Resource Constrained Stream Mining With Classifier Tree Topologies

Brian Foo, Deepak S. Turaga, Olivier Verscheure, Mihaela van der Schaar, and Lisa Amini

***Abstract*—Stream mining applications require the identification of several different attributes in data content and hence rely on a distributed set of cascaded statistical classifiers to filter and process the data dynamically. In this letter, we introduce a novel methodology for configuring cascaded classifier topologies, specifically binary classifier trees, with optimized operating points after jointly considering the misclassification cost of each end-to-end class of interest in the tree, the resource constraints for every classifier, and the confidence level of each data object that is classified. By configuring multiple operating points per classifier, we enable not only intelligent load shedding when resources are scarce but also intelligent replication of low confidence data across multiple edges when excess resources are available. Using a classifier tree constructed from support vector machine-based sports image classifiers, we verify huge cost savings and discuss how different classifier placements and costs can influence the gains obtained by various algorithms.**

***Index Terms*—Binary classifier tree, networked classifiers, resource management, stream mining.**

I. Introduction

THE processing and classification of continuous, high-volume data streams is of paramount importance for many applications including online financial analysis, real-time manufacturing process control, search engines, spam filters, security, and medical services [4], [6], [7], etc. Due to the naturally distributed set of data sources and jobs, as well as high computational burdens for the analytics, distributed stream mining systems have been recently developed [1], [3]. Many stream classification and mining applications implement topologies (such as trees or cascades) of low-complexity binary classifiers on such distributed systems to jointly accomplish the task of complex classification [4]. It has been shown that boosting trees of weak classifiers enables the successive identification and filtering of multiple attributes in the data, and it leads to improved accuracy over single classifier systems [5], [10].

Nevertheless, when voluminous data streams need to be processed, resource constraints pose a major challenge for optimizing the performance of cascades of classifiers. Recent work proposes reconfiguring the operating point of each classifier in a linear cascade to ensure that resource constraints are met at each classifier while maintaining high classification quality [9]. However, tree topologies go significantly beyond linearly cascaded classifiers by providing greater flexibility in data processing while also posing different challenges in terms of resource constrained configuration. Specifically, while excess load can be easily handled within the optimization framework for a binary classifier chain, using a single operating point for each classifier in a tree generates two output streams with a total sum output rate that is fixed. Hence, it may not be possible to simultaneously meet tight processing resource constraints for downstream classifiers along both output edges when using only one operating point.

To address this issue, we propose configuring each classifier with multiple operating points, one for each output, e.g., with multiple overlapping and non-overlapping decision thresholds for the different classes. This directly enables intelligent discard of low-confidence data across output edges of each classifier when resources are scarce, as well as intelligent replication of low-confidence data across both positive and negative edges when excess resources are available, which can significantly reduce the number of classification misses.

In Section II, we discuss our binary classifier tree model and formulate the misclassification cost minimization problem under resource constraints. In Section III, we present experimental results for a sports image classification application and discuss several derived insights. We conclude this letter in Section IV and present directions for future research.

II. Minimizing Misclassification Cost in Binary Classifier Trees

A. Binary Classifier Tree Model and Misclassification Cost

Consider a set of N binary classifiers numbered $1,\ldots,N$ cascaded in a tree topology, an example of which, for $N=2$, is shown in Fig. 1. This topology of classifiers may be used to identify data from three (in general K) end-to-end classes of interest. Each binary classifier n partitions input data objects into two classes, a "yes" class $\mathcal{H}_n^0$ and a "no" class $\mathcal{H}_n^1$, and forwards the classified data along respective output edges. For each respective class, denote the probability of correct detection by $(p_D^0)_n$ and $(p_D^1)_n$ and the probability of false alarms by $(p_F^0)_n$ and $(p_F^1)_n$. Note that when the classifier uses one operating point (e.g., thresholding) to label each data item as $\mathcal{H}_n^0$ or $\mathcal{H}_n^1$, we have the following relationships: $(p_D^1)_n = 1-(p_F^0)_n$ and $(p_F^1)_n = 1-(p_D^0)_n$. This coupling is, however, removed when we have multiple operating points (e.g., a different threshold for each output class).

