TREE CONFIGURATION GAMES FOR DISTRIBUTED STREAM MINING SYSTEMS

Hyunggon Park, Deepak S. Turaga+, Olivier Verscheure+ and Mihaela van der Schaar**

+IBM T. J. Watson Research Center, Hawthorne, NY, USA
*UCLA Electrical Engineering Department, Los Angeles, CA, USA

ABSTRACT

We consider the problem of configuring classifier trees in distributed stream mining system. The configuration involves selecting appropriate false-alarm detection tradeoffs for each classifier to minimize end-to-end penalty in terms of misclassification cost. We model this as a tree configuration game and design solutions, where individual classifiers select their operating points to maximize a local utility. We derive appropriate misclassification cost coefficients for intermediate classifiers, and determine the information that needs to be exchanged across classifiers, in order to successfully design the game. We analytically show that there is a unique pure strategy Nash equilibrium in operating points, which guarantees a convergence of the proposed approach. We evaluate the performance of our algorithm on an application for sports scene classification, and compare against centralized solutions. We show that our algorithm results in better performance than the centralized solution on average. Moreover, the algorithm approaches the optimal solution asymptotically with increasing number of actions per classifier.

Index Terms— Resource constrained stream mining, tree configuration games, binary classifier tree.

1. INTRODUCTION

There are an increasing number of applications that require processing and classification of continuous, high volume data streams. These include online photo and video streaming services, financial analysis, real-time manufacturing process control, search engines, spam filters, security, and medical services [1–3], etc. These applications are often composed as processing topologies of distributed operators [4–6] deployed on large-scale stream mining systems [6, 7]. Specifically, many stream mining applications implement topologies (ensembles such as trees or cascades) of low-complexity binary classifiers to hierarchically filter the data streams and jointly accomplish the task of complex classification [2, 8].

Stream mining applications pose several interesting research challenges. These include the optimal construction and training of such topologies, as well as configuration and management of individual classifiers to maximize end-to-end performance – especially under dynamically varying and distributed resource constraints and data characteristics. In this paper, we focus on the classifier configuration problem for binary tree topologies, i.e., determining the optimal operating point (detection - false alarm tradeoff) for each classifier in the tree, in order to maximize the end-to-end classification performance. Previous approaches model classifier tree configuration as an optimization problem and use centralized techniques such as Sequential Quadratic Programming (SQP) [9] to solve it. Such approaches require centralized control of all the classifiers, information and data. Hence, the designed solutions suffer disadvantages in terms of having a single central point of control and associated failure, issues with scaling and adaptation as the topology grows, and not allowing large scale applications with capabilities distributed across multiple proprietary entities.

In this paper, we model the problem as a *distributed tree configuration game*, where each classifier decides its optimal action in order to maximize its local utility, based on the available information. We investigate several issues: 1) the information that needs to be exchanged across classifiers to successfully design the game, 2) the design of appropriate schemes for determining cost coefficients for classification errors and local utility functions, 3) the existence of a unique Nash equilibrium for the game, and 4) the performance quantification of the Nash equilibrium. We present results on an application for hierarchical semantic concept detection application in sports images and evaluate performance against prior centralized approaches. We examine the impact of varying costs of misclassification as well as granularity of the decision space - in terms of the available quantized Detection Error Tradeoff (DET) curve[1] per classifier.

This paper is organized as follows. In Section 2, we introduce the model for individual classifiers and classifier trees. In Section 3, we introduce the tree configuration game, including the utility function definition, available actions etc. We present the application of interest and simulation results in Section 4, and conclude in Section 5 with directions for future research.

2. SYSTEM MODEL - BINARY CLASSIFIER TREES

We consider a stream mining system, which consists of several binary classifiers in a tree topology. An illustrative stream mining application [9] is depicted in Fig. 1. The topology of classifiers in this example is used to identify semantic con-

[1]It can be referred to as Receiver Operating Characteristic (ROC) curve.

