

---

# Multiclass Classification with Filter Trees

---

**Alina Beygelzimer**  
beygel@us.ibm.com  
IBM Research, Hawthorne, NY

**John Langford**  
jl@yahoo-inc.com  
Yahoo! Research, New York, NY

**Pradeep Ravikumar**  
pradeep@cs.cmu.edu  
School of Computer Science  
Carnegie Mellon University  
Pittsburgh, PA

## Abstract

We present a new method, filter tree, for reducing  $k$ -class classification to binary classification. The filter tree is provably *consistent* in the sense that given an optimal binary classifier, the reduction yields an optimal multiclass classifier. (The commonly used tree approach is provably inconsistent.) We show that the filter tree is *robust*: It suffers multiclass regret at most  $\log_2 k$  times the binary regret. The method can also be used for cost-sensitive multiclass classification, where each prediction may have a different associated loss. The resulting regret bound is superior to the guarantees provided by all previous methods.

## 1 Introduction

In  $k$ -class classification, the goal is to assign instances to one of  $k$  possible classes. Binary (two-class) classification is a well-studied special case.

Given that there are many good binary learning algorithms, a common approach has been to create meta-algorithms, or reductions, which solve multiclass learning problems by reducing them to learning binary classification problems.

We propose a new reduction method, Filter Tree, motivated by optimizing performance transformation properties of the reduction. The algorithm uses a carefully constructed tournament on the set of classes—a series of matches where the loser of each match is immediately eliminated from the tournament. Here we are interested in determining the “winning” class with the highest conditional probability given an observation. The tournament can be viewed as a binary tree where each node predicts which of its two inputs is more likely; for internal nodes, the inputs are the predictions of the left and the right subtrees entering the node. The key insight of this paper which makes a tournament reduction possible is defining the prediction problem at each interior node conditionally on the predictions of the parent nodes providing the inputs. Thus the learned classifiers from the first level of the tree are used to “filter” the distribution over examples reaching the second level of the tree. This process repeats until the root node is reached.

We review commonly used reductions to convey the motivation.

Probably the simplest reduction is One-Against-All [4], which creates one binary classifier for each of the  $k$  classes, distinguishing instances from this class from all other instances. Multiclass predictions are made by randomizing over all classes whose associated predictions are positive, or over all classes if all binary predictions are negative. The resulting multiclass error rate is at most  $(k - 1)$  times the average error rate of the learned binary classifiers. A simple modification, called Weighted-One-Against-All (WOA), yields an improvement by about a factor of two [2].

Another common reduction is Tree (see [5]), which is a divide-and-conquer technique that distinguishes between the labels using a binary tree. Starting from the root containing all labels, the set of labels at each internal node is recursively split in half, and a classifier is trained to distinguish between the two subsets. 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. If the average binary error rate is  $e$ , the tree reduction has multiclass error rate at most  $\lceil \log_2 k \rceil e$  (see, for example, [3]).

For such error transformation statements to be useful, the average binary error rate has to be small. If the problems are noisy and the binary Bayes error rate itself is large, then multiplying it by an error transform rate would give a trivial bound on the multiclass error rate. This motivates the analysis of how reductions transform regret instead of error. *Regret* of a classifier is the difference between its error rate and the lowest achievable error (or the Bayes error rate) on the same problem. In other words, regret subtracts off the unavoidable loss, making it possible to make nontrivial statements for noisy problems.

Unfortunately, many reductions provably do not have a finite regret transform: Given a Bayes-optimal binary classifier, these reductions do not yield an optimal multiclass classifier in the presence of noise. We call such reductions *inconsistent*.

The following table summarizes the characteristics of different reductions. Here  $e$  is the average binary error rate, and the Error column gives an upper bound on the error rate of the multiclass classifier as a function of  $e$ . The Regret column is defined similarly in terms of the average binary regret  $r$ . The entries with “none” indicate that the method is inconsistent. The Evaluations column gives the number of evaluations of the binary classifier which are necessary to make the multiclass prediction at test time.

	Error	Regret	Evaluations
WOA [2]	$\sim \frac{k}{2}e$	none	$k$
All Pairs [6]	$(k-1)e$	$(k-1)r$	$\frac{k(k-1)}{2}$
Tree	$\lceil \log_2 k \rceil e$	none	$\lceil \log_2 k \rceil$
ECOC [4]	$4e$	none	$O(\log k)$
PECOC [8]	$4\sqrt{e}$	$4\sqrt{r}$	large
Filter Tree	$\lceil \log_2 k \rceil e$	$\lceil \log_2 k \rceil r$	$\lceil \log_2 k \rceil$

All-Pairs [6] works by learning a classifier for each pair of classes, to predict which of the two classes is more probable than the other. Multiclass predictions are made by running all pairwise classifiers and outputting the label maximizing the number of pairwise “wins”, with ties broken arbitrarily.

The filter tree has the best performance in each category, except when compared to ECOC and PECOC, the Error Correcting Output Code reduction [4] and its probabilistic version from [8]. ECOC has a better error transform bound for  $k > 16$ , but it is inconsistent. PECOC has no dependence on  $k$ , but it has a significantly worse dependence on the binary regret since  $\sqrt{r} \geq r$  for  $0 \leq r \leq 1$ .

