Error Limiting Reductions Between Classification Tasks

Alina Beygelzimer[1], Varsha Dani[2], Tom Hayes[3], John Langford[3], and Bianca Zadrozny[1]

[1] IBM TJ Watson
[2] University of Chicago
[3] TTI-Chicago

Abstract. Supervised learning involves learning a function from supplied examples. There are two approaches to solving a new supervised learning task: either analyze the task independently or reduce it to a task that has already been thoroughly analyzed. This paper investigates the latter approach for classification-type problems. In addition to theoretical motivations, there is fairly strong empirical evidence that this style of analysis often produces learning algorithms that perform well in practice. We present a theoretical framework for analyzing reductions and discuss several known reductions from this standpoint. We also present, analyze, and empirically test a reduction from cost-sensitive classification to binary classification. Since every supervised learning task can (in a precise sense) be embedded in cost sensitive classification, this shows that any such task can be solved by binary classification algorithms.

1 Introduction

Supervised learning involves learning a function from supplied examples. There are many natural supervised learning tasks – even classification comes in a variety of different forms (e.g., binary, multi-class, cost-sensitive, importance weighted). There are two approaches that one can take. The first is to analyze all tasks independently and devise algorithms for each. The other approach is to use reductions to convert new, unexplored tasks into tasks that have been thoroughly analyzed, automatically transferring existing theory and algorithms from well-understood tasks into new tasks. We investigate the latter approach. Reductions present a powerful tool with a number of desirable properties:

1. They provide a natural way of comparing how easy different tasks are. If task A reduces to task B, it is reasonable to say that A is not much harder than B. If two tasks are reducible to each other, we can reasonably conclude that they are similar. The more restricted the notion of a reduction, the stronger the statements.
2. If task A can be reduced to task 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.
3. Reductions allow one to study an entire class of tasks by focusing on a single primitive. Any progress on the primitive immediately implies progress for every task in the class. It also simplifies the evaluation of different learning algorithms for effectiveness.
4. 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 difficulty in supervised learning tasks. On one hand, it is impossible to state that we can learn to predict well without making any assumptions. (For example, PAC-learning assumes that "the examples are sampled independently and the problem is to predict according to an unknown DNF".) On the other hand, learning algorithms are succesfully used in practice.

Reductions give us a way to cope with this difficulty. We can view a learning machine as a Turing machine, which has oracle access to a solver for some classification problem, and ask what learning problems are solvable with such a machine. This paper shows that all supervised learning problems (with finite prediction spaces or finitely-approximable prediction spaces) are solvable, and quantifies the way in which errors by the oracle induce errors in the supervised learning problem.

In addition to the above theoretical motivations, there appears to be significant empirical evidence (see [19], [6], [4]) that this style of analysis often produces learning algorithms that perform well in practice.

This paper makes the following (definitional and constructive) contributions.

1. We define a class of supervised learning tasks, and present a new, reduction-based model[4] for the analysis of such tasks. We believe that this is the "right" model based upon simplicity, empirical evidence, tractability of analysis in this model, and the fact that it captures the way learning tasks are divided up by practitioners. Furthermore, the dependencies of this model are fully observable. In particular, we need no assumption that examples are i.i.d. (as in ERM analysis [18]), no assumption that our learning problem is described perfectly by a decision list (or some other functional form as in PAC analysis [17]), no assumption that we have a correct prior over the process generating learning problems (as in Bayesian analysis), and no assumption that training and test examples came from the same distribution. Our model is comparable to the (competition-based) online learning model with respect to its universality.
2. We show how to analyze reductions in this model. Intuitively, this analysis has to answer the question: "If good performance is observed on all subproblems created by the reduction, what is the worst performance that could be observed on the original problem?" We show that several existing algorithms in the machine learning literature can be analyzed in this model.
3. We present a new reduction from cost-sensitive to binary classification. Since every supervised learning task can be embedded in cost sensitive classification, this is a reduction from any supervised learning problem to the simplest (and most studied) learning problem. Thus binary classification is complete for the whole class of supervised learning problems. We empirically test this new reduction showing that the method is sound for real-world problems.
4. We close with discussion of what it means to be complete for the set of tasks and prove several simple properties common to all reductions.

