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
pradeepr@cs.cmu.edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Abstract

We present a new algorithm, filter tree, for reducing (cost-sensitive) 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.)

The filter tree is *robust*. It suffers multiclass regret at most $\log_2 k$ times the binary regret. The filter tree 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 and many multiclass classification problems, a common approach has been to create meta-algorithms which use binary classifiers to make multiclass predictions.

Table 1 summarizes the characteristics of some known reductions from multiclass to binary classification. 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.

We are also interested in the *regret* of a classifier defined as the difference between its error rate and the smallest possible error rate on the same problem. Regret measures avoidable loss, and thus it is often a more meaningful notion for "hard" problems with high inherent noise. In the table, r is the average binary regret and the `Regret` column is an upper bound on the multiclass regret in terms of r. The entries with "none" indicate that no regret transform provably exists. This says that given an optimal (i.e., regret-zero) binary classifier, the reduction does not yield an optimal multiclass classifier in the presence of noise. Equivalently, 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 [9]	$4\sqrt{e}$	$4\sqrt{r}$	large
Filter Tree	$\lceil \log_2 k \rceil e$	$\lceil \log_2 k \rceil r$	$\lceil \log_2 k \rceil$

Table 1: A comparison of different reductions from multiclass to binary classification

In the table, WOA stands for Weighted One-Against-All, which is an improved version of One-Against-All [2]. ECOC is the Error Correcting Output Code reduction [4], and PECOC is a probabilistic version of ECOC from [9].

Tree (see, for example, [5]) is a divide-and-conquer technique, which distinguishes between the labels using a binary tree. The root node contains the entire label set $\{1, \ldots, k\}$. Starting from the root, the set of labels at each internal node is recursively split in half, and a classifier is trained to distinguish between the two subsets.

The All-Pairs reduction [6] works by learning a classifier to predict, for each pair of classes (i, j): "Is label i more probable than label j?" To make a multiclass prediction, the reduction runs the $\binom{k}{2}$ classifiers and outputs 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. ECOC has a better error transform bound, but it is inconsistent. PECOC has a better dependence on the number of labels k, but a 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 [9] (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, yielding a better bound.

The filter tree algorithm is similar to Adaptive Directed Acyclic Graphs [7], which were introduced as a computational optimization on Decision Directed Acyclic Graphs [10], 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.

The filter tree algorithm is presented in Section 3. Section 4 provides the analysis. In Section 5 we contrast our algorithm with existing methods. Section 6 concludes with experimental results.

2 Definitions

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

$$e(c, P) = \mathbf{Pr}_{(x,y)\sim P}[c(x) \neq y].$$

The *classification regret* of c on P is defined as

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

The motivation for regret analysis is to separate avoidable loss from loss inherent to the problem. The resulting bounds thus apply nontrivially for problems with large inherent noise.

Binary classification corresponds to the $k = 2$ case.

A *cost-sensitive k-class classification problem* is defined by a distribution D over $\mathbb{X} \times [0, \infty)^k$. The goal is to find a classifier $h : \mathbb{X} \to \{1, ..., k\}$ minimizing the expected cost

$$e(h, D) = E_{(x,\vec{c})\sim D}\left[c_{h(x)}\right].$$

Here $\vec{c} \in [0, \infty)^k$ gives the cost of each of the k choices for x. Similarly to the multiclass case, the regret of h on D is defined as $r(h, D) = e(h, D) - \min_{h^*} e(h^*, D)$.

In *importance-weighted binary classification*, predicting some examples correctly is more important than predicting others. The problem is specified by a distribution D on $X \times \{0, 1\} \times [0, \infty)$ and loss function $E_{(x,y,w)\sim D}\left[wI(b(x) \neq y)\right]$, 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, essentially 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. Prediction problems associated with the nodes are different, however. Working from the leaves toward the root, each internal node predicts which of its two inputs is more likely, given the features.

In the first round, the labels are paired according to the graph, 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. This process repeats until an overall winner is declared at the root.

Algorithm 1 The filter tree training algorithm

`Filter-Tree-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 $(x, c_1, ..., c_k) \in S$
      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 $\text{predict}_n = \texttt{Learn}(S_n)$
3. return $\{\text{predict}_n\}$

Algorithm 2 The filter tree testing algorithm

`Filter-Tree-Test` (classifiers $\{\text{predict}_n\}$, test example $x \in \mathbb{X}$)

Output the label l such that every classifier on the path from leaf l to the root prefers label l.