The filter tree also naturally supports cost-sensitive multiclass classification where each prediction has a different associated cost. The dependence on the costs is superior to the Weighted All-Pairs reduction [3] (a cost-sensitive variant of All-Pairs), and SECOC [8] (a cost-sensitive variant of PECOC); both have an upper bound dependent on the expected sum of costs. The dependence for a filter tree is on the expected sum of cost differences (see Theorem 4.1 for a precise definition), which is never larger than the sum of costs. Furthermore, Theorem 4.3 bounds the cost-sensitive multiclass regret by  $k/2$  times the binary regret, yielding a bound superior to the guarantees provided by all previous methods.

The filter tree algorithm is similar to Adaptive Directed Acyclic Graphs [7], which were introduced as a computational optimization on Decision Directed Acyclic Graphs [9], and analyzed as an error transform. The filter tree differs in the training mechanism, which is essential for the regret analysis, and in the generalization to cost-sensitive multiclass classification.

## 2 Definitions

A  $k$ -class classification problem is defined by a distribution  $P$  over  $X \times Y$ , where  $X$  is some observable feature space and  $Y = \{1, \dots, k\}$  is the label space. The goal is to find a classifier  $h : X \rightarrow Y$  minimizing the *classification loss* on  $P$  given by

$$e(h, P) = \Pr_{(x,y) \sim P}[h(x) \neq y].$$

Binary classification corresponds to the  $k = 2$  case. The *classification regret* of  $h$  on  $P$  is defined as

$$r(h, P) = e(h, P) - \min_{h^*} e(h^*, P).$$

The motivation for regret analysis is to separate avoidable loss from loss inherent to the problem, so that bounds apply nontrivially for problems with large noise rates.

A *cost-sensitive*  $k$ -class classification problem is defined by a distribution  $D$  over  $X \times [0, \infty)^k$ . The goal is to find a classifier  $h : X \rightarrow \{1, \dots, k\}$  minimizing the expected cost  $e(h, D) = E_{(x, \vec{c}) \sim D}[c_{h(x)}]$ . Here  $\vec{c} \in [0, \infty)^k$  gives the cost of each of the  $k$  choices for  $x$ . As in the multiclass case (which is just a special case), the regret of  $h$  on  $D$  is defined as  $r(h, D) = e(h, D) - \min_{h^*} e(h^*, D)$ .

*Importance-weighted* binary classification is binary classification where each example has an associated weight specifying how important it is to predict its label correctly. Formally, the problem is defined by a distribution  $D$  on  $X \times \{0, 1\} \times [0, \infty)$  and loss function  $E_{(x,y,w) \sim D}[wI(b(x) \neq y)]$ , where  $I(\cdot)$  is 1 when the argument is true and 0 otherwise.

## 3 The Filter Tree Algorithm

The filter tree algorithm is illustrated by Figure 1. In a nutshell, the algorithm is a “bottom-up” tree algorithm equivalent to a single-elimination tournament on the set of labels. The underlying graph structure is a binary tree, constructed recursively from the root as in the tree reduction described in the introduction. Prediction problems associated with the nodes are, however, different.

In the first round, the labels are paired according to the tree, and a classifier is trained for each pair to predict which of the two labels is more likely. (The labels that don’t have a pair in a given round, win that round for free.) The winning labels from the first round are in turn paired in the second round, and a classifier is trained to predict whether the winner of one pair is more likely than the winner of the other. The process of training classifiers to predict the best of a pair of winners from the previous round is repeated until the root classifier is trained.

The key task in the training stage (Algorithm 1) is to form the right training set to distinguish between the labels at interior nodes. Each training example for node  $n$  is formed *conditionally* on the predictions of its parent classifiers in the round before it. Thus the learned classifiers from the first level of the tree are used to “filter” the distribution over examples reaching the second level of the tree. This process repeats until the root node is reached.

To see why this is the right process to form induced distributions, consider the multiclass case. For any observation  $x$ , the optimal decision at any internal node is to choose the label proportionally to the true conditional probabilities of the two input labels given  $x$ . This can be done by using the outputs of the parent classifiers as a filter during the training process: For each observation, we set the label to 0 if the left parent’s output matches the multiclass label, 1 if the right parent’s output matches, and reject the example otherwise. Algorithm 1 extends this idea to the cost-sensitive multiclass case.

The testing algorithm 2 is very simple. We just predict which input to the root node has the best label, go to the node predicting that output, and repeat until a leaf is reached, determining the multiclass prediction.

As shown in Figure 1, the actual predictions form a filter tree. While the training algorithm is stated for cost-sensitive multiclass classification, the multiclass variant can be formed by simply projecting multiclass examples  $(x, y)$  into cost sensitive examples  $(x, \vec{c})$  where  $c_y = 0$ , and for all  $y' \neq y$ ,  $c_{y'} = 1$ .

---

**Algorithm 1** The filter tree training algorithm

---

Filter-Train (cost-sensitive training set  $S$ , importance-weighted binary learner Learn)

1. Fix a binary tree  $T$  over the labels.
  2. For each internal node  $n$  in the order from leaves to roots:
    - (a) For each example  $(x, c_1, \dots, c_k) \in S$ 
      - i.  $S_n = \emptyset$
      - ii. Let  $a$  and  $b$  be the two classes input to  $n$  (for internal nodes, these are the predictions of the left and the right subtrees on input  $x$ ).
      - iii.  $S_n \leftarrow S_n \cup \{(x, \arg \min\{c_a, c_b\}, |c_a - c_b|)\}$
    - (b) Let  $\text{predict}_n = \text{Learn}(S_n)$
  3. return  $\{\text{predict}_n\}$
- 

---

**Algorithm 2** The filter tree testing algorithm

---

Filter-Test (classifiers  $\{\text{predict}_n\}$ , test example  $x \in X$ )Output the label  $l$  such that every classifier on the path from leaf  $l$  to the root prefers  $l$ .