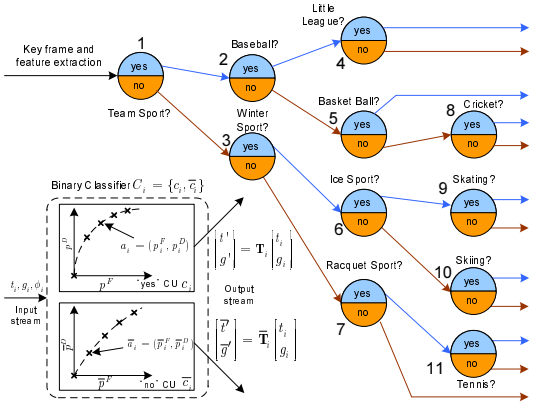

Fig. 1. An illustrative classifier topology.

cepts from sports image data using hierarchical filtering. Leaf classifiers (e.g. classifier 4, 8 etc.) represent the actual class of interest, while intermediate classifiers assist in hierarchical filtering of data based on a semantic hierarchy of concepts.

· *Configuration of Binary Classifier:* A binary classifier filters input data into the "yes" class and the "no" class. We model each classifier C_i with two classification units (CUs), i.e., $C_i = \{c_i, \bar{c}_i\}$, corresponding to the "yes" and "no" outputs respectively (Fig. 1). We use notation $i \rightsquigarrow k$ to denote that c_i (or $\bar{c}_i$) is a *preceding* CU for c_k (or $\bar{c}_k$). The topology allows disambiguation between the two CUs per classifier.

· *Stream Characteristics:* The input stream for classifier C_i is characterized by *throughput* t_i and *goodput* g_i, which represent total data rate and correctly labeled data rate, respectively. The average fraction of the input stream data that represents true positive for CU c_i is denoted by ϕ_i. ϕ_i is predetermined based on the classifier topology and data characteristics. For $\bar{c}_i$, $\bar{\phi}_i = 1 - \phi_i$.

· *Performance of CU:* As in [9], performance of c_i ($\bar{c}_i$) is controlled by its tradeoff between probability of false alarm p_i^F ($\bar{p}_i^F$) and probability of detection p_i^D ($\bar{p}_i^D$). The two CUs may have decoupled operating points, e.g., through the use of independent thresholds (one for "yes" and one for "no") for score based classifiers. The set of operating points (p_i^F, p_i^D) represent the DET curve – a non-decreasing concave function.

· *Misclassification Cost:* Cost coefficients λ_i^F ($\bar{\lambda}_i^F$) and λ_i^M ($\bar{\lambda}_i^M$) represent the cost/penalty per unit data rate of *false alarm* and *miss* for CU c_i ($\bar{c}_i$). These coefficients are specified by the application for leaf classifiers – and may be derived for other classifiers based on the topology.

· *Input and Output Rates:* For c_i and $\bar{c}_i$, the output stream rates (t_i', g_i') and $(\bar{t}_i', \bar{g}_i')$ may be derived as [9]

$$\begin{bmatrix} t_i' \\ g_i' \end{bmatrix} = \mathbf{T}_i \begin{bmatrix} t_i \\ g_i \end{bmatrix}, \text{and } \begin{bmatrix} \bar{t}_i' \\ \bar{g}_i' \end{bmatrix} = \bar{\mathbf{T}}_i \begin{bmatrix} t_i \\ g_i \end{bmatrix}, \quad (1)$$

where $\mathbf{T}_i$ and $\bar{\mathbf{T}}_i$ are given by

$$\mathbf{T}_i = \begin{bmatrix} p_i^F & \phi_i(p_i^D - p_i^F) \\ 0 & \phi_i p_i^D \end{bmatrix}, \text{and } \bar{\mathbf{T}}_i = \begin{bmatrix} \bar{p}_i^D & \phi_i(\bar{p}_i^F - \bar{p}_i^D) \\ 0 & \bar{\phi}_i \bar{p}_i^D, \end{bmatrix}.$$

