# REDUCTIONS FROM SUPERVISED LEARNING TO CLASSIFICATION

Abstract. A reduction converts a solver for one problem into a solver for another problem. We discuss and analyze reductions which automatically convert classification solvers into solvers for other supervised learning problems.

## 1. Introduction

### 1.1. Motivation.

There are many natural learning problems such as "classification", "cost sensitive classification", "ranking", "reinforcement learning", "outlier detection", and others. There are two extremes we use to cope with the many learning problems:

(1) Attempt to analyze every problem in great detail, coming up with unique analysis and algorithms for each problem.
(2) Construct reductions from unanalyzed problems to a thoroughly analyzed problem, attempting to transfer this thorough analysis into new domains.

We investigate approach (2). The reductions we analyze simultaneously:

(1) Show how related these natural problems are. Two learning problems are related if we have a short efficient reduction from one to the other and vice versa.
(2) Extend much of the hard work of older theories into new domains.
(3) Provide algorithms for solving these natural learning problems, when combined with a basic learning algorithm.

In addition to the above theoretical motivations, there appears to be some significant evidence (see [7], [3], [2]) that this style of analysis often accurately predicts good learning algorithm constructions.

### 1.2. Method.

A natural question when working with reductions is: "What should we reduce to and from?" Our answers are "to classification" and "from any formalized prediction problem". The choice of classification is motivated on three counts:

(1) Classification has been previously studied in significant detail.
(2) Classification seems to be the simplest possible prediction problem.
(3) Classification performance is upper and lower boundable under the (occasionally met) assumption that data is drawn i.i.d. from some (unknown but fixed) distribution $D$.

Two reasonable alternatives are density estimation (which satisfies (1) but not (2) and (3)) and regression (which satisfies (1) and (3), but not (2)). We use classification because the strongest case can be made for it.

Another natural question is "What is a good reduction?" It is easy to agree that a good reduction has:

(1) Good performance in the formalized problem's metric of success.
(2) Low computational overhead.
(3) Few invocations of the classifier.

(4) Parallelizes invocations of the classifier.
(5) Obeys the invariants the classification algorithm needs to succeed.

Of the above, (1) is tricky. We *know* that classification is hard in general so it's difficult to say that any algorithm using a classifier as a subroutine can perform well. To address this we make an assumption:

For the classification problems created by our reduction, our classifier learning algorithm succeeds with error rate $\epsilon$ or less.

This assumption is powerful because it allows us to abstract away the difficulty of classification (impossible in the worst case, but often feasible in practice) and focus on the efficiency and error transform characteristics of the reduction itself.

This assumption also invites abuse. Some very difficult problems (such as predicting plain text given examples of cipher text and plain text) can be phrased as learning problems. An analysis given this assumption may suggest that a reduction will perform well on this problem. This is not so, simply because the assumption is not met in practice.

A more subtle form of assumption misuse occurs when an otherwise-reasonable reduction fails to obey (5). One standard way this occurs is when the classification problem created is too noisy. If we considered "Adaboost" in this framework, we might say that it fails to obey (5) because it tends to concentrate learning on the subset of examples which are noisy. Another common misuse, which occurs in "Bagging", is failing to produce a dataset which is drawn i.i.d. from the appropriate distribution.

## 1.3. **Prior Work.**

There is actually quite a bit of significant prior work which is deeply related to the reductions analyzed here, simply due to the natural character of this construction. One of the goals of this paper is unifying the analysis of these algorithms in a common framework.

Boosting algorithms [3] are about reducing (typically binary but sometimes multiclass) classification with a small error rate to solving importance weighted classification problems with a loss rate of nearly $\frac{1}{2}$, a rather surprising statement. Some boosting variants have suffered with respect to criterion (5) above: they end up placing too much emphasis on noisy datapoints (see [4] for an analysis).

Bagging [1] can be thought of as a self-reduction of classification to classification. Bagging turns out to be useful because it turns learning algorithms with a high "variance" (dependence on the exact feature values) into a voting classifier with a lower "variance". Bagging also suffers with respect to criterion (5) since the subproblems it creates are not independently distributed according to the original distribution.