Each end-to-end class k is determined after a set of cascaded local classifications. Class k is characterized by the following: N^k—the number of classifiers that data from that class need to pass through, $\mathbf{v}^k$—the sequence of classifiers in the path (e.g., each component from the set of classifiers $1,\ldots,N$),

Manuscript received March 24, 2008; revised May 04, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Nikolaos V. Boulgouris.

B. Foo and M. van der Schaar are with the Electrical Engineering Department, University of California Los Angeles, Los Angeles, CA 90095-1594 USA (e-mail: bkungfoo@ee.ucla.edu; mihaela@ee.ucla.edu).

D. S. Turaga, O. Verscheure, and L. Amini are with IBM T.J. Watson Research Center, Yorktown Heights, Hawthorne, NY 10598 USA (e-mail: turaga@us.ibm.com; ov1@us.ibm.com; aminil@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2008.2001566

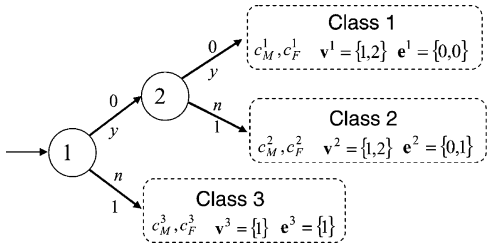

Fig. 1. Example of a depth-2 tree of classifiers with three terminal classes.

$\mathbf{e}^k$—the sequence of branch "types" in the path (class 0 or 1), and c_M^k, c_F^k—the misclassification costs per miss and false alarm in class k. Assuming a (normalized) unit input data rate, the misclassification cost for each class k can be computed from the total rate of data labeled as k (*throughput*) $\bar{t}^k$, the total rate of correctly labeled data in class k (*goodput*) $\bar{g}^k$, and the *a priori* probability of data belonging to class k, $\bar{\pi}^k$, by

$$C^k = c_M^k(\bar{\pi}^k - \bar{g}^k) + c_F^k(\bar{t}^k - \bar{g}^k) \tag{1}$$

where the first term denotes the cost of misses, and the second term denotes the cost of false alarms for class k. The total misclassification cost for the entire tree can be computed as

$$\mathcal{C} = \sum_{k=1}^{K} C^k. \tag{2}$$

To determine $\bar{t}^k$ and $\bar{g}^k$, we model the impact of filtering at classifier v_i^k on the received data stream at classifier v_{i+1}^k by a conditional *a priori* probability $\phi_{v_{i+1}^k}^{e_{i+1}^k, e_i^k}$, where e_{i+1}^k, e_i^k represents the four possible combinations (0,0), (0,1), (1,0), (1,1) corresponding to the "yes" and "no" answers along the two successive branches.[1] The throughputs $(t_{v_{i+1}^k}^0, t_{v_{i+1}^k}^1)$ and goodputs $(g_{v_{i+1}^k}^0, g_{v_{i+1}^k}^1)$ outputted by classifier v_{i+1}^k can be computed from $t_{v_i^k}^{e_i^k}$ and $g_{v_i^k}^{e_i^k}$ using a set of recursive relationships described by the following transfer matrices:

$$\begin{bmatrix} t_{v_{i+1}^k}^{e_{i+1}^k} \\ g_{v_{i+1}^k}^0 \end{bmatrix} = \mathbf{T}_{v_{i+1}^k}^{e_{i+1}^k} \begin{bmatrix} t_{v_i^k}^{e_i^k} \\ g_{v_i^k}^{e_i^k} \end{bmatrix} \tag{3}$$

where $\mathbf{T}_{v_{i+1}^k}^0$ and $\mathbf{T}_{v_{i+1}^k}^1$ are given in the following:

$$\mathbf{T}_{v_{i+1}^k}^0 = \begin{bmatrix} \left(p_F^0\right)_{v_{i+1}^k} & \phi_{v_{i+1}^k}^{0,e_i^k}\left(\left(p_D^0\right)_{v_{i+1}^k} - \left(p_F^0\right)_{v_{i+1}^k}\right) \\ 0 & \phi_{v_{i+1}^k}^{0,e_i^k}\left(p_D^0\right)_{v_{i+1}^k} \end{bmatrix}$$