· *End-to-End System Utility:* Let $\mathbf{C}_L$ be a set of leaf CUs and denote the outgoing throughput and goodput from a leaf CU $c_l \in \mathbf{C}_L$ by t_l' and g_l', respectively. Then, the incurred end-to-end cost of misclassification can be expressed as

$$(t_l' - g_l')\lambda_l^F + (\Lambda_l - g_l')\,\lambda_l^M,$$

where $\Lambda_l = t_r \hat{\phi}_l \cdot \prod_{\forall k \in \{j | j \rightsquigarrow l\}} \hat{\phi}_k$, with $\hat{\phi}_k = \phi_k$ for c_k and $\hat{\phi}_k = \bar{\phi}_k$ for $\bar{c}_k$. Λ_l represents a true fraction of stream data that belong to c_l for input stream rate t_r to the tree. We define the utility achieved by c_l as the negative cost, or

$$U_l = -\left[(t_l' - g_l')\lambda_l^F + (\Lambda_l - g_l')\,\lambda_l^M\right]. \quad (2)$$

Similarly, $\bar{U}_l$ achieved by $\bar{c}_l$ can be expressed in terms of coefficients $\bar{\lambda}_l^F$ and $\bar{\lambda}_l^M$. The system utility $U_{\mathbf{S}}$ may be expressed as

$$U_{\mathbf{S}} = \sum\nolimits_{c_l \in \mathbf{C}_L} U_l + \sum\nolimits_{\bar{c}_l \in \mathbf{C}_L} \bar{U}_l. \quad (3)$$

3. TREE CONFIGURATION GAMES

For a stream mining system with N classifiers, a tree configuration game consists of a finite set of CUs $\{(c_i, \bar{c}_i) | 1 \leq i \leq N\}$ (i.e., players), a nonempty set of actions $\mathbf{A}_i$, a utility function $U_i(\cdot)$, and a strategy $\pi_i(\cdot)$ for each CU c_i.

3.1. Action Set $\mathbf{A}_i$

An action $a_{ik} \in \mathbf{A}_i$ $(1 \leq k \leq |\mathbf{A}_i|)$ for CU c_i represents the selection of operating point $\left(p_{ik}^F, p_{ik}^D\right)$, i.e. the kth operating point among $A_i \triangleq |\mathbf{A}_i|$ available operating points. Note that A_i is determined based on a quantization of the DET curve – represented by a differentiable concave function $f_i : [0,1] \rightarrow [0,1]$, defined as $p_i^D = f_i(p_i^F)$, for $0 \leq p_i^F \leq 1$. Hence, action a_{ik}, for uniform quantization, corresponds to selecting operating point $\left(\frac{k-1}{A_i - 1}, f_i\left(\frac{k-1}{A_i - 1}\right)\right)$ (see Fig. 1). We can similarly define the action set $\bar{\mathbf{A}}_i$ for CU $\bar{c}_i$.

3.2. Local Utility Function

As in (2), the utility function for an intermediate CU c_i ($c_i \notin \mathbf{C}_L$) can be expressed as

$$U_i(a_i) = -\left[(t_i' - g_i')\lambda_i^F + (\Lambda_i - g_i')\,\lambda_i^M\right], \quad (4)$$

where λ_i^F and λ_i^M denote intermediate cost coefficients, that are not explicitly defined by the application. We need to derive them by back-propagating from leaf classifiers, accounting for the topology. λ_i^F and λ_i^M for intermediate CU c_i may be derived from its immediately successive classifier C_k as:

$$\begin{aligned} \lambda_i^F &= \phi_k \lambda_k^F + \bar{\phi}_k \bar{\lambda}_k^F = \phi_k \lambda_k^F + (1 - \phi_k)\bar{\lambda}_k^F, \\ \lambda_i^M &= \phi_k \lambda_k^M + \bar{\phi}_k \bar{\lambda}_k^M = \phi_k \lambda_k^M + (1 - \phi_k)\bar{\lambda}_k^M, \end{aligned} \quad (5)$$

We assume that this back-propagation is performed before the tree configuration game.

