner is applied to produce a classifier c for this new dataset.[1]

[1]The actual costing algorithm repeats this process and predicts according to the majority of learned c.

Although this algorithm is remarkably simple, composing costing with decision trees has been shown to yield superior performance on benchmark prediction problems (Zadrozny et al., 2003).

Theorem 4. *(Costing error efficiency) For all importance weighted problems $(D, X, \mathcal{T})$, if the classifiers have error rate ϵ, costing has loss rate at most*

$$\epsilon E_{(x,k_1,k_2)\sim D} k_2.$$

Proof. Trivial. See (Zadrozny et al., 2003). □

6.2 Reductions from Multiclass Classification to Binary Classification

In this section we analyze several well known reductions for efficiency. Before we start, it is important to note that there are boosting algorithms for solving multiclass classification given weak binary importance weighted classification (Schapire, 1997), (Schapire & Singer, 1999). Some of these methods require extra powerful classification algorithms, but others are genuine reductions, as defined here. A reductionist approach for creating a multiclass boosting algorithm is to simply compose any one of the following reductions with binary boosting.

The Tree Reduction In the tree reduction (which is well known, see chapter 15 of (Fox, 1997)), we construct $r-1$ binary classifiers distinguishing between the labels using a binary tree. The tree has depth $\log_2 r$, and each internal node is associated with a set of labels that can reach the node, assuming that each classifier above it predicts correctly. The set of labels at that node is split in half, and a classifier is trained to distinguish between the two sets of labels. The root node starts with the set of all labels $\{1, \ldots, r\}$ which are all indistinguishable at that point. Predictions are made by following a chain of classifications from the root down to the leaves, each of which is associated with a unique label. Note that instead of $r-1$ parallel, one could use $\log_2 r$ adaptive queries.

Theorem 5. (*Tree error efficiency*) *For all multiclass problems $(D, X, \mathcal{T})$, if the binary classifiers have loss rate ϵ, the tree reduction has loss rate at most $\epsilon \log_2 r$.*

Proof. A multiclass error occurs only if some binary classifier on the path from the root to the leaf errs. □

The Error Correcting Output Codes (ECOC) Reduction Let $C \subseteq \{0,1\}^n$ be an error-correcting code with l codewords a minimum distance of d apart. Let $C_{i,j}$ denote the i'th bit of the j'th codeword of C. The ECOC reduction (Dietterich & Bakiri, 1995) corresponding to C learns n classifiers: the i'th classifier predicts $C_{i,y}$ where y is the correct label given input x. The loss rate for this reduction is at most $2n\epsilon/d$, as we shall prove in Theorem 6 below.

We mention three codes of particular interest. The first is when the codewords form a subset of the rows of a Hadamard matrix (an $n \times n$ binary matrix with any two rows differing in exactly $n/2$ places). Such matrices exist and are easy to construct when n is a power of 2. Thus, for Hadamard codes, the number of classifiers needed is less than $2r$ (since a power of 2 exists between r and $2r$), and the loss rate is at most 4ϵ. When the codewords correspond to binary representations of $0, 1, \ldots, r-1$, the ECOC reduction is similar to (although not the same due to decoding differences) the tree reduction above. Finally, if the codewords form the $r \times r$ identity matrix, the ECOC reduction corresponds to the one-against-all reduction.

Theorem 6. (*ECOC*) *For all multiclass problems $(D, X, \mathcal{T})$, if the binary classifiers have loss rate ϵ, the ECOC reduction has loss rate less than $\frac{2n}{d}\epsilon$.*
This theorem is identical to Lemma 2 of (Guruswami & Sahai, 1999) (which was generalized in (Allwein et al., 2001)), except that we generalize the statement to any distribution D generating (test) examples rather than the loss rate on a training set.

Proof. When the loss rate is zero, the correct label is consistent with all n classifiers, and wrong labels are consistent with at most $n-d$ classifiers, since C has minimum distance d. In order to wrongly predict the multiclass label, at least $d/2$ binary classifiers must err simultaneously. Since every multiclass error implies at least a fraction $d/2n$ of the binary classifiers erred, a binary loss rate of ϵ can yield at most a multiclass loss rate of $2n\epsilon/d$. □

Sometimes ECOC is analyzed assuming the errors are independent. This gives much better results, but seems less justifiable in practice. For example, in any setting with label noise errors are highly *dependent*.