$$\mathbf{T}_{v_{i+1}^k}^1 = \begin{bmatrix} \left(p_D^1\right)_{v_{i+1}^k} & \left(\phi_{v_{i+1}^k}^{1,e_i^k} - 1\right)\left(p_D^1\right)_{v_{i+1}^k} + \phi_{v_{i+1}^k}^{0,e_i^k}\left(p_F^1\right)_{v_{i+1}^k} \\ 0 & \phi_{v_{i+1}^k}^{1,e_i^k}\left(p_D^1\right)_{v_{i+1}^k} \end{bmatrix}. \tag{4}$$

[1]For a tree topology, only one data stream enters each classifier. Hence, we do not need to include v_i^k while parameterizing ϕ.

Note that the "yes" (0) and "no" (1) output edges have different transfer matrices due to classifier exclusivity. [9][2]

The end-to-end throughput and goodput $\bar{t}^k, \bar{g}^k$ for class k can be computed as

$$\begin{bmatrix} \bar{t}^k \\ \bar{g}^k \end{bmatrix} = \left(\prod_{i=1}^{N_k} \mathbf{T}_{v_i^k}^{e_i^k}\right) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5}$$

and similarly, for the end-to-end *a priori* probability $\bar{\pi}^k$

$$\bar{\pi}^k = \prod_{i=1}^{N^k} \phi_{v_i^k}^{e_i^k, e_{i-1}^k}. \tag{6}$$

B. Resource Consumption and Constraints

Due to the high-complexity operations that need to be performed by each classifier on each data object, limited computational resources in the system impose a heavy constraint on the performance of the classifier tree when the volume of the incoming data stream is large. Since each classifier generally performs the same set of functions on each data object, we model computational resource requirements $r_{v_i^k}$ for each individual classifier as being directly proportional to the rate of data entering it, i.e.,

$$r_{v_i^k} = \alpha_{v_i^k} t_{v_{i-1}^k}^{e_{i-1}^k}$$

where $\alpha_{v_i^k}$ is the amount of resources required per unit rate for classifier v_i^k. Since some of the classifiers for each class overlap, i.e., for multiple classes k, $v_i^k = n$, we concisely denote the entire tree configuration vector by indexing the configuration for each output edge of each of the N classifiers

$$\mathbf{p}_F = \begin{bmatrix} \left(p_F^0\right)_1 & \left(p_F^1\right)_1 & \cdots & \left(p_F^0\right)_N & \left(p_F^0\right)_N \end{bmatrix}$$

and the resource consumption vector for each classifier

$$\mathbf{r}(\mathbf{p}_F) = \begin{bmatrix} r_1 & \cdots & r_N \end{bmatrix}^T.$$

Suppose that each classifier is uniquely placed on one of M different processing nodes. Let $\mathbf{A}_{M\times N}$ be the binary node assignment matrix that maps each classifier onto a processing node, with

$$A_{m,n} = \begin{cases} 1, & \text{if classifier } n \text{ is placed on node } m \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Given that processing node m has R_m available resources, any feasible joint configuration of classifiers needs to satisfy

$$\mathbf{A}\mathbf{r}(\mathbf{p}_F) \le \mathcal{R}, \quad \text{where } \mathcal{R} = \begin{bmatrix} R_1 & \cdots & R_M \end{bmatrix}^T.$$

Hence, the entire cost minimization problem can be given as

$$\min_{\mathbf{p}_F} \mathcal{C} \quad \text{such that } \mathbf{A}\mathbf{r}(\mathbf{p}_F) \le \mathcal{R}, \quad 0 \le \mathbf{p}_F \le 1. \tag{8}$$

Due to the non-convexity of the cost function, discovering the optimal solution may require trying different starting points for convex programming algorithms such as sequential quadratic programming (SQP) [2]. In practice, due to the sharpness of the

[2]Exclusivity in the classifiers implies $\phi_{v_{i+1}^k}^{0,1-e_i^k} = 1 - \phi_{v_{i+1}^k}^{1,1-e_i^k} = 0$.