---

The training algorithm relies upon an importance weighted binary learning algorithm, which takes examples of the form  $(x, y, w)$ , where  $x$  is a feature vector used for prediction,  $y$  is a binary label, and  $w$  is a positive real-valued importance. (The examples are formed in line 2(a)(iii) of Algorithm 1.) Importance-weighted binary classification can be further reduced to binary classification using the Costing reduction [10] (which alters the underlying distribution using rejection sampling on the importance weights), or other methods.

## 4 Reduction Analysis

In this section, we analyze the regret transform of the filter tree algorithm. First, we define several concepts necessary to understand the theorem statement.

Filter-Train transforms cost-sensitive multiclass examples into importance-weighted binary examples. This process implicitly transforms a distribution  $D$  over cost-sensitive multiclass examples into a distribution over importance-weighted binary examples, denoted by  $\text{Filter-Train}(D)$ .

There are many induced problems, one for each call to the oracle Learn. To simplify the analysis, we use a simple trick which allows us to consider only a single induced problem. The trick is to add the node index  $n$  as an additional feature into each importance weighted binary example, and then train based upon the union of all the training sets. The learning algorithm produces a single binary classifier  $\text{predict}(x, n)$  for which we can redefine  $\text{predict}_n(x)$  as  $\text{predict}(x, n)$ . Given this, we can define the induced distribution  $\text{Filter-Train}(D)$  by the following process: (1) draw a cost-sensitive example  $(x, \vec{c})$  from  $D$ , (2) pick a uniform random  $n$ , (3) create an importance-weighted sample  $((x, n), y, w)$  according to line 2(a)(iii), except with  $n$  added into the features.

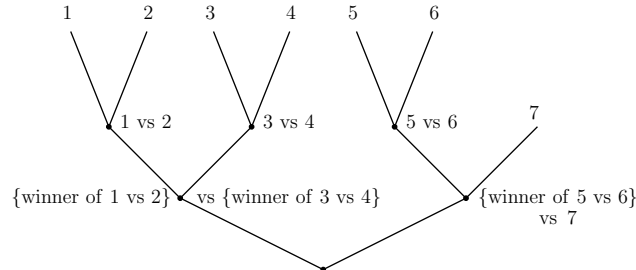


Figure 1: Filter Tree. Each node predicts whether the left or the right input label is more likely, conditioned on a given  $x \in X$ . The final output node predicts the best label for  $x$ .

The fact that classifiers are conditionally dependent on other classifiers closer to the leaves is not problematic: In practice, there are known effective iterative techniques for dealing with this cyclic dependence; alternatively, a separate classifier can be trained for each node. In theory, we quantify over *all* predictors, which means that we can regard the learned predictor as fixed, implying that the induced problem is well defined.

When reducing to importance-weighted classification, the theorem statement depends on importance weights. To remove the importances, we compose the reduction with the Costing reduction [10], which alters the underlying distribution using rejection sampling on the importance weights. This composition transforms  $\text{Filter-Train}(D)$  into a distribution  $D' = \text{Costing}(\text{Filter-Train}(D))$  over binary examples.

We use the folk theorem from [10] saying that for all binary classifiers  $b$  and all importance weighted binary distributions  $P$  the importance weighted binary regret of  $b$  on  $P$  is upper bounded by  $E_{(x,y,w)\sim P}[w]$  times the binary regret of  $b$  on the induced binary distribution  $\text{Costing}(P)$ .

The core theorem relates the regret of a binary classifier to the regret of a cost sensitive classifier.

**Theorem 4.1.** *For all binary classifiers  $b$  and all cost sensitive multiclass distributions  $D$ ,*

$$r(\text{Filter-Test}(b, \cdot), D) \leq r(b, D') E_{x, \vec{c} \sim D} \sum_{n \in T} w_{n,x,\vec{c}}$$

where  $D' = \text{Costing}(\text{Filter-Train}(D))$  and  $w_{n,x,\vec{c}}$  is the importance weight given by line 2(a)(iii) in Algorithm 1.

Before proving the main theorem, we state the corollary for multiclass classification.

**Corollary 4.2.** *For all binary classifiers  $b$  and all multiclass distributions  $D$  on  $k$  labels,*

$$r(\text{Filter-Test}(b, \cdot), D) \leq \lceil \log_2 k \rceil \cdot r(b, \text{Filter-Train}(D))$$

The proof of the corollary given the theorem is simple since for any  $(x, y)$ , the induced  $(x, \vec{c})$  has at most one node per level with induced importance weight 1; all other importance weights are 0. Therefore,  $\sum_n w_{n,x,\vec{c}} \leq \lceil \log_2 k \rceil$ .

Theorem 4.3 provides an alternative bound for cost-sensitive classification. It is the first known bound giving a worst-case dependence of less than  $k$ .

**Theorem 4.3.** *For all binary classifiers  $b$  and all cost-sensitive  $k$ -class distributions  $D$ ,*

$$r(\text{Filter-Test}(b, \cdot), D) \leq \frac{k}{2} r(b, D').$$

A simple example in the appendix shows that this bound is asymptotically tight.

The remainder of this section proves Theorems 4.1 and 4.3.

**Proof of Theorem 4.1** It is sufficient to prove the claim for any  $x \in X$  because that implies that the result holds for all expectations over  $x$ .