The filter tree algorithm comes in two parts: training (Algorithm 1) and testing (Algorithm 2). The actual predictions form a filter tree, as shown in Figure 1. The algorithm here is stated for cost-sensitive multiclass prediction. The multiclass variant is formed by projecting multiclass examples (x, y) into cost sensitive examples $(x, \vec{c})$ where $c_y = 0$, and for all $y' \neq y$, $c_{y'} = 1$. The filter tree algorithm relies upon an importance weighted binary learning

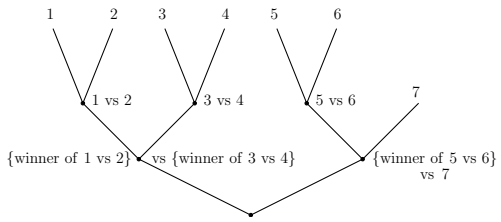

Figure 1: Filter Tree. Each node predicts whether the left or the right input label is more likely, conditioned on a given $x \in \mathbb{X}$. The final output node predicts the best label for x.

algorithm, which takes examples of the form (x, y, i), where x is a feature vector used for prediction, y is a binary label, and i is a positive real-valued importance. Importance-weighted binary classification can be further reduced to binary classification using the costing reduction [11] or other methods.

One important detail in Algorithm 1 is forming the right training set to distinguish between the labels. Each training example for node n is formed *conditionally* on the predictions of the classifiers on the path from n to the true label. 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 testing algorithm is very simple. We just predict which input to the root node has the best label, then go to the node predicting that output, and repeat until a leaf is reached, determining the multiclass prediction.

There are several variants of this algorithm, which may have significant differences in performance in practice.

1. `All-pairs variant`: 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 (confirmed experimentally; see Section 6).
2. `Matching measures variant`: For the cost-sensitive multiclass version of the tree, there is room for improvement. 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. Iterative approaches which gradually shift importance onto the most relevant nodes may help.

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-Tree-Train` transforms cost-sensitive multiclass examples into importance-weighted binary examples. This process implicitly transforms a distribution D over examples for the original problem into a distribution `Filter-Tree`(D) over examples for the problem we reduce to. There are many induced problems, one for each call to the oracle. However, there is 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 al-

gorithm produces a single binary classifier predict(x, n) for which we can redefine predict$_n(x)$ as predict(x, n). Given this, we can define the induced distribution `Filter-Tree`(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 $((x, n), y, i)$ according to line 2(a)(iii), except with n added into the features.

This one-classifier trick may appear suspect because classifiers are conditionally dependent on other classifiers closer to the leaves. In practice, there are known effective iterative techniques for dealing with this cyclic dependence. In theory, we quantify over all predictors, which means that we can regard the learned predictor as fixed (i.e., independent of the drawn samples), implying that the induced problem is well defined.

When reducing to importance-weighted classification, the theorem changes as importance weights change. To remove the importances, we compose the reduction with the Costing reduction [11] (by altering the underlying distribution using rejection sampling on the importance weight).

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,*

$$r(\texttt{Filter-Tree}(b, .), D)$$

$$\leq r(b, \texttt{Filter-Tree}(D)) E_{x,\vec{c}\sim D} \sum_{n \in T} i_{n,x,\vec{c}}$$

where $i_{n,x,\vec{c}}$ is the importance weight given by 2(a)(iii).

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,*

$$r(\texttt{Filter-Tree}(b, .), D)$$

$$\leq r(b, \texttt{Filter-Tree}(D)) \lceil \log_2 k \rceil$$

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, $\sum_n i_{n,x,\vec{c}} \leq \lceil \log_2 k \rceil$.

4.1 Proof of Main Theorem

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 `Filter-Tree`$(b, x) = y'$, 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:

1. Show that the sum over the binary problems of the importance weighted regret is at least $r(y', D|x)$.
2. 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

$$\begin{aligned} r_n &= |E_{\vec{c}\sim D|x}[c_a - c_b]| \\ &= |E_{\vec{c}\sim D|x}[c_a] - E_{\vec{c}\sim D|x}[c_b]|, \end{aligned}$$

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

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

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

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

which proves the induction.

For the first possibility, we have

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

which completes the induction. The inductive hypothesis for the root is that

$$r(y', D|x) \leq \sum_{n\in T} r_n,$$

implying

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

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

The remainder of the proof is about the reduction from importance weighted binary classification to binary classification. The folk theorem proved in [11] says that for all binary classifiers b and importance weighted binary problems D' we have:

$$r_i(b, D) = r(b, D')E_{x,y,i\sim D}[i]$$

where r_i is the importance weighted binary regret and D' is the induced binary distribution.