Error Correcting Output Codes (ECOC) [2] is a technique for solving multiclass classification given a solver for binary classification.

## 1.4. **Layout.**

(1) We first formally define several natural problems. These problems are the ones that we reduce from and to.
(2) We state reductions to binary classification.
(3) We state reductions to importance weighted classification.
(4) We discuss the meaning of these results.

## 2. Basic Definitions

### 2.1. **Supervised Learning.**

We first formalize some basic definitions which we rely upon throughout the paper.

**Definition 2.1.** (Supervised Learning Domain) A supervised learning domain is specified by a set $(K, Y, l)$ where $K$ is extra information available at training time and to the loss function, $Y$ is the space of predictions and $l : K \times Y \rightarrow [0, \infty)$ is a loss function.

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

**Definition 2.2.** (Supervised Learning Problem) A supervised learning problem $(D, X, K, Y, l)$ where $D$ is a distrbution over $X \times K$, $Y$ is a prediction space, and $l$ is a loss function $l : K \times Y \rightarrow [0, \infty)$.

A supervised learning problem is essentially a domain plus the extra information necessary to make the problem well specified. The goal in solving the supervised learning problem is to find a function $f : X \rightarrow Y$ which minimizes the loss with respect to $D$.

$$\arg\min_f E_{(x,k)\sim D} l(k, f(x))$$

**Definition 2.3.** (Supervised Learning Algorithm) A supervised learning algorithm $A$ takes as input a set of examples $(X \times K)^*$ and outputs a prediction function $f : X \rightarrow Y$.

Standard supervised learning algorithms such as "support vector machines" and "logistic regression" fit this definition. A "good" supervised learning algorithm approximately achieves the learning problem goal for natural learning problems in some domain. Notice that these algorithms don't actually work for arbitrary $X$, but our idealized supervised learning algorithm does.

We instantiate several domains next to give some intuition (and for future use).

**Definition 2.4.** (Binary Classification) Binary classification is a supervised learning domain where $K = Y = \{0, 1\}$ and $l(k, y) = I(k \neq y)$.

The binary classification domain is appropriate when we want a predictor for "right" or "wrong" choice.

**Definition 2.5.** (Importance Weighted Classification) Importance weighted classification is a supervised learning domain where $Y = \{1, ..., \ell\}$, $K = \{1, ..., \ell\} \times [0, \infty)$ and $l((k, i), y) = iI(k \neq y)$.

The importance weighted classification problem is appropriate when we want a predictor, but some of these predictions are more important than others.

**Definition 2.6.** (Multiclass Classification) Multiclass classification is a supervised learning domain where $Y = K = \{1, ..., \ell\}$ and $l(k, y) = I(k \neq y)$.

Multiclass classification is the obvious generalization of classification to situations with more than two choices.

**Definition 2.7.** (Multilabel Classification) Multilabel classification is a supervised learning domain where $Y = \{1, ..., \ell\}$, $K = 2^{\{1,...,\ell\}}$ and $l(k, y) = I(y \notin k)$.

Multilabel classification is a generalization of multiclass classification to situations with multiple "right" choices.

**Definition 2.8.** (Cost Sensitive Classification) Cost sensitive classification is a supervised learning domain where $Y = \{1, ..., \ell\}$, $K = [0, \infty)^\ell$ and $l(k, y) = k_y$.

Cost sensitive classification is appropriate when each choice has a varying value. The binary version of this problem is essentially isometric to importance weighted binary classification.