Conditioned on the value of  $x$ , each label  $y$  has a distribution over costs  $c_y$  with an expected value  $E_{\vec{c} \sim D|x}[c_y]$ . The zero regret cost sensitive classifier predicts  $\arg \min_y E_{\vec{c} \sim D|x}[c_y]$ . Suppose that  $\text{Filter-Test}(b, x) = y'$ , inducing cost sensitive regret

$$r(y', D|x) = E_{\vec{c} \sim D|x}[c_{y'}] - \min_y E_{\vec{c} \sim D|x}[c_y].$$

The proof of the theorem is done in two steps: First, we show that the sum over the binary problems of the importance weighted regret is at least  $r(y', D|x)$ . Then we apply the costing analysis from importance weighted binary classification to binary classification.

For the first step we use induction, starting at the leaves. The induction hypothesis is that the sum of the regrets of importance-weighted binary classifiers in any subtree bounds the regret of the subtree output.

For node  $n$ , each importance weighted binary decision between class  $a$  and class  $b$  has an importance weighted regret which is either 0 or

$$r_n = |E_{\bar{c} \sim D|x}[c_a - c_b]| = |E_{\bar{c} \sim D|x}[c_a] - E_{\bar{c} \sim D|x}[c_b]|,$$

depending on whether the prediction is correct or not.

Assume without loss of generality that the predictor outputs class  $b$ . The regret of the subtree  $T_n$  rooted at  $n$  is given by

$$r_{T_n} = E_{\bar{c} \sim D|x}[c_b] - \min_{y \in \Gamma(T_n)} E_{\bar{c} \sim D|x}[c_y],$$

where  $\Gamma(T_n)$  denotes the set of leaves in  $T_n$ .

As a base case, the inductive hypothesis is trivially satisfied for trees with one label. Inductively, assume that

$$\sum_{n' \in L} r_{n'} \geq r_L, \quad \sum_{n' \in R} r_{n'} \geq r_R$$

for the left subtree  $L$  of  $n$  (providing  $a$ ) and the right subtree  $R$  (providing  $b$ ).

There are two possibilities. Either the minimizer comes from the leaves of  $L$  or the leaves of  $R$ . The second possibility is easy since we have

$$r_{T_n} = E_{\bar{c} \sim D|x}[c_b] - \min_{y \in \Gamma(R)} E_{\bar{c} \sim D|x}[c_y] = r_R \leq \sum_{n' \in R} r_{n'} \leq \sum_{n' \in T_n} r_{n'},$$

which proves the induction.

For the first possibility, we have

$$\begin{aligned} r_{T_n} &= E_{\bar{c} \sim D|x}[c_b] - \min_{y \in \Gamma(L)} E_{\bar{c} \sim D|x}[c_y] = E_{\bar{c} \sim D|x}[c_b] - E_{\bar{c} \sim D|x}[c_a] + E_{\bar{c} \sim D|x}[c_a] - \min_{y \in \Gamma(L)} E_{\bar{c} \sim D|x}[c_y] \\ &= E_{\bar{c} \sim D|x}[c_b] - E_{\bar{c} \sim D|x}[c_a] + r_L \leq r_n + \sum_{n' \in L} r_{n'} \leq \sum_{n' \in T_n} r_{n'}, \end{aligned}$$

which completes the induction. The inductive hypothesis for the root is that  $r(y', D|x) \leq \sum_{n \in T} r_n$ , implying

$$r(y', D|x) \leq \sum_{n \in T} r_n = (k-1) \cdot r_i(b, \text{Filter-Train}(D)),$$

where  $r_i$  is the importance weighted binary regret on the induced problem. (Recall that  $(k-1)$  is the number of nodes in the filter tree.)

It remains to reduce from importance weighted binary classification to binary classification. Using the folk theorem from [10], we have

$$r_i(b, \text{Filter-Train}(D)) = r(b, D') E_{(x,y,w) \sim D}[w].$$

The expected importance is

$$\frac{1}{k-1} E_{(x,\bar{c}) \sim D} \sum_{n \in T} i_{n,x,\bar{c}}.$$

Plugging this in, we get the theorem.  $\square$

The proof of Theorem 4.3 makes use of the following inequality. Consider a filter tree  $T$  evaluated on a cost-sensitive multiclass instance with cost vector  $c \in [0, 1]^k$ . Let  $S_T$  be the sum of importances over all nodes in  $T$ , and  $I_T$  be the sum of importances over the nodes where the class with the larger cost was selected for the next round. Let  $c_T$  denote the cost of the winner chosen by  $T$ .

**Lemma 4.4.** *For any filter tree  $T$  on  $k$  labels,  $S_T + c_T \leq I_T + \frac{k}{2}$ .*

*Proof.* The inequality follows by induction, the result being clear when  $k = 2$ . Assume that the claim holds for the two subtrees,  $L$  and  $R$ , providing their respective inputs  $l$  and  $r$  to the root of  $T$ , and  $T$  outputs  $r$  without loss of generality. Using the inductive hypotheses for  $L$  and  $R$ , we get  $S_T + c_T = S_L + S_R + |c_r - c_l| + c_r \leq I_L + I_R + \frac{k}{2} - c_l + |c_r - c_l|$ . If  $c_r \geq c_l$ , we have  $I_T = I_L + I_R + (c_r - c_l)$ , and  $S_T + c_T \leq I_T + \frac{k}{2} - c_l \leq I_T + \frac{k}{2}$ , as desired. If  $c_r < c_l$ , we have  $I_T = I_L + I_R$  and  $S_T + c_T \leq I_T + \frac{k}{2} - c_r \leq I_T + \frac{k}{2}$ , completing the proof.  $\square$

**Proof of Theorem 4.3** We will fix  $(x, \vec{c}) \in X \times [0, 1]^k$  and take the expectation over the draw of  $(x, \vec{c})$  from  $D$  as the last step.

