CONFIGURING NETWORKED CLASSIFIERS IN DISTRIBUTED AND RESOURCE CONSTRAINED STREAM PROCESSING SYSTEMS

F. Fu[‡*], *D. S. Turaga*[†], *O. Verscheure*[†], *M. van der Schaar*[‡], *L. Amini*[†]

[†]IBM. T.J. Watson Research Center, 19 Skyline Dr, Hawthorne, NY
[‡]University of California, Los Angeles, 54-147A Engineering IV Building, Los Angeles, CA

ABSTRACT

We design algorithms for optimally configuring, in terms of model complexity as well as operating point, a cascade of *exclusive* classifiers on distributed systems, given underlying resource constraints. Under rate-dependent constraints we design a Viterbi-like strategy to determine the optimal solution with significantly lower computational complexity than an exhaustive search. We illustrate the performance of this strategy on a cascade of classifiers for a speaker verification task, and highlight performance gains over using a single classifier.

Index Terms – Resource constrained classification, Multi-stage classification, Distributed Stream Processing

1. INTRODUCTION

There is a large set of emerging applications that perform classification, filtering, aggregation and correlation over high-volume, unbounded, continuous data (email, instant messages, transactional data, audio, video, sensor data etc.). Distributed stream processing systems (Aurora, Borealis, Telegraph CQ, System S [1]) provide the framework to deploy and run such applications on various resource topologies. Distributed stream processing requires the decomposition of these classification tasks into a set of networked operations. In this context, both classifiers in series with the same model (boosting) and classifiers in parallel with multiple models (bagging) have resulted in improved classification performance. Recent work [2] shows that the optimal design of two-stage classification must explicitly consider interaction between stages. However, this work has not considered resource management issues [3, 4].

Prior research on resource constrained stream processing focuses primarily on load-shedding [5, 6]. These approaches are limited by their assumption that the impact of load shedding on performance is known a-priori. Furthermore, load shedding is often performed using locally available information and metrics, and this may lead to sub-optimal end-to-end performance. Finally, shedding load at intermediate classifiers leads to resource wastage as data has been processed by upstream classifiers.

In [7] we introduced a framework that allows individual classifiers in the ensemble to operate at different performance levels given the resources allocated to them. Expressed simply, instead of deciding *what fraction of the data to process*, as in load-shedding, we determine *how to process available data* given underlying resource constraints. In this paper, we extend that framework to consider classifiers distributed across several machines, and develop algorithms for optimally configuring such a cascade of classifiers, under resource constraints. The paper is organized as follows. We formulate the problem in Section 2, and provide solutions in Sections 3 and 4. We present experimental results on a real cascade for speaker verification in Section 5, and conclude in Section 6.

2. DEFINITIONS AND PROBLEM FORMULATION

2.1. Utility Definition for Classifier Cascade

Consider a cascade of N binary classifiers with input data X (Figure 1). Classifier C^k ($1{\le}k{\le}N$) classifies data X^{k-1} into either class H_0^k or class H_1^k, and passes through only data X^k that it classifies as belonging to H_0^k (*filtering* classifier). Let p_D^k and p_F^k respectively denote the probabilities of correct *detection* and *false alarm*.

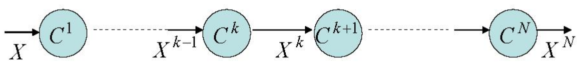

Fig. 1. Classifier Cascade

The total proportion (throughput) of data forwarded by classifier C^k is labeled t^k and the total proportion of data correctly forwarded is labeled g^k. We may compute these under *exclusivity*, i.e. $P(X \in H_0^k | X \in H_1^{k-1}) = 0$ ([7]) as:

$$\begin{bmatrix} t^k \\ g^k \end{bmatrix} = \underbrace{\begin{bmatrix} p_F^k & \phi^k(p_D^k - p_F^k) \\ 0 & \phi^k p_D^k \end{bmatrix}}_{\mathbf{T}^k} \begin{bmatrix} t^{k-1} \\ g^{k-1} \end{bmatrix} \quad (1)$$

where $\phi^k = P(X \in H_0^k | X \in H_0^{k-1})$ is the data conditional probability (on unfiltered data) across these two classifiers[1] in the cascade. The above derived relationship accounts for the