2.2. **Reductions.** A reduction is a meta-algorithm which uses another learning algorithm as an oracle (or black box) to handle some critical subproblem of another learning problem. We are fundamentally interested in meta-algorithms which perform well whenever the oracle learning algorithm performs well. A learning algorithm is said to perform $\epsilon$ well with respect to a distribution $D'$ if:

$$f^*_{D'} < \max_D \min_f E_{x,k\sim D} l(k, f(x))$$

A performance metric implies that we must specify a distribution $D'$ for each invocation of the black box. In general, a set of data $(x, k)^m$input to the meta-algorithm induces a a measure $D^{'}|(x, k)^m((x', k')^n)$ over $(x', k')^n$ given $(x, k)^m$. This distribution can be made into a distribution $D'|(x, k)^m$ over $X' \times K'$ by averaging over the index,

$$D^{''}|(x, k)^m(x', k') = E_{i\sim\{1,...,n\}} D'_i|(x, k)^m$$

where $D'_i$ is the measure on the $i$th component of $(x', k')^m$. The measure $D''|(x, k)^m$ is often a measure on a discrete set of points. We extend this to a more general measure by taking the expectation over $(x, k)^m$. In particular, $D' = E_{(x,k)^m\sim D} D''|(x, k)^m$. This gives us a well defined distribution to measure performance with respect to, given a distribution on the inputs of the meta algorithm.

One subtle issue arises: what is $D'$ for a second sequential invocation of the oracle learning algorithm? Suppose the first invocation produces a function $f$. We replace the first invocation with the oracle which always returns $f$, and use the above definition to find $D'$ for the second oracle.

**Definition 2.9.** (Supervised Learning Reduction) A supervised learning reduction $R_{(K,Y,I)\to(K',Y',l')}(A')$ given a supervised learning algorithm $A'$ for domain $(K', Y', l')$ is a supervised learning algorithm for domain $(K, Y, l)$ satisfying the following properties:

(1) (Error Transformation) $\forall A'$, $\forall X$, $\forall D \sim X \times K$, if $A'$ performs well on all invoked problems $(D'(D), X', K', Y', l')$ then $R(A')$ performs well on $(D, X, K, Y, l)$.
(2) (Error Transformation II) $\forall A'$, $\forall X$, $\forall D \sim X \times K$, if $A'$ performs badly on all invoked problems $(D'(D), X', K', Y', l')$ then $R(A')$ performs badly on $(D, X, K, Y, l)$.
(3) (Nontriviality) $\exists A', X, D$ such that $A'$ performs well.

The error transformation property is the important element of the above definition. It states the the performance of the derived algorithm on $D$ is intrinsically related to the performance of $A'$ on the derived problems $D'$. This implies that $A'$ is used in a nontrivial way to solve the prediction problem.

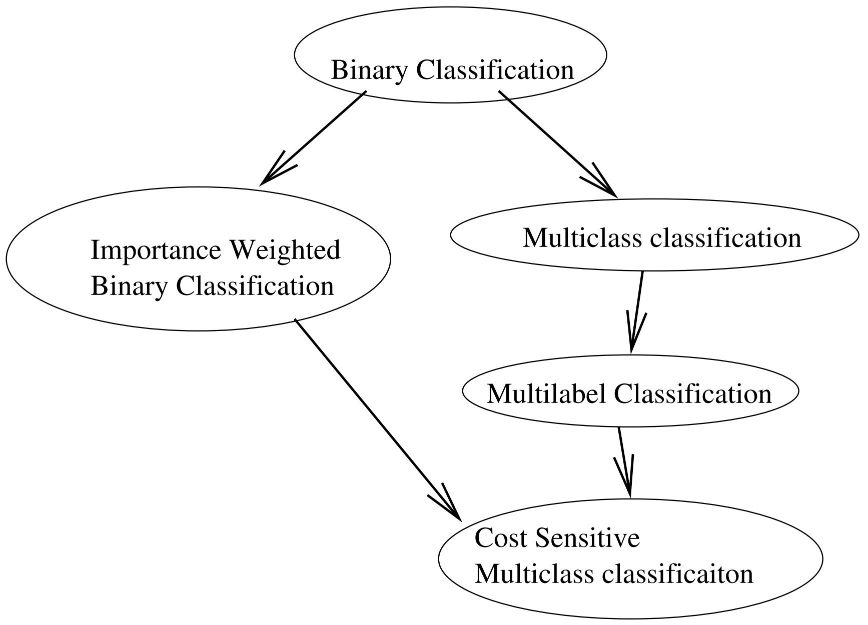
Binary Classification
Importance Weighted Binary Classification
Multiclass classification
Multilabel Classification
Cost Sensitive Multiclass classificaiton

FIGURE 3.1. A figure showing the "easy" direction for reduction amongst the learning problems discusses here.