Consider a filter tree  $T$  evaluated on  $(x, \vec{c})$  using a given binary classifier  $b$ . As before, let  $S_T$  be the sum of importances over all nodes in  $T$ , and  $I_T$  be the sum of importances over the nodes where  $b$  made a mistake. Recall that the regret of  $T$  on  $(x, \vec{c})$ , denoted in the proof by  $\text{reg}_T$ , is the difference between the cost of the tree's output and the smallest cost  $c^*$ . The importance-weighted binary regret of  $b$  on  $(x, \vec{c})$  is simply  $I_T/S_T$ . Since the expected importance is upper bounded by 1,  $I_T/S_T$  also bounds the binary regret of  $b$ .

The inequality we need to prove is  $\text{reg}_T S_T \leq \frac{k}{2} I_T$ . The proof is by induction on  $k$ , the result being trivial if  $k = 2$ . Assume that the assertion holds for the two subtrees,  $L$  and  $R$ , providing their respective inputs  $l$  and  $r$  to the root of  $T$ . (The number of classes in  $L$  and  $R$  can be taken to be even, by splitting the odd class into two classes with the same cost as the split class, which has no effect on the quantities in the theorem statement.)

Let the best cost  $c^*$  be in the left subtree  $L$ .

Suppose first that  $T$  chooses  $r$  and  $c_r > c_l$ . Let  $w = c_r - c_l$ . We have  $\text{reg}_L = c_l - c^*$  and  $\text{reg}_T = c_r - c^* = \text{reg}_L + w$ . The left hand side of the inequality is thus

$$\begin{aligned} \text{reg}_T S_T &= (\text{reg}_L + w)(S_R + S_L + w) = w(\text{reg}_L + S_R + S_L + w) + \text{reg}_L(S_L + S_R) \\ &\leq w \left( \text{reg}_L + I_R + I_L - c_r - c_l + w + \frac{k}{2} \right) + \text{reg}_L \left( I_R + I_L - c_l - c_r + \frac{k}{2} \right) \\ &\leq \frac{k}{2} w + I_R(w + \text{reg}_L) + I_L(w + \text{reg}_L) + \text{reg}_L \left( \frac{k}{2} - c_r - c_l \right) \\ &\leq \frac{k}{2} w + I_R(w + \text{reg}_L) + I_L \left( w + \text{reg}_L + \frac{k}{2} - c_r - c_l \right) \\ &\leq \frac{k}{2} w + I_R(w + \text{reg}_L) + \frac{k}{2} I_L \leq \frac{k}{2} (w + I_R + I_L) = \frac{k}{2} I_T \end{aligned}$$

The first inequality follows from lemma 4.4. The second and fourth follow from  $w(\text{reg}_L - c_l - c_r + w) \leq 0$ . The third follows from  $\text{reg}_L \leq I_L$ . The fifth follows from  $\text{reg}_T \leq \frac{k}{2}$  for  $k \geq 2$ .

The proofs for the remaining three cases ( $c_T = c_l < c_r$ ,  $c_T = c_l > c_r$ , and  $c_l > c_r = c_T$ ) use the same machinery as the proof above; complete details can be found in the appendix.

Taking the expectation over  $(x, \vec{c})$  completes the proof.  $\square$

## 5 Experimental Results

There is a variant of the filter tree algorithm, which has a significant difference in performance in practice. Every classification at any node  $n$  is essentially between two labels computed at test time, implying that we could simply learn one classifier for every pair of labels that could reach  $n$  at test time. (Note that a given pair of labels can be compared only at a single node, namely their least common ancestor in the tree.) The conditioning process and the tree structure gives us a better analysis than is achievable with the All-pairs algorithm. This variant uses more computation and requires more data but often maximizes performance when the form of the classifier is constrained.

We compared the performance of Filter Tree and its All-Pairs variant described above to the performance of All-Pairs and the Tree reduction, on a number of publicly available multiclass datasets [11]. Some datasets came with a standard training/test split. For all other datasets, we reported the average result over 10 random splits, with 2/3 of the dataset used for training and 1/3 for testing. All datasets tested are reported. More details are provided in the appendix.

If the computation is constrained and we can afford only  $O(\log k)$  evaluations of a binary classifier per multiclass prediction, the Filter Tree is typically as good as or better than the Tree reduction, as shown in Figure 2 (left).

If the computation is relatively unconstrained, plausible choices include All-Pairs and the All-Pairs Filter Tree. The comparison in Figure 2 (right) shows that there is no real dominance between the algorithms, while the All-Pairs Filter Tree uses  $O(k)$  evaluations instead of  $O(k^2)$ .

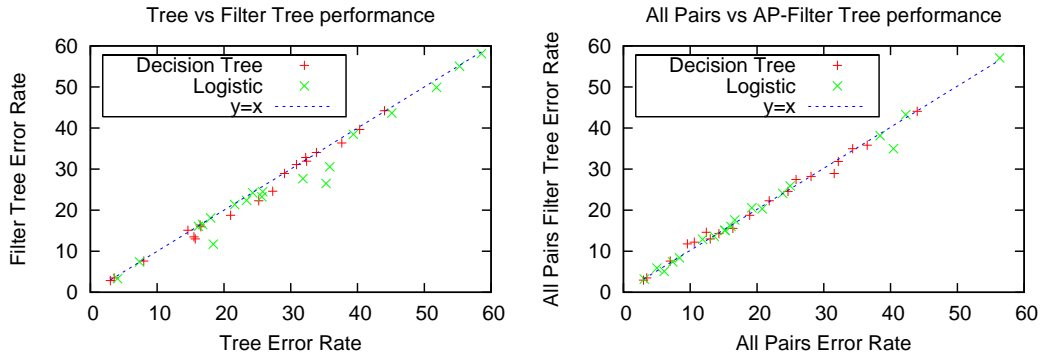


Figure 2: Error rates (in %) of Tree versus Filter-Tree (left) and All-Pairs versus All-Pairs Filter Tree (right) on several different datasets with a decision tree or logistic regression classifier.