2 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.

[4] Attempts to understand this model in terms of sample complexity based models of learning (see [18, 17]) will likely fail – this is simply a different model.

Boosting algorithms [6, 11] 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 (i.e. probabilistic). What this implies for a learning machine is that the loss rate becomes unbounded from $\frac{1}{2}$ after some number of rounds of boosting.

Bagging [3] 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.

Error Correcting Output Codes (ECOC) [4] generalize the tree reduction [7] and the one-against-all reduction [14] used for solving multiclass classification given a solver for binary classification. The "Costing" algorithm [19] is a reduction of importance weighted classification to classification. The "Probing" algorithm [12] is a reduction of class probability estimation to binary classification.

Pitt and Warmuth [13] also introduced a form of a reduction between classification problems called *prediction-preserving reducibility.* The reductions presented here differ in several respects: (1) they are between arbitrary learning *tasks* rather than between restricted sets of classification problems; (2) they are representation independent; (3) prediction-preserving reducibility is a *computational* notion, while the reductions here are *informational*; (4) prediction-preserving reductions make only one call to the oracle (i.e., problem they reduce to) and output the oracle's answer as its own answer. We enforce no restriction on how the oracle is used, making the setting directly applicable to ECOC, Boosting, and other common transformations[5]; (5) prediction-preserving reductions are typically used to show negative, *non*-predictability results, while we show positive results transferring learning algorithms between tasks.

3 The Error-Limiting Reduction Model

This section presents the basic notions and definitions of the model.

3.1 Basic Definitions

Definition 1. *A* supervised learning task *is a tuple (K, Y, ℓ), where K is the information given at training time, Y is the space of predictions, and $\ell : K \times Y \to [0, \infty)$ is a (nonconstant in both arguments) loss function.*

As illustrative examples, we now specify (K, Y, ℓ) for several classification tasks; lower-case symbols k and y denote the actual arguments of function ℓ; if the elements of K are tuples, subscripts index into the corresponding tuple.

[5] Abe and Watanabe [1] defined a Turing analogue of the prediction preserving reducibility, and showed that new reductions are feasible in this setting.

Classification Task	K	Y	ℓ
Binary	$\{0,1\}$	$\{0,1\}$	$I(k \neq y)$
Multiclass	$\{1,\ldots,r\}$	$\{1,\ldots,r\}$	$I(k \neq y)$
Importance Weighted	$\{1,\ldots,r\} \times [0,\infty)$	$\{1,\ldots,r\}$	$k_2 I(k_1 \neq y)$
Subset	$2^{\{1,\ldots,r\}}$	$\{1,\ldots,r\}$	$I(y \notin k)$
Cost-Sensitive	$[0,\infty)^r$	$\{1,\ldots,r\}$	k_y

Note that the space K of information provided at training time need not equal the prediction space Y. We sometimes need this additional flexibility (also used in [10]) to provide some information to the learning algorithm about the loss of different choices. In importance weighted classification, for example, we want to specify that predicting some examples correctly is more important than predicting others, while in cost-sensitive classification, K is used to associate a cost to each prediction $y \in Y$. In subset classification, K is the subset of correct predictions.

Normalization Issues. Some of these learning tasks are unnormalized—they can express arbitrarily large losses. This is realistic, but inconvenient for analysis. Consequently, we typically make two assumptions about the form of ℓ.

1. For all k, there exists y such that $\ell(k,y) = 0$. This is a convention to ease analysis, and not a restriction because we could always redefine $\ell'(k,y) = \ell(k,y) - \min_y \ell(k,y)$. Since any (approximate) minimizer of ℓ' is an (approximate) minimizer of ℓ, the prediction problem is unchanged.
2. We normalize the expected value of $\ell(k,y)$. Again, this is a convention to ease analysis since we can always redefine $\ell'(k,y) = a\ell(k,y)$ for any constant a without changing the prediction problem. The exact choice of normalization is discussed in each analysis.