2.3. **Reduction analysis method: Error transformation efficiency.** One of our primary methods for comparing reductions is according to their error transformation efficiency. A typical theorem statement has the form "If the oracle has error rate $\epsilon$, then the reduction has error rate $f(\epsilon)$" for some continuous $f(x)$ with $f(x) = 0$.

2.4. **Reduction Analysis method: Double reduction.** Suppose we have two reductions $R_{(X,Y,l)\to(X',Y',l')}$ and $R_{(X',Y',l')\to(X,Y,L)}$. We would like to observe that: $R_{(X',Y',l')\to(X,Y,L)}(R_{(X,Y,l)\to(X',Y',l')}(A')) = A'$. In particular, this implies that the set of problems solvable by $A'$ are the same as the set of problems solvable by the double reduction $R_{(X',Y',l')\to(X,Y,L)}(R_{(X,Y,l)\to(X',Y',l')}(A'))$.

This desire is too strong to hope for in general, but we can sometimes prove weakenings of this statement.

## 3. REDUCTIONS, THE EASY WAY

The learning problems above have an 'easy' direction for reduction related to whether a particular problem is definable as a subset of other problems.

For example, solving binary classification with importance weighted binary classification is easy because importance weighted binary classification for $i = 1$ is identical to binary classification. Figure 3.1 shows the easy direction for these reductions. The remainder of this paper is devoted to reductions in the other "hard" direction.

## 4. Reductions to Binary Classification

### 4.1. **Reduction from Importance Weighted Classification to Classification.**

### 4.2. **Reduction from Multiclass Classification to Binary Classificaton.**

There are several well known methods of reduction which we analyze for error efficiency next. Before we start, it's important to note that there are (see [6],[5]) 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.

4.2.1. *The one-against-all reduction.* In the one-against-all reduction, we learn $k$ classifiers $h_\ell$, using the mapping:

$$(x, y) \to (x, I(y = \ell))$$

In order to construct a multiclass classifier from the binary classifier we use the following procedure:

Predict$(h, x)$

(1) If there exists a label $\ell$ such that $h_\ell(x) = 1$ then predict $\ell$, breaking ties with randomization.
(2) Otherwise predict randomly.

**Theorem 4.1.** *(One-against-all error efficiency) If the binary classifiers have error rate $\epsilon$, the one-against-all reduction has error rate at most:*

$$(k-1)\epsilon$$

*Proof.* A "false negative" when the output should be 1 but is actually 0 produces an error a $\frac{k-1}{k}$ fraction of the time. The other error modes to consider involve (possibly multiple) false positives. If there are $n$ false positives, the error probability is either $\frac{n}{n+1}$ or 1 if there was also a false negative. The efficiency of creating errors is: $\frac{\frac{k-1}{k}}{1} = \frac{k-1}{k}$, $\frac{\frac{n}{n+1}}{n} = \frac{1}{n+1}$, and $\frac{1}{n+1}$. Taking the maximum, we get $\frac{k-1}{k}$. Multiplying by $k$ (because we have $k$ opportunities to err, one for each classifier) we get the bound. □

Note, it's actually possible to do the one-against-all reduction using $(x, y) \to \forall l \;\; ((x, l), I(y = l))$, and learning one classifier. This method has the same error transfer efficiency.

4.2.2. *The Tree Reduction.* In the tree reduction we construct $k-1$ binary classifiers which distinguish between the labels using a binary tree. The tree has depth $\log_2 k$, and each internal node is associated with a set of labels that can reach the node, assuming each classifier above it predicts correctly. The set of labels at that node is split into two sets of half the size, and a classifier is learned to distinguish between the two sets of labels.