DET curve, the global minimum can often be found by selecting a starting point near the origin, $\mathbf{p}_F = \mathbf{0}$.

C. Discussion: Single Versus Multiple Operating Points Per Classifier

Recall that using multiple operating points decouples the two output rates by configuring each output class separately, i.e., p_F^0 and p_F^1 are configured independently. This enables flexibility in terms of allowing data to be intelligently discarded or replicated across both branches based on the cost functions. For example, suppose a classifier uses thresholding on the resulting data prediction scores and forwards all data with scores above 0 across the "yes" branch and all data with scores below d across the "no" branch. If $d < 0$, all data between d and 0 are dropped from both branches, leading to load shedding. If $d > 0$, all data between 0 and d are transmitted across both branches, leading to replication.

On the other hand, when a classifier uses one threshold or operating point, the sum of output rates for each classifier is equal to the input rate entering it. Under tight resource constraints, such a strategy may not be feasible and may require the downstream classifiers to discard input data using "arbitrary load shedding." Arbitrary load shedding effectively moves the operating point below the DET curve, towards the origin $(p_F, p_D) = (0, 0)$. Using multiple operating points can always outperform arbitrary load shedding, since for any point below the DET curve on an output edge, moving the point leftwards (until it intersects the DET curve) reduces the false alarm probability while maintaining the same detection probability, which decreases the overall cost while also reducing the throughout for the respective edge.

In Fig. 2, we introduce four algorithms for comparison.

- *Algorithm A* uses the equal error rate (EER) configuration for each classifier, where the probability of false alarm, and the probability of misses across both output edges of each classifier, are equal. This may seem an intuitive approach when the costs are equal for all classes.
- *Algorithm B* minimizes the overall cost without considering resource constraints. Whenever a classifier is overloaded, arbitrary load shedding brings the effective operating point below the DET curve.
- *Algorithm C* use a single operating point for each classifier as in *Algorithm B* but jointly determines the point on the DET curve, and the percentage of output load to shed (randomly) across each branch, such that resource consumption is feasible, and the overall cost is minimized.
- *Algorithm D* selects multiple operating points per classifier to independently filter data across each output edge.

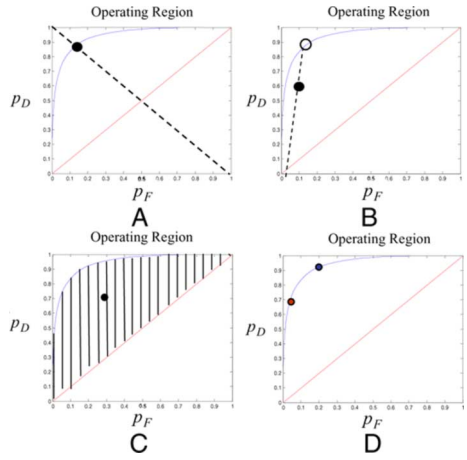

Fig. 2. Pictorial representations of configuration choices in algorithms A–D.

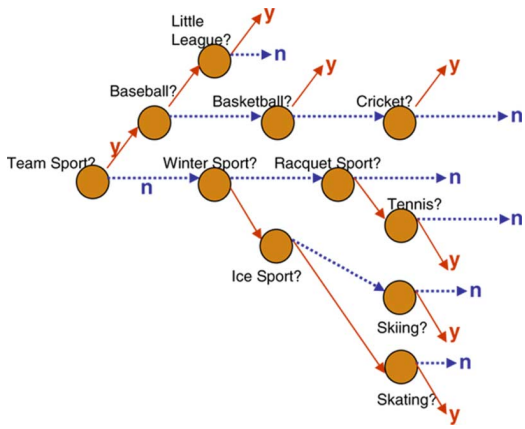

Fig. 3. Structure of a hierarchical sports image classifier system.