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 $h : X \to Y$ 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 $h : X \to Y$.*

Standard supervised learning algorithms such as support vector machines, neural networks, decision trees, and logistic regression fit this definition[6]. We emphasize that we do

[6] Note, however, that these algorithms do not actually produce a classifier for *all* learning problems in their task because they do not work for all input spaces. We assume that our idealized supervised learning algorithm does. The same abstraction is also made in practice by learning practitioners.

not require that training and test examples are drawn from the same distribution. Our definitions capture natural learning tasks without depending on any assumptions that cannot be verified, or do not generally hold in practice. Note that we also do not require the learned hypothesis to belong to some predetermined class of hypotheses (see footnote 4).

Ultimately, the real test of a learning model is whether it allows for tractable analysis leading to useful learning algorithms. To motivate the definitions that follow, we first state the core theorem we prove on error-limitation bounds for a reduction from cost-sensitive classification to binary classification (presented in Section 5). We first analyze several standard reductions in this model in Section 4.

Theorem 1. *(Weighted All-pairs Reduction) For all cost sensitive problems $(D, X, \mathcal{T})$, if the learned binary classifiers have error rate at most ϵ, then the weighted all-pairs reduction has cost at most $2\epsilon E_{(x,k)\sim D} \sum_r k_r$.*

3.2 Reductions

Roughly speaking, 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 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 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 random bits 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 above argument.

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 normalized 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*

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

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, however. First, learning tasks are information theoretically hard. In particular, because D is typically not known in practice, the best classifier simply can not be determined, even with arbitrary amounts of computational effort. In complexity theory, computational constraints on reductions are necessary for structure to exist. We are, of course, interested in (and analyze) the efficiency of reductions, and it may be desirable in the future to consider the finer structure imposed by computational constraints. In complexity theory, 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.

It should be noted that not all reductions in the learning literature are black box. For example, Allwein et al. [2] give a reduction from multi-class 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. It is natural to wonder if there are inherent limitations in using learning algorithms as black boxes. Regardless of the answer, however, a black-box type of analysis is interesting for reasons listed in the introduction.

3.3 Metrics of success

We consider several ways to measure reductions.

Error Efficiency of a reduction *for a given example* is 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. This is the metric essential to the definition of an error limiting reduction.

Time Efficiency We want the transformation to be computationally efficient assuming that all oracle calls take unit time.

Call Efficiency We quantify how extensively a reduction uses its oracle based on (1) the number of oracle calls it makes; (2) whether the calls are 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).

Note that parallel calls can often be turned into a single call. Suppose that we make several parallel queries to the oracle. Let $\{S_i\}$ be the corresponding inputs (i.e., representing sample sets over the same feature space X), and let the corresponding hypotheses be

$\{h_i\}$. We could instead create a new sample set $S = \{(\langle x, i\rangle, k) : \langle x, k\rangle \in S_i\}$, then call the oracle on S to get a hypothesis h, and define $h_i(x) = h(\langle x, i\rangle)$.

Independence Preservation Many common learning algorithms are built under the assumption that examples are drawn independently from some distribution D'. Reductions which take an independently drawn dataset, and construct dependent datasets for the oracle learning algorithm are often undesirable for this reason.

4 Examples of Error-Limiting Reductions

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

4.1 Reduction from Importance Weighted Classification to Classification

Name	$g(\epsilon)$	Time	#Calls	Indep. Preserve
Costing	$\epsilon E_{(x,k_1,k_2)\sim D} k_2$	$O(m)$	O(1) parallel	Yes

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" [19] 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[7].

Although this algorithm is remarkably simple, composing costing with a decision tree has been shown to yield superior performance (again, see [19] for details) on championship prediction problems.

Theorem 2. *(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 [19] for details.

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

4.2 Reduction 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 (see [16],[15]) boosting algorithms for solving multiclass classification given weak binary importance weighted classification. 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.

Name	$g(\epsilon)$	Time	#Calls	Indep. Preserve
Tree	$\epsilon \log_2 r$	$O(\|S\|r)$	$r-1$ parallel	Yes
ECOC	4ϵ	$O(\|S\|r)$	$2r$ parallel	Yes

The Tree Reduction In the tree reduction (which is well known, see chapter 15 of [7]), we construct $r-1$ binary classifiers, which distinguish 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. In particular, we start with the set of all labels $\{1, \ldots, r\}$ at the top most node, and split the labels into ordered intervals $\{1, \ldots, r/2-1\}, \{r/2, r\}$ at each node. Predictions are made by following a chain of classifications down to the leaves, each of which is associated with a unique label.