Predictions are made by following the chain of classifications down to the leaves, which are associated with a unique label.

**Theorem 4.2.** *(Tree error efficiency) If the binary classifiers have error rate $\epsilon$, the tree reduction has error rate at most:*

$$\epsilon \log_2 k$$

*Proof.* An error occurs only if some classifier on the path from root to leaf errs. □

4.2.3. *The error correcting output code (ECOC) reduction.* In the (limiting) ECOC reduction [2], we learn a classifier for each of the $2^k$ subsets of labels. Any particular label will have a $2^k$ length binary vector associated with it corresponding to inclusion or not in the subset. Predictions are made by choosing the label smallest hamming distance to the classifications.

**Theorem 4.3.** *(ECOC error efficiency) If the binary classifiers have error rate $\epsilon$, the ECOC reduction has error rate less than:*

$$4\epsilon$$

The ECOC reduction is typically applied to a subset of the powerset rather than the full powerset. The same basic result (with a worsened constant) applies. 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 will be highly *dependent*.

*Proof.* When the error rate is zero, the correct label will be consistent with every classifier (out of$2^k$) and wrong labels will be consistent with only half of the classifiers ($2^{k-1}$). In order to misclassify the multiclass label, we must have at least $2^{k-2}$ classifiers err simultaneously.

Since every multiclass error implies at least $\frac{1}{4}$ of the binary classifiers erred, a binary error rate of $\epsilon$ can produce at most a multiclass error rate of $4\epsilon$ □

## 5. Reductions to Importance Weighted Binary Classification

### 5.1. **Classification to Weak Importance Weighted Binary Classification (aka boosting).**

### 5.2. **Multilabel Classification.**

5.2.1. *The One-against-all Reduction.* This reduction is an optimized one-against-all reduction for multilabel classification to importance weighted binary classification. It appears to be new.

In the one-against-all reduction, we learn $k$ classifiers, one for each possible output label using data from the mapping:

$$(x, y_1, y_2, ..., y_n) \rightarrow (x, I(\ell \in \{y_1, ..., y_n\}), i_{I(\ell \in \{y_1,...,y_n\})})$$

where $i_0 = \frac{k}{n+1}$ and $i_1 = \frac{k}{n+1}$ if $k \leq n^2 + n$ or $i_1 = \frac{k-n}{n}$ for $k \geq n^2 + n$

Suppose we use the same prediction function as in section 4.2.1. Then, we can prove the following theorem:

**Theorem 5.1.** *(Multilabel classification error efficiency) Let $c_{kn} = \frac{k}{n+1}$ for $k \leq n^2 + n$ or $c_{kn} = \frac{k+1}{n+1} - \frac{n}{k}$ otherwise. If the binary classifiers have (normalized) loss rate $\epsilon$, the one-against-all reduction has error rate at most:*

$$\epsilon E_{x,y_1,...,y_n \sim D} c_{kn} \leq \epsilon E_{x,y_1,...,y_n \sim D} \frac{k}{n}$$

This theorem should be compared directly with the one-against-all reduction to binary classication which had an error efficiency of $k-1$. Here, with $n=1$, we get $c_{kn} = \frac{k+1}{2} - \frac{1}{k} = \frac{(k+2)(k-1)}{2k}$ implying an error efficiency of about $\frac{k}{2}$, or about half the error rate of the earlier reduction. Reducing to importance weighted binary classification has an improved efficiency.

*Proof.* Outline: For a particular example with $n$ "right" labels out of $k$ possible labels. We first show that there are 3 possible "most efficient" methods for a binary classifier to induce a multiclass error. These possibilities are: 1) a single false positive error, 2) all possible false negative errors, or 3) (1) and (2). After showing these are the only possibilities, we analyze which is the worst, and then take an expectation over $n$ and $k$.