3.3. Strategy π_i for CU c_i

The strategy for CU c_i is to select the action that maximizes its local utility, i.e.,

$$a_i^* = \pi_i(\mathbf{I}_i) = \arg\max_{a_i \in \mathbf{A}_i} U_i(a_i), \tag{6}$$

where $\mathbf{I}_i = \{t_i, g_i, \phi_i, \lambda_i^F, \lambda_i^M, \mathbf{A}_i, \Lambda_i\}$ denotes the available information. Since a function $f_i(\cdot)$ that characterizes the DET curve of c_i is non-decreasing and concave, the utility function is also a concave function. Hence, an optimal action a_i^* can be uniquely determined. Moreover, determining the optimal action requires a constant computational complexity even for different sizes of action sets. The information $\mathbf{I}_i$ consists of a set of information $\{t_i, \phi_i, \lambda_i^F, \lambda_i^M, \mathbf{A}_i\}$ that is always available, as well as $\{g_i, \Lambda_i\}$ that is not directly available. We show that a set of actions $\mathbf{a}_{-i}$, defined as $\mathbf{a}_{-i} = \{\hat{a}_k | k \rightsquigarrow i, \forall k\}$, ($\hat{a}_k = a_k$ for c_k and $\hat{a}_k = \bar{a}_k$ for $\bar{c}_k$), taken by preceding CUs of c_i and $\Phi_i = \left\{\hat{\phi}_k | k \rightsquigarrow i, \forall k\right\}$ need to be exchanged across classifiers to correctly determine the optimal action at c_i.

***Proposition* 1.** *For a CU c_i, $\mathbf{a}_{-i}$ and Φ_i are the minimum information that needs to be exchanged to correctly determine an optimal action.*

Proof. Without loss of generality, we consider a "yes" CU c_i. To correctly specify the utility function given in (4), c_i requires information $\{g_i, \Lambda_i\}$ in addition to available information $\{t_i, \phi_i, \lambda_i^F, \lambda_i^M, \mathbf{A}_i\}$.

Since Λ_i is defined as $\Lambda_i = t_r \phi_i \cdot \prod_{\forall k \in \{j | j \rightsquigarrow i\}} \hat{\phi}_k$, Φ_i is sufficient to specify Λ_i.

Based on the input-output relationship of c_i as in (1), t_i and g_i can be expressed as

$$[t_i \ g_i]^T = \prod\nolimits_{k \in \{j | j \rightsquigarrow i\}} \hat{\mathbf{T}}_k \cdot [t_r \ t_r]^T, \tag{7}$$

where $\hat{\mathbf{T}}_k = \mathbf{T}_k$ for c_k and $\hat{\mathbf{T}}_k = \bar{\mathbf{T}}_k$ for $\bar{c}_k$. Since $\hat{\mathbf{T}}_k$ can be explicitly specified based the information $\mathbf{a}_{-i}$ and Φ_i, $\prod_{k \in \{j | j \rightsquigarrow i\}} \hat{\mathbf{T}}_k$ becomes a 2×2 matrix $\mathbf{T}$ with its deterministic constant elements T_{mn}, $1 \leq m, n \leq 2$. Hence, c_i can compute g_i, since t_i and g_i are expressed as $t_i = (T_{11} + T_{12}) t_r$ and $g_i = (T_{21} + T_{22}) t_r$, respectively. □

Hence, we require that information $\mathbf{a}_{-i}$ and Φ_i are always forwarded with outgoing stream data. Note that Proposition 1 implies that both the size of messages and the number of messages that need to be exchanged increase linearly with the tree depth.

3.4. Convergence Analysis

We now show that if each of the CUs determines its operating point based on the strategy in (6), the determined operating points are in a unique pure strategy Nash equilibrium.