Note that the trick of combining many calls into one described in Section 3.3 applies here as well. Also, instead of $r-1$ parallel, one could use $\log_2 r$ queries.

Theorem 3. *(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 code (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 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 4 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.

Also note that the "many calls to one call" trick in section 3.3 (call efficiency) can be applied here.

Theorem 4. *(ECOC loss efficiency) 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 [8] (which was generalized in [2]), 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.*

5 Reduction from Cost Sensitive Classification to Binary Classification

In this section we define a new reduction from cost sensitive classification to importance weighted binary classification. It can be composed with the "Costing" reduction from importance weighted classification to classification to yield a reduction to binary classification. Since cost sensitive classification can express any supervised learning task we consider, this implies that *any* such task can be solved using binary classification. We present experimental results for this reduction and prove the following efficiency guarantees:

Name	$g(\epsilon)$	Time	#Calls	Indep. Preserve
weighted all-pairs	$2\epsilon E_{(x,k)\sim D} \sum_r k_r$	$O(r^2)$	1	No

5.1 The Algorithm

We reduce (multiclass) cost-sensitive classification to importance weighted (binary) classification using maps of the form:

$$(x, k_1, \ldots, k_r) \mapsto ((x, i, j), I(k_i < k_j), |v_j - v_i|),$$

for all pairs (i, j) with $i < j$. The values $v_1, \ldots, v_r$ are functions of $k_1, \ldots, k_r$ defined below. The values are order-preserving (for all i, j, $k_i < k_j$ iff $v_i < v_j$).

Given m samples of training data for a cost-sensitive problem, we generate $m\binom{r}{2}$ training inputs for an importance weighted problem, by iterating through all $\binom{r}{2}$ possibilities for i and j. Now we use our oracle and this training data to generate a classifier, h. Given an input x for the cost-sensitive problem, we evaluate $h(x, i, j)$, for all the $\binom{r}{2}$ pairs $i, j \in [r]$, with $i < j$. Say label i "beats" label j for input x if either $i < j$ and $h(x, i, j) = 1$, or $i > j$ and $h(x, j, i) = 0$. Note that, if our classifier makes no errors and $k_i \neq k_j$, then label i beats label j exactly when $k_i < k_j$ (we are not concerned with the case $k_i = k_j$). Our algorithm outputs a label i which beats the maximum number of labels $j \neq i$. The mechanism for breaking ties does not affect our main result.

For $t \in [0, \infty)$, let $L(t) = \#\{j \mid k_j \leq t\}$. By shifting, we may assume that the minimum cost $k_{\min} = 0$, so that $t \geq 0$ implies $L(t) \geq 1$. Values v_i are defined by:

$$v_i = \int_0^{k_i} 1/L(t)\, dt.$$

This method of assigning importances is provably near-optimal for our algorithm.

Theorem 5. *(Error Limitation of Weighted All-pairs)*
Upper bound: *If the importance weighted binary classifier has loss rate at most ϵ, then the weighted all-pairs reduction has expected cost at most 2ϵ.*
Lower bound: *For any other assignments of importances $w_{i,j}$ to the sample points (x, i, j) in the above algorithm, there exists a distribution with expected cost $\epsilon/2$.*

The upper bound of theorem 5 and the costing reduction 2 together prove theorem 1.

For notational simplicity in the analysis, assume that $k_1 \leq \cdots \leq k_r$. Note that no generality is lost in this assumption 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 (of Lemma 1). 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 (of Theorem 5). For a particular input $(x, k_1, \ldots, k_r)$, suppose that our algorithm chooses the wrong label i. 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 time 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 of the reduction is at most 2.