There are two forms of errors, false positive and false negative. It is never more efficient to have multiple false positive errors. This is because for any number $n \geq p > 0$ of true positives, $\frac{2}{p+2} < \frac{1}{p+1} + \frac{1}{p+1}$ and for $p = 0$, only one false negative is required to err with probability 1.

If the optimal error rate occurs for $f < n$ false negatives then there must be 1 false positive (since the error rate is 0 otherwise). For $f < n$ false negatives and 1 false positive, we have an error rate of $\frac{1}{n-f+1}$ using an importance $fi_1 + i_0$ giving us an efficiency of $\frac{\frac{1}{n-f+1}}{fi_1+i_0}$. Since

$$\frac{\frac{1}{n-f}}{(f+1)i_1 + i_0} > \frac{\frac{1}{n-f+1}}{fi_1 + i_0}$$
$$\Leftrightarrow \frac{fi_1 + i_0}{n-f} > \frac{(f+1)i_1 + i_0}{n-f+1}$$
$$\Leftrightarrow (fi_1 + i_0)(n-f+1) > ((f+1)i_1 + i_0)(n-f)$$
$$\Leftrightarrow fi_1 + i_0 > i_1(n-f)$$
$$\Leftrightarrow i_0 > i_1(n-2f)$$

We can always improve the error efficiency by increasing $f$ whenever $i_0 > i_1(n-2f)$ and by decreasing $f$ otherwise. Consequently, the number of false negative errors is either $f=0$ or $f=n$.

For the $f=0$ case we have an error efficiency of $\frac{\frac{1}{n+1}}{i_0}$.

For the $f=n$ case there are again two possibilities: one or zero false positives. These cases have an error rate of $\frac{k-n}{k}$ and 1 with an importance consumption of $ni_1$ or $ni_1 + i_0$ implying error efficiencies of $\frac{\frac{k-n}{k}}{ni_1}$ and $\frac{1}{ni_1+i_0}$.

The next step is determining which of these failure modes is "most efficient" in the sense that it induces the largest number of multiclass error for the least cost in "importance".

The maximal error efficiency is $\max\left\{\frac{k-n}{kni_1}, \frac{1}{ni_1+i_0}, \frac{1}{(n+1)i_0}\right\}$. There are two regimes for this maximum, $k \geq n^2+n$ and $k \leq n^2+n$.

For $k \leq n^2+n$, we get:$\max\left\{\frac{k-n}{\frac{k^2n}{n+1}}, \frac{1}{k}, \frac{1}{k}\right\} = \frac{1}{k}$.

For $k \geq n^2+n$, we get: $\max\left\{\frac{1}{k}, \frac{1}{k-n+\frac{k}{n+1}}, \frac{1}{k}\right\} = \frac{1}{k}$.

Since the maximal error efficiency is $\frac{1}{k}$ and there are $k$ classifications, a binary importance weighted loss of $\epsilon$ implies a multiclass error rate of $\epsilon$. However, the importance weighted loss is unnormalized since $\frac{n}{k}i_1 + \frac{k-n}{k}i_0 = \frac{k}{n+1}$ for $k \leq n^2 + n$ or $(\frac{k+1}{n+1} - \frac{n}{k})$ for $k \geq n^2 + n$. Taking the expectation over $n$ and $k$ drawn according to $D$, we get the result. □

5.3. **Multiclass cost sensitive classification.** Here we attempt to reduce multiclass cost sensitive classification to simpler problems.

5.3.1. *The XYZ reduction.* Consider the reduction which maps $(x, k_1, k_2, ..., k_l) \rightarrow (x, y, \max_i k_i - k_y)$ for a uniformly shosen $y \in \{1, ..., l\}$. This dataset is fed into the importance weighted classification solve resulting in a classifier $c : X \rightarrow \{1, ..., l\}$ which is used to solve the classification problem.

The difficulty with this reduction is that it can be simply impossible to produce a predictor with a small error rate. In particular, it's only for lucky distributions where just one of the costs is large that the problem we reduce to can have a small error rate. Nonetheless, in this lucky case, a very strong statement can be made.