***Proposition* 2.** *If each of CUs determines its action based on the strategy in* (6)*, then there exists a unique pure strategy Nash equilibrium in operating points.*

Proof. Let c_i be a "yes" CU, which determines its optimal action based on the strategy π_i in (6). Since the utility of c_i is determined based on its own action $a_i \in \mathbf{A}_i$ as well as the set of actions $\mathbf{a}_{-i}$ of its preceding CUs, the utility function in (4) can be explicitly expressed as $U_i(a_i, \mathbf{a}_{-i})$. Since $\mathbf{a}_{-i}$ is already determined and fixed, we have

$$U_i(a_i^*, \mathbf{a}_{-i}) > U_i(a_i, \mathbf{a}_{-i}), \tag{8}$$

for all actions $a_i (\neq a_i^*) \in \mathbf{A}_i$. Similarly, the same argument holds for a "no" CU $\bar{c}_i$, i.e.,

$$\bar{U}_i(\bar{a}_i^*, \bar{\mathbf{a}}_{-i}) > \bar{U}_i(\bar{a}_i, \bar{\mathbf{a}}_{-i}), \tag{9}$$

for all actions $\bar{a}_i (\neq \bar{a}_i^*) \in \bar{\mathbf{A}}_i$. Since each CU can uniquely determine its optimal action, there exists a unique pure strategy Nash equilibrium in operating points of the CUs. □

Proposition 2 implies that the decisions on operating points of all the CUs always converge if their actions are determined based on the strategy. Moreover, the convergence time increases linearly with the tree depth.

4. SIMULATION RESULTS

4.1. Simulation Set-up

To evaluate the performance of the proposed approaches, we consider the semantic concept detection application in Fig. 1. Each classifier operates on low level image features such as color histograms, color correlograms, etc. using a Support Vector Machine, and classifiers are organized into a semantic hierarchy of concepts [9]. We compare performance of the proposed approach against the centralized approach of [9].

4.2. Impact of the Action Granularity and Backward Cost Coefficient Propagation Scheme

To highlight the impact of the number of available actions and the proposed backward cost coefficient propagation scheme on the end-to-end system cost, we consider an elementary sub-tree of our application, consisting of classifiers 1, 2, and 3 in Fig. 1. We set $\lambda_l^F = 4, \lambda_l^M = 1$ and $\bar{\lambda}_l^F = 1, \bar{\lambda}_l^M = 4$ for $l = 2, 3$, and consider different levels of quantization granularity, i.e., increased number of available actions $A_i = 10, 20, 40, 80, 160, 320$ for $i = 1, 2, 3$. We set the input stream rate $t_r = 1$. The resulting misclassification cost is shown in Fig. 2.

Fig. 2 clearly shows that increasing the number of available actions leads to a lower system cost – approaching the performance of the best result for the centralized algorithm. There is a slight performance gap, as CUs in the distributed approach determine their actions individually with no consideration of the impact on the end-to-end system utility. Moreover, unlike in the centralized approach, only finite number

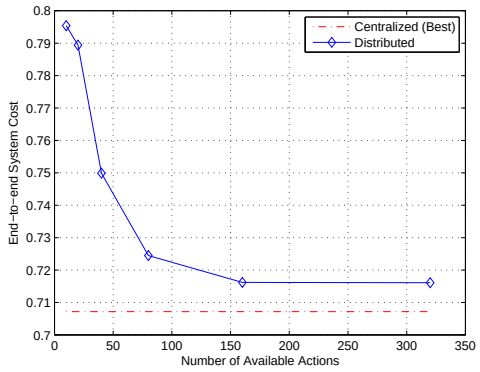

Fig. 2. End-to-end system costs.