III. Experimental Results

A. Application Scenario: Classifying Sports Images

We performed experiments by applying our algorithms to a tree topology of classifiers constructed for a sports image retrieval system [8]. Based on the natural hierarchy in data characteristics, we constructed a classifier tree of the form given in Fig. 3. Each classifier is implemented as a support vector machine (SVM) trained specifically to the characteristic it detects, and each one uses up to 82 features, with complexity on the order of 4000–21 000 support vectors. The DET curves for individual classifiers were measured by testing the classifier on a set of images disjoint from the training set. We observed that the complexity of processing one image is approximately proportional to the number of support vectors. In Table I, we list the approximate amount of processing complexity (normalized) per image for different classifiers. The test image set consists of approximately 20 000 sports image scenes that are streamed at a data rate of one image per second.

TABLE I
Complexity Per Image for Each Classifier. C is Given by Seconds to Process Each Image, Times the Processor Speed

Classifier:	*Complexity:*	*Classifier:*	*Complexity:*
Team Sports	$0.3884 \times C$	**Winter Sports**	$0.3199 \times C$
Baseball	$0.1761 \times C$	Ice Sports	$0.2223 \times C$
Little League	$0.1307 \times C$	Skating	$0.2403 \times C$
Basketball	$0.0772 \times C$	Skiing	$0.2608 \times C$
Cricket	$0.2006 \times C$	**Racquet Sports**	$0.1276 \times C$
		Tennis	$0.1720 \times C$

B. Effect of Resource Constraints on Classifier Configurations

We tested the four algorithms for three different types of system conditions and placements under equal cost for misses and false alarms. The first assumes system resources are abundant, and hence, rate constraints do not need to be considered while configuring classifiers. The second involves placing classifiers on heavily resource constrained processing nodes in a manner that reduces cross-talk between nodes, i.e., traffic across

TABLE II
COSTS OF ALGORITHMS UNDER DIFFERENT RESOURCE CONSTRAINTS AND CLASSIFIER PLACEMENTS

Algorithm	*No Res. Cons.*	*Placement Fig. 4a*	*Placement Fig. 4b*
A	1.9563	1.2971	1.3604
B	0.7742	0.9226	0.9442
C	0.7907	0.9158	0.8964
D	0.6959	0.8640	0.8419

TABLE III
COST SAVINGS UNDER DIFFERENT COST FUNCTIONS

Alg.	*A*	*B*	*C*	*D*
$c_M^k = 1, c_F^k = 4$	3.8906	1.9356	0.9655	0.8703
$c_M^k = 4, c_F^k = 1$	3.8906	1.9355	1.9365	1.5438

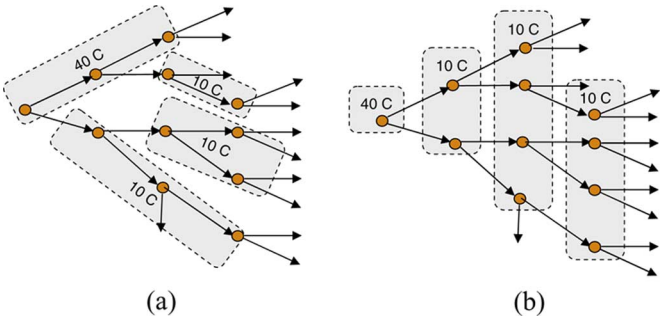

Fig. 4. Placement of classifiers (a) to minimize cross-talk between nodes and (b) to ensure some level of failure resiliency. Note that different nodes have different processing constraints. The constraints are measured in terms of the processor speed (in cycles/second).

the network. The third involves placing classifiers on nodes hierarchically to enable fault tolerance, where more important (upstream) classifiers are placed on more reliable nodes. The placements for types 2 and 3 are shown in Fig. 4.

To evaluate our algorithms, we ran *Algorithms B–D* using SQP from 50 different randomized starting points and provided the minimum costs incurred by the application over all trials in Table II. Note that *Algorithms B and C* show significant reduction in cost over the seemingly intuitive EER configuration for all scenarios. In the non-resource constrained case, the cost of EER was approximately 2.5 times that of *Algorithms B and C*. For resource constrained cases, the cost of EER was 40%–50% greater than *Algorithms B and C*, which demonstrates that the structure of the tree and resource constraints need to be considered jointly in order to optimize a tree of classifiers. Finally, in all cases, enabling multiple operating points (*Algorithm D*) saves 6%–12% in cost over the *Algorithms B and C* due to intelligent filtering and replication.