For the lower bound, consider examples $\left(x, 0, \frac{1}{r-1}, \ldots, \frac{1}{r-1}\right)$. Suppose 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 a total cost of 1. The importance of these two errors is $w_{1,i} + w_{1,j}$, out of a 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. ∎

5.2 Experiments

We conducted experiments comparing the weighted all-pairs reduction to the original all-pairs reduction [9] (which ignores the costs and attempts to predict the label with minimal cost) and to a state-of-the-art multi-class cost-sensitive learning algorithm called MetaCost [5]. We used boosted Naive Bayes as the oracle classifier learner and applied the methods to five 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 the table below. Note that we often observe order-of-magnitude improvements in performance.

Dataset	All-Pairs	MetaCost	Weighted All-Pairs v.1	Weighted All-Pairs v.2
splice	59.8 ± 24	49.8 ± 3.05	197 ± 23	46.5 ± 3.5
solar	3989 ± 1415	5317 ± 390	24.7 ± 2.0	36.8 ± 11
anneal	310 ± 205	207 ± 43	29.9 ± 3.0	164 ± 43
kdd-99	234 ± 62	49.4 ± 9.3	0.956 ± 0.12	1.76 ± 0.66
satellite	137 ± 33	104 ± 6.4	428 ± 21	148 ± 8.5

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 by design.

6 The Structure of Error Limiting Reductions

In this section we discuss discuss various what complete means, and common properties of all error limiting reductions.

6.1 Easy Directions and Completeness

The learning tasks we consider have an easy direction for reduction related to whether a particular task is expressible as another task. For example, solving binary classification with importance weighted binary classification is easy, because importance weighted binary classification with $k_2 = 1$ is identical to binary classification. Figure 6.1 shows the easy direction for these reductions.

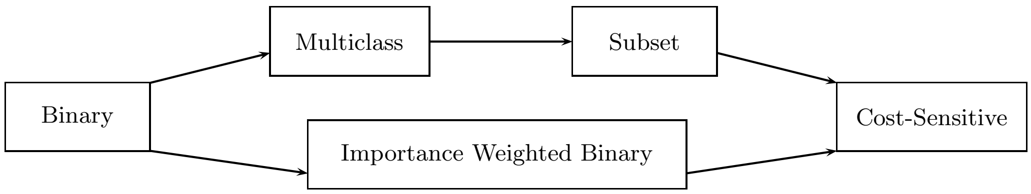

Fig. 6.1. The "easy" direction for reductions between the learning problems discussed here.

Note that all supervised learning tasks we consider can be embedded into cost-sensitive classification using the mapping $k'_y = \ell(k, y)$ for the task (k, y, ℓ). Consequently, cost-sensitive classification is (vacuously) complete for the set of supervised learning tasks. Section 5 shows that cost-sensitive classification is reducible to binary classification. Thus we can consider binary classification as complete for the set of supervised learning tasks considered. Stated another way, a learning machine consisting of a turing machine with a generalizing classification oracle can solve any supervised learning task.

6.2 Properties of Reductions

Error limiting reductions satisfy a nontriviality guarantee and can be shown to compose in the straight-forward manner.

Non-triviality of the definition Definition 4 is only significant when the oracle learner's loss rate is small. We can show that for every reduction this is sometimes the case. Consequently, trivial reductions that either never call A or create "hard" subproblems with arbitrarily large informational lower bounds on errors are not allowed.

Proposition 1. *For every error-limiting reduction R, every training set S, and test example (x, k), there exists an oracle A such that*

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

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

This proposition states that the reduction must make a non-trivial use of its oracle A (although it does not forbid some almost always suboptimal behaviors like encrypting x).

Proof. We first note that there must exist a sub-hypothesis h' used by the hypothesis h which the reduction creates. This is so because the convention $\max_\emptyset = 0$ in the error limiting reduction implies $\ell(k, h(x)) = g(\max_\emptyset) = g(0) = 0$. (Recall that the error limitation property holds for all D, in particular for the deterministic D with all probability mass on (x, k).) The fact that $\ell(k, y)$ is nonconstant (in both arguments) implies that there exists (x, k) such that any prediction machine[8] h satisfies $\ell(k, h(x)) > 0$, a contradiction. Consequently, at least one call to A must be made.