We note that all reductions in this section preserve independence. Many common learning algorithms are built under the assumption that examples are drawn independently from some distribution, and it is a desirable property for reductions to preserve this independence when constructing datasets for the oracle.

7 Prior Work

There has been much work on reductions in learning. One of our goals is to give a unifying framework, allowing a better understanding of strengths and weaknesses in forms of reductions.

Boosting algorithms (Freund & Schapire, 1997; Kalai & Servedio, 2003) can be thought of as reductions from binary (or sometimes multiclass) classification with a small error rate to importance weighted classification with a loss rate of nearly $\frac{1}{2}$. Given this, it is important to ask: "Can't a learning machine apply boosting to solve any classification problem perfectly?" The answer is "No" since it is impossible to always predict correctly whenever the correct prediction given features is not deterministic. As a result, the loss rate becomes unbounded from $\frac{1}{2}$ after some number of rounds of boosting.

Bagging (Breiman, 1996) can be seen as a self-reduction of classification to classification, which turns learning algorithms with high "variance" (i.e., dependence on the exact examples seen) into a classifier with lower "variance". Bagging performance sometimes suffers because the examples fed into the classifier learning algorithm are not necessarily independently distributed according to the original distribution.

Pitt and Warmuth (Pitt & Warmuth, 1990) also introduced a form of a reduction between classification problems called *prediction-preserving reducibility*. There are several important differences between this notion and the notion presented here: (1) prediction-preserving reductions are typically used to show negative, *non*-predictability results, while we show positive results transferring learning algorithms between tasks; (2) reductions presented here are representation independent; (3) prediction-preserving reducibility is a *computational* notion, while the reductions here are *informational*.

It should be noted that not all reductions in the learning literature are black box. For example, Allwein et al. Allwein et al. (2001) give a reduction from multiclass to binary classification that uses a "margin," which is a heuristic form of prediction confidence that can be extracted from many (but not all) classifiers.

8 Conclusion

We introduced the analysis model of error-limiting reductions in learning. This model has several nice properties (often not shared with other methods of learning analysis). First, the analysis in this model does not depend upon *any* unprovable (or unobservable) assumptions such as independence. Error-limiting reductions often have tractable analysis and capture several common machine learning algorithms. They also often work well in practice.

To test the model, we constructed, proved, and empirically tested an error-limiting reduction from multiclass cost-sensitive classification to binary classification. Since cost sensitive classification can express any supervised learning task considered here, this reduction can be used to reduce *any* such task to binary classification.

References

Allwein, E., Schapire, R., & Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *J. of Machine Learning Research*, *1*, 113–141.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *26*, 123–140.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. *Proceedings of the 5th KDD Conference* (pp. 155–164).

Elkan, C. (1997). *Boosting and naive bayesian learning* (Technical Report CS97-557). UC San Diego.

Fox, J. (1997). *Applied regression analysis, linear models, and related methods*. Sage Publications.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and Sys. Sci.*, *55*, 119–139.

Guruswami, V., & Sahai, A. (1999). Multiclass learning, boosting, and error-correcting codes. *Proceedings of the 12th Annual Conference on Computational Learning Theory (COLT)* (pp. 145–155).

Hastie, T., & Tibshirani, R. (1997). Classification by pairwise coupling. *Advances in Neural Information Processing Systems 10 (NIPS)* (pp. 507–513).

Kalai, A., & Servedio, R. (2003). Boosting in the presence of noise. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)* (pp. 195–205).

Pitt, L., & Warmuth, M. (1990). Prediction-preserving reducibility. *J. of Comp. and Sys. Sci.*, *41*, 430–467.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Schapire, R. (1997). Using output codes to boost multiclass learning problems. *Proceedings of the 14th International Conference on Machine Learning (ICML)* (pp. 313–321).

Schapire, R., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, *37*, 297–336.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, *27*, 1134–1142.

Vapnik, V. N., & Chevonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of probability and its applications*, *16*, 264–280.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD)* (pp. 694–699).

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