Table 1. Achieved End-to-End System Costs (80 actions)

Exp. Cases	$\lambda^F = \lambda^M$	$\lambda^F = 4\lambda^M$	$4\lambda^F = \lambda^M$
Cent. (avg.)	0.587 (0 %)	0.963 (0 %)	1.373 (0 %)
Distributed	0.536 (68.0 %)	0.806 (86.3 %)	1.187 (88.6 %)
Cent. (best)	0.512 (100 %)	0.781 (100 %)	1.163 (100 %)

of (quantized) actions are available to each CU. Finally, instead of the backward propagation approach, if misclassification cost coefficients of C_1 are determined randomly or based on a heuristic average approach (e.g., $\lambda_1^F = (\lambda_2^F + \bar{\lambda}_2^F)/2$), the resulting misclassification costs are 1.0108 and 0.8167 (for $A_i = 80$), respectively. These are 139.5% and 112.7% of the cost (0.7245) achieved by the proposed approach. The performance degradation is caused because these approaches do not consider the stream characteristics.

4.3. Quantification of End-to-End System Performance

In this section, we quantify the performance achieved by the proposed tree configuration games in Fig 1. In this simulation, we assume that each CU has 80 available actions. We compare against the centralized solution using SQP in [9]. Since SQP is gradient descent based, we use 500 different randomized starting points, and provide the minimum (best) as well as the average cost (avg.). The results are shown in Table 1.

It is clear that the proposed distributed approach always outperforms the average performance of centralized approach for different misclassification cost scenarios i.e., $(\lambda^F = \lambda^M = 1)$, $(\lambda^F = 4, \lambda^M = 1)$, and $(\lambda^F = 1, \lambda^M = 4)$. These correspond to equal cost for false alarms and misses, high costs for false alarms, and high costs for misses respectively. Additionally, as the costs become unbalanced, the distributed algorithm performance increasingly approaches the optimal performance of the centralized algorithms. This is reflected in the percentages in the table, computed as $(Cost^{dist} - Cost^{cent}_{avg})/(Cost^{cent}_{best} - Cost^{cent}_{avg}) \times 100\%$.

5. CONCLUSIONS

In this paper, we model the configuration of classifier tree topologies in distributed stream mining system as a tree configuration game. We determine the minimum information that needs to be exchanged across classifiers and propose a novel scheme for determining the local utilities for intermediate classification units, which are required to successfully design the tree configuration game. We analytically show that the proposed approach guarantees a convergence to a unique pure strategy Nash equilibrium in operating points of each CU. Simulation results, performed on a semantic concept detection application for sports image analysis, show that the performance of the proposed approach is comparable to a centralized solution – outperforming the average performance, and closely approaching the optimal performance (68%∼88.6% depending on misclassification cost coefficients). We also show that the distributed algorithm performance improves with the number of actions available to each classifier, and with unbalanced costs for false alarms and misses. A key direction for future research involves extending the current approach to the case where CUs have additional information about their successive CUs. Based on this information, a CU can form coalitions with other CUs and can consider actions that maximize the coalition utility.

6. REFERENCES

[1] M. A. Shah, J. M. Hellerstein, S. Chandrasekaran, and M. J. Franklin, "Flux: An adaptive partitioning operator for continuous query systems," in *ICDE*, March 2003, pp. 25–36.

[2] R. Lienhart, L. Liang, and A. Kuranov, "A detector tree for boosted classifiers for real-time object detection and tracking," in *ICME*, 2003.

[3] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in *WWW*, May 2006, pp. 83–92.

[4] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *SIGMOD*, June 2003, pp. 563–574.

[5] 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, November 2005.

[6] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. B. Zdonik, "Scalable distributed stream processing," in *CIDR*, January 2003.

[7] M. Balazinska, H. Balakrishnan, S. Madden, and M. Stonebraker, "Fault tolerance in the borealis distributed stream processing system," in *SIGMOD*, June 2005, pp. 13–24.

[8] Y. Mao, X. Zhou, D. Pi, Y. Sun, and S. T. C. Wong, "Multiclass cancer classification by using fuzzy support vector machine and binary decision tree with gene selection." *Journal of Biomedical Biotechnology*, vol. 2005, no. 2, pp. 160–71, 2005.

[9] D. S. Turaga, B. Foo, O. Verscheure, and M. van der Schaar, "Configuring topologies of distributed semantic concept classifiers for continuous multimedia stream processing," in *ACM Multimedia 2008*, 2008.