For the filter tree reduction, the expected importance is

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

Plugging this in, we get the theorem.

5 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.

5.1 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 $\texttt{Tree}(D)$, 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 [9].

Theorem 5.1. *For all $k \geq 3$, for all binary trees over the labels, there exists a multiclass distribution D such that*

$$r(\texttt{Tree}(b^*, .), D) > 0$$

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

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 $1/4 - 3/100 > 0$. □

5.2 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 $\texttt{All-Pairs}(D)$ denote the induced binary distribution, we have the following observation.

Theorem 5.2. *For all $k \geq 2$, there exists binary classifiers b and multiclass distributions D such that*

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

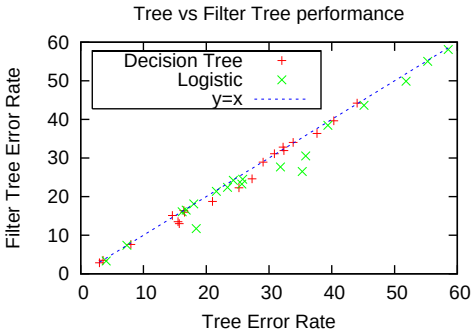

Figure 2: Performance comparison of `Tree` versus `Filter-Tree` on several different datasets with a decision tree or logistic regression classifier. The Filter Tree appears to perform better in practice.

The frailty which this theorem demonstrates has *not* been strongly exhibited by experimental evidence, as seen in Section 6.

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. □

6 Experimental Results

We compared the performance of Filter Trees and the All-Pairs variant of Filter Trees (discussed at the end of Section 3) to the performance of All-Pairs and the Tree reduction, on a number of publicly available multiclass datasets [12].

Table 4 gives the basic properties of the datasets tested. 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.)

We used decision trees (J48) and logistic regression as binary classifier learners, both available within Weka [13] (default parameters were used). We did not perform any optimization or parameter tuning. (Indeed, some datasets have significantly better error rates reported using other, more computationally involved methods.) Our objective in performing these experiments was to provide a basic sanity check for the analysis presented in the paper, across a variety of different datasets.

Test set error rates using each of the binary classifier learners are shown in Tables 2 and 3. The lowest error rate in each row is shown in bold (ignoring the difference, which in some cases is insignificant).

There are two different settings to consider.

1. Computation constrained. We might be able to afford only $O(\log k)$ evaluations of a binary classifier. For this setting, the Filter Tree is typically as good as or better than the Tree reduction as shown in Figure 2. (Table 2 provides the numbers.)

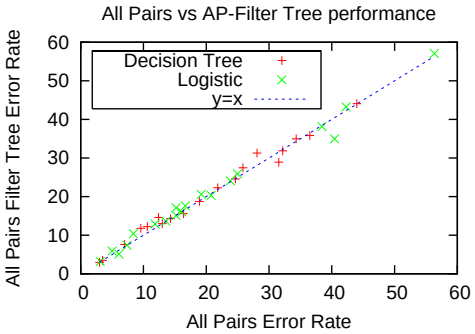

Figure 3: Performance comparison of All-Pairs versus All-Pairs Filter Tree on several different datasets with a decision tree or logistic regression classifier. There is no clear dominance.

2. Computation unconstrained. In this setting, plausible algorithms include All-Pairs (which is $O(k^2)$) and the All-Pairs Filter Tree (which is $O(k)$). Figure 3 compares these approaches, where we discover no real dominance between the algorithms. (Table 3 provides the numbers.)

7 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 filter tree reduction analysis is simply superior to all previous approaches. Our experiments show that the filter tree approach is generally superior to the tree approach in practice. When computation is unconstrained, the comparison of all-pairs and all-pairs filter tree shows that neither dominate in practice.

Two mysteries arise here.

1. 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 achieving this sorts of analysis. Is there an error correcting variant of the filter tree which can achieve an independent-of-k analysis? The natural extension of the filter is an algorithm which repeatedly compares classes with a similar number of losses until all classes but one have suffered at least n losses.
2. 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 is striking for the filter tree. Nevertheless, there is significant room for improvement because the expected sum of cost differences can be $O(k)$ even when the costs are bounded $[0, 1]$. Is an $O(k)$ dependence necessary, or is it possible to reduce cost sensitive multiclass to binary just as well as we can reduce multiclass to binary?

References

[1] Erin Allwein, Robert Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers, ICML 2000.

[2] Alina Beygelzimer, John Langford, and Bianca Zadrozny. Weighted One Against All, AAAI 2005.

[3] Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, and Bianca Zadrozny. Reductions between classification tasks, ICML 2005.