For cost-sensitive multiclass problems, there is another variant that may improve performance in practice. The essential observation is that predicting well at most of the nodes in the filter tree is irrelevant. In particular, the prediction at any node not on the path from the root to the predicted leaf is irrelevant. Given this observation, it may be possible to improve performance by optimizing prediction on the relevant nodes at the expense of prediction on the irrelevant nodes. This can be done using iterative approaches which gradually shift importance onto the most relevant nodes.

## 6 Discussion

We show that there exists a tight consistent method for reducing multiclass classification to binary classification. All previous methods for multiclass to binary reduction have either an incomparable or dominated analysis; see the table in the introduction. For cost-sensitive classification, the guarantees provided by the filter tree are superior to all previous approaches.

There are several questions left. Error correcting codes have transform properties *independent* of  $k$  (and a poor dependence on the binary regret). The ability to error correct seems to be an intrinsic necessity to eliminate the dependence on  $k$ . We conjecture that this dependence can be removed by using  $\log_2 k$ -elimination tournaments. The losers of the single elimination tournament play among another tournament, and so on until all classes but one have suffered at least  $\log_2 k$  losses.

All previous approaches for reducing cost sensitive multiclass classification had a dependence on the expected sum of costs, so the improved dependence on the expected sum of cost differences and an alternative bound of  $k/2$  is striking for the filter tree. The worst-case dependence on  $k/2$  is asymptotically optimal.

## References

- [1] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers, ICML 2000.
- [2] A. Beygelzimer, J. Langford, and B. Zadrozny. Weighted one against all, AAI 2005.
- [3] A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Reductions between classification tasks, ICML 2005.
- [4] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
- [5] J. Fox. *Applied Regression Analysis, Linear Models, and Related Methods*, 1997.
- [6] T. Hastie and R. Tibshirani. Classification by pairwise coupling, NIPS 1997.
- [7] B. Kijssirikul, N. Ussivakul, S. Meknavin. Adaptive directed acyclic graphs for multiclass classification, PRICAI 2002.
- [8] J. Langford and A. Beygelzimer. Sensitive Error Correcting Output Codes, COLT 2005.

- [9] J. Platt, N. Christianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification, NIPS 2000.
- [10] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting, ICDM 2003.
- [11] C. Blake and C. Merz, UCI Repository of machine learning databases.
- [12] I. Witten and E. Frank. Data Mining: Practical machine learning tools with Java implementations, 2000: <http://www.cs.waikato.ac.nz/ml/weka/>.

## A Inconsistency and Frailty of Other Approaches

In this section, we formalize the claims made in the introduction about the inconsistency and frailty of other approaches.

### A.1 Inconsistency of the Tree reduction

One standard approach for solving multiclass classification with a binary classifier learner is to split the set of labels in half, learn a binary classifier to distinguish between the subsets, and repeat recursively until each subset contains only one label. Multiclass predictions are made according to the leaf for which all binary predictors above the leaf in the tree prefer the leaf.

The fundamental drawback of this approach is that it is inconsistent.

The following theorem shows that there exist multiclass problems such that if we have an optimal binary classifier for the induced distribution  $\text{Tree}(D)$ , the reduction does not yield an optimal multiclass predictor. The proof is constructive, and the intuition closely follows the similar theorem for error correcting output codes [8].

**Theorem A.1.** *For all  $k \geq 3$ , for all binary trees over the labels, there exists a multiclass distribution  $D$  such that*

$$r(\text{Tree}(b^*, \cdot), D) > 0$$

for any  $b^* = \arg \min_b e(b, \text{Tree}(D))$ .

*Proof.* Find a node with one subset corresponding to two labels and the other subset corresponding to a single label. (If the tree is perfectly balanced, simply let  $D$  assign probability 0 to one of the labels.) Since we can freely rename labels without changing the underlying problem, let the first two labels be 1 and 2, and the third label be 3.

We choose a distribution  $D$  with the property that labels 1 and 2 have a slightly larger than 1/4 chance of being drawn given  $x$ :

$$D(y = 1 \mid x) = D(y = 2 \mid x) = 1/4 + 1/100,$$

while label 3 has a probability slightly less than 1/2:

$$D(y = 3 \mid x) = 1/2 - 2/100.$$

Under this distribution, the fraction of examples for which label 1 or 2 is correct is  $1/2 + 2/100$ , so any minimum error rate binary predictor must choose either label 1 or label 2. Each of these choices has an error rate of  $3/4 - 1/100$ . The optimal multiclass predictor chooses label 3 and suffers an error rate of  $1/2 + 2/100$ , implying that the regret of the tree classifier based on an optimal binary classifier is  $1/4 - 3/100 > 0$ .  $\square$

### A.2 Frailty of the All-Pairs Reduction

The All-Pairs reduction *is* consistent (as an application of a theorem in [1]). However, as a reduction it may be relatively frail against difficult binary classifier and problem pairs.