**Theorem 5.2.** *(error transformation) If the importance weighted classifier has loss rate $\epsilon$, then the cost sensitive classifier has expected cost $\epsilon$.*

*Proof.* Assume that

$$\begin{aligned}
\epsilon &= E_{x,y,i\sim D^*} iI(c(x) \neq y) \\
&= E_{x,k_1,...,k_l\sim D, y\sim\{1,...,l\}} (\max_i k_i - k_y) I(c(x) \neq y) \\
&= E_{x,k_1,...,k_l\sim D, y\sim\{1,...,l\}} \max_i k_i I(c(x) \neq y) - E_{x,k_1,...,k_l\sim D, y\sim\{1,...,l\}} k_y I(c(x) \neq y) \\
&= \frac{l-1}{l} E_{x,k_1,...,k_l\sim D} \max_i k_i - E_{x,k_1,...,k_l\sim D} \frac{\sum_i k_i - k_{c(x)}}{l} \\
&= E_{x,k_1,...,k_l\sim D} \frac{(l-1)\max_i k_i - \sum_i k_i + k_{c(x)}}{l} \\
&= E_{x,k_1,...,k_l\sim D} \frac{k_{c(x)} - \max_i k_i}{l} + \max_i k_i - E_{i\sim\{1,...,l\}} k_i
\end{aligned}$$

Rewriting, we get:

$$\begin{aligned}
E_{x,k_1,...,k_l\sim D} k_{c(x)} &= \epsilon - l[E_{x,k_1,...,k_l\sim D} \max_i k_i - E_{i\sim\{1,...,l\}} k_i] + E_{x,k_1,...,k_l\sim D} \max_i k_i \\
&= \epsilon - E_{x,k_1,...,k_l\sim D} (l-1) \max_i k_i - \sum_i k_i \\
&\leq \epsilon
\end{aligned}$$

Where the last inequality follows because on of the $k_i$ is 0. □

5.3.2. *sdf.* Nothing here is coherent yet.

Suppose we want to reduce from multiclass2 cost sensitive classification to cost sensitive classification. Then what?

Suppose we pick some threshold $w$. For example below the threshold, we try to learn to classify "1", while for those above we learn to classify "0". What should the importance weights be?

Choose importance weights so that the most efficient error mechanism is completely false negative and 1 or 0 false positives.

No more than 1 false positive: Suppose we have 2 false positives with costs $c_1$ and $c_2$ and importances $i_1$ and $i_2$. Suppose we also have $p > 0$ true positives with an average cost of $c_p$. Then, the change in the expected cost is $\frac{c_1+c_2}{p+2}$ incurring a loss of $i_1 + i_2$. Suppose instead that we committed each individual false positive independently. Then, $\frac{c_1}{p+1} + \frac{c_2}{p+1} > \frac{c_1+c_2}{p+2}$ , implying a greater error efficiency. The same is true for $p = 0$.

Complete false negative: Must show that it is always desirable to decrease the number to 0 or increase the number to all.
$\frac{E_l c_l}{\sum_{l:c_l<w} i_l}, \frac{c_l}{i_l+\sum_{l:c_l<w} i_l}, \frac{1}{i_l}\frac{c_l+nc_w}{n+1}$

## 6. Conclusion

We should construct a table of results here.

## References

[1] L. Breiman. Bagging predictors. Machine Learning, 26(2):123-140, 1996.

[2] 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.

[3] 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.

[4] Adam Kalai and Rocco Servedio, "Boosting in the Presence of Noise" To appear in Proceedings of the Thirty-Fifth Annual ACM Symposium on the Theory of Computing (STOC 2003).

[5] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3):297-336, 1999.

[6] Robert E. Schapire. Using output codes to boost multiclass learning problems. In Machine Learning: Proceedings of the Fourteenth International Conference, 1997.

[7] Bianca Zadrozny, John Langford, and Naoki Abe Cost Sensitive Learning by Cost-Proportionate Example Weighting ICDM 2003