C. Effect of Unequal Costs on Classifier Configurations

To further highlight the benefit of multiple operating points, we considered two cost functions: 1) $c_M^k = 1$, $c_F^k = 4$ for all classes k and 2) $c_M^k = 4$, $c_F^k = 1$ for all classes. The experiments were performed under loose resource constraints to highlight the effects of cost.

Table III depicts the gains derived for each type of cost metric. For high costs of false alarms, we discovered significant savings when load shedding at the output was considered (*Algorithm C*). The reason for this large gain is that, unlike *Algorithm B*, which always keeps the entire output load from each classifier, *Algorithm C* can completely shed the output load whenever the quality of decision (e.g., goodput to throughput ratio) falls below a certain threshold. In our simulations, the load was completely shed by *Algorithm C* at the edges going into classifiers "Winter Sports" and "Cricket." Nevertheless, using multiple operating points performed the best because rather than shedding the entire load down certain branches, it could shed only data objects of lower confidence.

For high costs of misses, a huge cost saving resulted from using multiple operating points (approximately 21%) due to the intelligent replication of data, which reduces the probability of miss for each class. For example, we discovered that approximately 18% of the data from "Team Sports,", 10% from "Baseball," and 9% from "Winter Sports" was replicated.

IV. CONCLUSIONS AND FUTURE WORK

In this letter, we introduced the paradigm of jointly configuring binary classifier trees to minimize misclassification costs under resource constraints. By using multiple thresholds for each classifier, significant cost savings can be achieved through the intelligent filtering and replication of data. Future work can consider more general processing topologies, such as directed acyclic graphs, and classifiers instantiated on multiple nodes, where many thresholds can be used to multicast "*yes*" and "*no*" output data to downstream classifiers based on their respective resource constraints. Joint classifier placement and configuration under resource constraints can also be explored.

ACKNOWLEDGMENT

The authors would like to thank the MARVEL [8] team, including J. R. Smith, R. Yan, and J. Yang, for providing access to the SVM-based classifiers as well as the training and testing image data sets.

REFERENCES

[1] L. Amini, H. Andrade, F. Eskesen, R. King, Y. Park, P. Selo, and C. Venkatramani, The Stream Processing Core, IBM T.J. Watson Research Center, Tech. Rep. RSC 23798, Nov. 2005.
[2] P. Boggs and J. Tolle, *Sequential Quadratic Programming*. Cambridge, U.K.: Acta Numerica, 1995.
[3] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. B. Zdonik, "Scalable distributed stream processing," in *Proc. 2nd Biennial Conf. Innovative Data Systems Research (CIDR)*, Jan. 2003.
[4] R. Lienhart, L. Liang, and A. Kuranov, "A detector tree for boosted classifiers for real-time object detection and tracking," in *Proc. Int. Conf. Multimedia and Expo (ICME)*, 2003.
[5] H. J. Z. R. Xiao and L. Zhu, "Boosting chain learning for object detection," in *Proc. 9th IEEE Int. Conf. Computer Vision*, 2003.
[6] T. E. Senator, "Multi-stage classification," in *Proc. 5th Int. Conf. Data Mining (ICDM)*, 2005, pp. 386–393.
[7] M. A. Shah, J. M. Hellerstein, S. Chandrasekaran, and M. J. Franklin, "Flux: An adaptive partitioning operator for continuous query systems," in *Proc. 19th Int. Conf. Data Engineering (ICDE)*, Mar. 2003, pp. 25–36.
[8] J. R. Smith, IBM Multimedia Analysis and Retrieval System (MARVEL). [Online]. Available: http://mp7.watson.ibm.com/marvel/.
[9] D. S. Turaga, O. Verscheure, U. V. Chaudhari, and L. Amini, "Resource management for chained binary classifiers," in *Proc. Workshop Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, 2006.
[10] L. M. X. Carreras, "Boosting trees for anti-spam email filtering," in *Proc. Recent Advances in Natural Language Processing*, 2001.