The All-Pairs reduction starts by constructing  $\binom{k}{2}$  binary classifiers, one for every pair of classes  $(i, j)$ . Given a training dataset  $S = \{(x, y)\}$ , the binary classifier for the  $(i, j)$ -class pair is trained with dataset  $\{(x, I(y = i)) : (x, y) \in S \text{ and } y = i \text{ or } y = j\}$ . Thus each binary classifier is trained

to discriminate between two particular classes. Given a test example, each of the binary classifiers predicts a winner amongst its two classes, and the class with the highest number of wins is chosen as the multiclass prediction, with ties broken randomly.

Letting All-Pairs( $D$ ) denote the induced binary distribution, we have the following observation.

**Theorem A.2.** *For all  $k \geq 2$ , there exists binary classifiers  $b$  and multiclass distributions  $D$  such that*

$$r(\text{All-Pairs}(b, \cdot), D) = (k - 1)r(b, \text{All-Pairs}(D)).$$

The frailty which this theorem demonstrates has *not* been strongly exhibited by experimental evidence; see the main draft.

*Proof.* The proof is constructive. Pick any distribution  $D$  with a deterministic dependence between  $y$  and  $x$ . Conditioned on  $x$  suppose that  $y = 1$  is the probability 1 label. The All Pairs reduction only produces binary examples with the property that  $i = 1$  or  $j = 1$ , which implies that the classifier suffers no regret for predictions that do not involve label 1. There are  $k - 2$  predictions involving label 2 but not label 1, and in each of these the classifier might predict that label 2 is the winner. By suffering regret  $\frac{1}{k-1}$  (1 error out of  $k - 1$ ) for mispredicting the class 1 and 2 decision, class 2 can have  $k - 1$  wins implying that it is the winner overall, inducing multiclass regret 1.  $\square$

## B Addendum to the Proof of Theorem 4.3

For completeness, we analyze the remaining three cases appearing in the proof of Theorem 4.3. The proofs use the same machinery and are very similar to the proof of the first case presented in the main text. Refer to the main text for notation.

**Case 2:**  $T$  outputs  $l$ , and  $c_l < c_r$ . In this case  $\text{reg}_T = \text{reg}_L = c_l - c^*$ . The left hand side can be rewritten as

$$\begin{aligned} \text{reg}_T S_T &= \text{reg}_L(S_R + S_L + c_r - c_l) = \text{reg}_L S_L + \text{reg}_L(S_R + c_r - c_l) \\ &\leq \text{reg}_L \left( I_L + I_R - 2c_l + \frac{k}{2} \right) \\ &\leq I_R + \text{reg}_L \left( I_L - 2c_l + \frac{k}{2} \right) \\ &\leq I_R + I_L \left( \text{reg}_L - 2c_l + \frac{k}{2} \right) \\ &\leq I_R + \frac{k}{2} I_L \\ &\leq \frac{k}{2} I_T \end{aligned}$$

The first inequality from the lemma, the second from  $\text{reg}_L \leq 1$ , the third from  $\text{reg}_L \leq I_L$ , the fourth from  $-c_l - c^* < 0$ , and the fifth because  $I_T = I_L + I_R$ .

**Case 3:**  $T$  outputs  $l$ , and  $c_l > c_r$ . We have  $\text{reg}_T = \text{reg}_L = c_l - c^*$ . The left hand side can be written as

$$\begin{aligned} \text{reg}_T S_T &= \text{reg}_L(S_R + S_L + c_l - c_r) \leq \frac{|L|}{2} I_L + \text{reg}_L \left( I_R + \frac{k - |L|}{2} - c_r + c_l - c_r \right) \\ &\leq \frac{k}{2} I_L + I_R + (c_l - 2c_r) \leq \frac{k}{2} (I_L + I_R + (c_l - c_r)) = \frac{k}{2} I_T, \end{aligned}$$

The first inequality follows from the inductive hypothesis and the lemma, the second from  $\text{reg}_L < 1$  and  $\text{reg}_L < I_L$ , and the third from  $c_r > 0$  and  $k/2 > 1$ .

**Case 4:**  $T$  outputs  $r$ , and  $c_l > c_r$ . Let  $w = c_l - c_r$ . We have  $\text{reg}_T = c_r - c^* = \text{reg}_L - w$ . The left hand side can be written as

$$\begin{aligned} \text{reg}_T S_T &= (\text{reg}_L - w)(S_R + S_L + w) = \text{reg}_L S_L - w S_L + (\text{reg}_L - w)(S_R + w) \\ &\leq \frac{|L|}{2} I_L - w \left( I_L + \frac{|L|}{2} - c_l \right) + (\text{reg}_L - w) \left( I_R + c_l - 2c_r + \frac{k - |L|}{2} \right) \\ &\leq \frac{|L|}{2} I_L - w \left( I_L + \frac{|L|}{2} - c_l \right) + (I_L - w) \frac{k - |L|}{2} + (\text{reg}_L - w) (I_R + c_l - 2c_r) \\ &\leq \frac{k}{2} (I_L + I_R) - w \frac{k}{2} - w (I_L - c_l) + (\text{reg}_L - w) (c_l - 2c_r). \end{aligned}$$

The first inequality follows from the inductive hypothesis and the lemma, the second from  $\text{reg}_L \leq I_L$ , and the third from  $\text{reg}_L \leq \frac{k}{2}$ .