Next note that for all error limiting reductions R, samples S, and test samples (x, k), there exists an h' (and thus an A outputting h') such that $h'(x') = \arg\min_{y' \in Y'} \ell(k', y') = 0$. This is obvious if the reduction is deterministic. Since h' is a prediction machine, this is sufficient to predict perfectly for every element $(x', k') \in S'$.

In the case of a randomized reduction, note that for every randomized reduction R 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 $h(x) = \arg\min_{y \in Y} \ell(k, y) = 0$. ∎

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 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 $h'_{D_2} \le g_{23}(h''_{D_3})$, where h''_{D_3} is the maximum loss rate of a hypothesis h'' output by A

[8] Note that a machine with state is required rather than a (stateless) function to cope with the situation where the reduction uses the same x' twice.

(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 $g_{12}(g_{23}(0)) = 0$. ∎

7 Concluding Remarks

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).

1. Error-limiting reductions are very basic. Their analysis does not depend upon *any* unprovable (or unobservable) assumptions such as independence.
2. Error-limiting reductions capture several common machine learning algorithms.
3. Error-limiting reductions often have tractable analysis.
4. Error-limiting reductions often work well in practice.

We constructed, proved, and empirically tested an error-limiting reduction from cost sensitive classification to binary classification. Since cost sensitive classification can express any supervised learning task considered here, this means that *any* such task can be reduced to binary classification.

Several important problems remain to be addressed.

1. (scope) There are several learning problems (for example, outlier detection and clustering) that are not captured by the notion of supervised learning defined here and are not known to be reducible to binary classification. A model capable of capturing all of these would be of significant interest. An obvious question is whether there is a complete task for these problems and whether this primitive is fundamentally different from binary classification.
2. (refinement) While the notion of error limiting reductions is non-trivial, it does not (intuitively) prevent the reduction of an "easy" problem to a "hard" problem. Alternative definitions enforcing this do exist, but it is unclear what can be proved under these much stronger definitions.

References

1. Naoki Abe and Osamu Watanabe. Polynomially Sparse Variations and Reducibility among Prediction Problems, *IEICE Transactions on Communications, Electronics, Information, and Systems*, E75-D(4):449–458, 1992.
2. Erin Allwein, Robert Schapire, Yoram Singer. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, *Journal of Machine Learning Research*, 1:113–141, September 2001.
3. Leo Breiman. Bagging predictors, *Machine Learning*, 26(2):123–140, 1996.
4. Thomas G. Dietterich and Ghulum Bakiri. Solving Multiclass Learning Problems via Error-correcting Output Codes, *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.
5. Pedro Domingos. Metacost: A general method for make classifiers cost sensitive. *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155-164. ACM Press, 1999.

6. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
7. John Fox. *Applied Regression Analysis, Linear Models, and Related Methods*, Sage Publications, 1997.
8. Venkatesan Guruswami and Amit Sahai. Multiclass Learning, Boosting, and Error-correcting codes, *Proceedings of the 12th Annual Conference on Computational Learning Theory (COLT)*, 145–155, 1999.
9. Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*, 507–513, MIT Press, 1998.
10. David Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications, *Information and Computation*, 100:78–150, September 1992.
11. Adam Kalai and Rocco Servedio. Boosting in the Presence of Noise, *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing (STOC)*, 195–205, 2003.
12. John Langford and Bianca Zadrozny, Estimating class membership probabilities using classifier learners, AISTAT 2005.
13. Lenny Pitt and Manfred Warmuth. Prediction-Preserving Reducibility, *Journal of Computer and System Sciences*, 41:430–467, 1990.
14. Ryan Rifkin and Aldebaro Klautau. In Defense of One-Vs-All Classification, *Journal of Machine Learning Research*, 5:101–141, 2004.
15. Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
16. Robert E. Schapire. Using output codes to boost multiclass learning problems, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, 313–321, 1997.
17. L.G. Valiant, A Theory of the Learnable. *Communications of the ACM*, 27(11):1134-1142, November 1984
18. V. N. Vapnik and A. Y. Chevonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of probability and its applications*, 16(2):264-280, 1971.
19. Bianca Zadrozny, John Langford, and Naoki Abe. Cost Sensitive Learning by Cost-Proportionate Example Weighting, *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, 2003.