[4] Thomas Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Dataset	Tree	FT	All-Pairs	APFT
arrhythmia	37.64	36.37	**34.32**	34.97
audiology	32.37	31.93	**28.08**	31.30
ecoli	21.00	**18.75**	18.90	**18.75**
flare	16.42	16.38	16.38	**15.57**
glass	33.84	34.02	32.18	**31.86**
isolet	27.30	24.60	**12.40**	14.60
kropt	40.32	39.66	36.50	**35.81**
letter	16.53	15.96	**9.58**	11.77
lymph	25.22	22.28	**21.83**	22.28
mfeat-zernike	32.24	32.81	**25.84**	27.45
nursery	3.55	**3.49**	**3.49**	**3.49**
optdigits	15.50	13.50	**10.60**	12.20
page-blocks	2.99	**2.84**	3.00	2.95
pendigits	8.00	7.60	**7.00**	7.60
satimage	14.60	15.10	**14.30**	**14.30**
soybean	15.70	**13.00**	**13.00**	**13.00**
vehicle	30.86	31.11	31.57	**28.93**
vowel	29.06	28.92	24.64	**24.57**
yeast	44.04	44.21	**43.99**	44.06

Table 2: Test error rates (in %) using J48 as the binary learner. The last column, APFT, corresponds to the all-pairs variant of the Filter Tree discussed in Section 3.

[5] John Fox. *Applied Regression Analysis, Linear Models, and Related Methods*, Sage Publications, 1997.

[6] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling, *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*, 507–513, 1998.

[7] Boonserm Kijsirikul, Nitiwut Ussivakul, Surapant Meknavin. Adaptive directed acyclic graphs for multiclass classification, PRICAI 2002.

[8] Eun Bae Kong and Thomas G. Dietterich, Error-Correcting Output Coding corrects bias and variance, ICML 1995.

[9] John Langford and Alina Beygelzimer. Sensitive Error Correcting Output Codes, COLT 2005.

[10] John Platt, Nello Christianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification, NIPS 2000.

[11] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting, ICDM 2003.

[12] C. Blake and C. Merz, UCI Repository of machine learning databases, University of California, Irvine.

[13] Ian H. Witten and Eibe Frank. Data Mining: Practical machine learning tools with Java implementations, 2000:
`http://www.cs.waikato.ac.nz/ml/weka/`.

Dataset	Tree	FT	All-Pairs	APFT
arrhythmia	55.27	55.04	40.44	**34.97**
audiology	31.83	27.69	**24.98**	25.90
ecoli	18.00	18.10	**15.20**	17.10
flare	16.17	16.07	16.09	**16.03**
glass	39.37	38.46	38.43	**38.13**
isolet	35.30	26.50	**8.40**	10.40
kropt	58.55	58.09	**56.34**	57.06
letter	51.84	49.89	**16.66**	17.62
lymph	24.32	24.20	**23.86**	24.07
mfeat-zernike	25.68	23.24	**19.21**	20.56
nursery	**7.36**	7.41	7.39	7.39
optdigits	18.40	11.70	**5.00**	5.90
page-blocks	4.06	3.31	**3.12**	3.21
pendigits	23.40	22.40	6.10	**5.10**
satimage	25.80	24.50	15.20	**15.10**
soybean	16.80	16.50	**13.60**	**13.60**
vehicle	21.60	21.37	20.78	**20.31**
vowel	35.85	30.53	**11.85**	12.90
yeast	45.13	43.66	**42.28**	43.26

Table 3: Test error rates (in %) using logistic regression as the binary learner. The last column, APFT, corresponds to the all-pairs variant of the Filter Tree discussed in Section 3.

Dataset	k	Examples	H
arrhythmia	13	452	0.65
audiology	24	226	0.75
ecoli	8	336	0.73
flare	7	1,388	0.31
glass	6	214	0.84
isolet	26	7,797	1.00
kropt	18	28,056	0.84
letter	25	20,000	1.00
lymph	4	148	0.61
mfeat-zernike	10	2000	1.00
nursery	5	12,960	0.74
optdigits	10	5,620	1.00
page-blocks	5	5473	0.27
pendigits	10	10,992	1.00
satimage	6	6,435	0.96
soybean	19	683	0.90
vehicle	4	846	0.99
vowel	11	990	1.00
yeast	10	1,484	0.75

Table 4: Basic properties of the datasets. To give some sense of balance of the empirical class distribution P, we list the value of $H = H(P)/\log(k)$, where $H(P)$ is the empirical entropy of P.