The last three terms are upper bounded by

$$\begin{aligned} &-w - w \text{reg}_L + w c_l + \text{reg}_L c_l - 2c_r \text{reg}_L - w c_l + 2w c_r \\ &\leq -w - \text{reg}_L (c_r + c_l) + \text{reg}_L c_l + 2w c_r \\ &\leq -w - (c_l - c^*) c_r + w c_r + (c_l - c_r) c_r \leq 0, \end{aligned}$$

and thus can be ignored, yielding  $\text{reg}_T S_T \leq \frac{k}{2} (I_L + I_R) = \frac{k}{2} I_T$ , which completes the proof.

## B.1 Lower Bound

The following simple example shows that the theorem is essentially tight in the worst case.

Let  $k$  be a power of two, and let every label have cost 0 if it is even, and 1 otherwise. The tree structure is a complete binary tree of depth  $\log k$  with the nodes being paired in the order of their labels. Suppose that all pairwise classifications are correct, except for class  $k$  wins all its  $\log k$  games leading to cost-sensitive multiclass regret 1. If  $T$  is the resulting filter tree, we have  $\text{reg}_T = 1$ ,  $S_T = \frac{k}{2} + \log k - 1$ , and  $I_T = \log k$ , leading to the regret ratio of

$$\frac{\text{reg}_T S_T}{I_T} \leq \frac{k/2 + \log k - 1}{\log k} = \Omega\left(\frac{k}{2 \log k}\right),$$

almost matching the theorem's bound of  $\frac{k}{2}$  the regret ratio.

## C Experimental Results (Details)

We compared the performance of Filter Trees and the All-Pairs variant of Filter Trees to the performance of All-Pairs and the Tree reduction, on a number of publicly available multiclass datasets [11].

Some datasets came with a standard training/test split: isolet (isolated letter speech recognition), optdigits (optical handwritten digit recognition), pendigits (pen-based handwritten digit recognition), satimage, and soybean. For all other datasets, we reported the average result over 10 random splits, with  $2/3$  of the dataset used for training and  $1/3$  for testing. (The splits were the same for all methods.)

Test error rates using decision trees (J48) and logistic regression as binary classifier learners are reported in Table 1. We used Weka [12] implementation of both learners (with default parameters). No optimization or parameter tuning was performed. (In fact, some datasets have significantly better error rates reported using other, more involved methods.) The lowest error rate in each row is shown in bold (ignoring the difference, which in some cases is insignificant).

The experiments show that the filter tree approach is generally superior to the tree approach in practice; both require  $O(\log k)$  evaluation per multiclass prediction. When computation is relatively unconstrained, the comparison of all-pairs and all-pairs filter tree shows that neither dominate in practice.

Dataset	J48				Logistic Regression				Properties	
	Tree	FT	AP	APFT	Tree	FT	AP	APFT	$k$	Size
arrhythmia	37.64	36.37	<b>34.32</b>	34.97	55.27	55.04	40.44	<b>34.97</b>	13	452
audiology	32.37	31.93	<b>28.08</b>	28.21	31.83	27.69	<b>24.98</b>	25.90	24	226
ecoli	21.00	<b>18.75</b>	18.90	<b>18.75</b>	18.00	18.10	15.20	<b>15.06</b>	8	336
flare	16.42	16.38	16.38	<b>15.57</b>	16.17	16.07	16.09	<b>16.03</b>	7	1,388
glass	33.84	34.02	32.18	<b>31.86</b>	39.37	38.46	38.43	<b>38.13</b>	6	214
isolet	27.30	24.60	<b>12.40</b>	14.60	35.30	26.50	<b>8.40</b>	<b>8.40</b>	26	7,797
kropt	40.32	39.66	36.50	<b>35.81</b>	58.55	58.09	<b>56.34</b>	57.06	18	28,056
letter	16.53	15.96	<b>9.58</b>	11.77	51.84	49.89	<b>16.66</b>	17.62	25	20,000
lymph	25.22	22.28	<b>21.83</b>	22.28	24.32	24.20	<b>23.86</b>	24.07	4	148
mfeat-zernike	32.24	32.81	<b>25.84</b>	27.45	25.68	23.24	<b>19.21</b>	20.56	10	2000
nursery	3.55	<b>3.49</b>	<b>3.49</b>	<b>3.49</b>	<b>7.36</b>	7.41	7.39	7.39	5	12,960
optdigits	15.50	13.50	<b>10.60</b>	12.20	18.40	11.70	<b>5.00</b>	5.90	10	5,620
page-blocks	2.99	<b>2.84</b>	3.00	2.95	4.06	3.31	<b>3.12</b>	3.21	5	5,473
pendigits	8.00	7.60	<b>7.00</b>	7.60	23.40	22.40	6.10	<b>5.10</b>	10	10,992
satimage	14.60	15.10	<b>14.30</b>	<b>14.30</b>	25.80	24.50	15.20	<b>15.10</b>	6	6,435
soybean	15.70	<b>13.00</b>	<b>13.00</b>	<b>13.00</b>	16.80	16.50	<b>13.60</b>	<b>13.60</b>	19	683
vehicle	30.86	31.11	31.57	<b>28.93</b>	21.60	21.37	20.78	<b>20.31</b>	4	846
vowel	29.06	28.92	24.64	<b>24.57</b>	35.85	30.53	<b>11.85</b>	12.90	11	990
yeast	44.04	44.21	<b>43.99</b>	44.06	45.13	43.66	<b>42.28</b>	43.26	10	1,484

Table 1: Test error rates (in %) using J48 and logistic regression as binary learners. AP and FT stand for All-Pairs and Filter Tree respectively. APFT is the All-Pairs variant of the Filter Tree.