*This work was done while at the IBM T.J. Watson Research Center. Contact author: fwfu@ee.ucla.edu

[1]In this paper, we assume that classifiers are independently trained on the

fact that the data density (a priori probability of belonging to any class) is modified due to filtering by each classifier in the cascade. Hence, the end to end throughput and goodput may be computed as:

$$\begin{bmatrix} t^N \\ g^N \end{bmatrix} = \mathbf{T}^N \cdots \mathbf{T}^1 \begin{bmatrix} t^0 \\ g^0 \end{bmatrix} \tag{2}$$

where $t^0 = g^0 = 1$. This may be geometrically interpreted (Figure 2 a.) in the performance space (*throughput vs goodput*) where the impact of $\mathbf{T}^k$ corresponds to a translation in the operating performance. The cascade of classifiers thus moves the input point t^0, g^0 towards the optimal *ideal* performance point $t^N = g^N = P(X \in H_0^N)$ (i.e. the point where all the data is correctly identified). The individual operating points of the classifiers determine the matrices $\mathbf{T}^k$ or equivalently the path in this space.

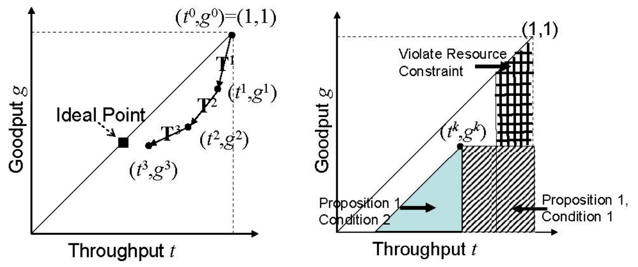

Fig. 2. a) Operating Performance Characteristic. b) Set of points for discard

The end to end throughput and goodput map directly to the performance of a virtual combined classifier with $\mathcal{P}_D = g^N$ and $\mathcal{P}_F = t^N - g^N$. Hence, as for a single classifier [7], we may define the performance of this cascade in terms of utility $U = \mathcal{P}_D - \Theta\mathcal{P}_F$ or $U = (1+\Theta)g^N - \Theta t^N$, where Θ is a control parameter that defines the desired tradeoff between false alarms and misses.

2.2. Resource Consumption for Classifiers

The performance of each classifier may be modified by changing its underlying model complexity, as well as selecting its operating point, i.e. individual p_D-p_F tradeoff on the Detection Error Tradeoff (DET) curve. This may also affect the resource consumption of future classifiers in the cascade as it changes the amount of data filtered through. Consider a simple example of a Gaussian Mixture Model (GMM) classifier using the likelihood ratio test. Changing the model complexity corresponds to changing the number of Gaussians in the mixture, while changing the operating point corresponds to using a different threshold during the likelihood ratio test. We represent the model complexity of classifier C^k as Q^k, and the operating point in terms of the corresponding p_F^k. We consider two different types of resource consumption models, one for rate-independent resources (e.g. Memory) and the other for rate-dependent resources (e.g. CPU). We model rate-independent resource consumption as $f^k(Q^k) = \eta^k Q^k$, with η^k a constant that depends on data dimensionality, model parameter set (means, covariances), points (for lookup) stored per Gaussian, and the underlying system architecture. The rate-dependent resource consumption depends on operating points and complexities of $C^1..C^k$ as well as the input data rate w and may be defined as:

$$h^k(\mathbf{Q}^{1\to k}, \mathbf{p_F}^{1\to(k-1)}) = w\gamma^k Q^k t^{k-1} \tag{3}$$

where γ^k is a constant similar to η^k (in general $\gamma^k \neq \eta^k$). Furthermore, $\mathbf{p_F}^{1\to(k-1)} = \begin{bmatrix} p_F^1 & \cdots & p_F^{k-1} \end{bmatrix}^T$ and $\mathbf{Q}^{1\to k} = \begin{bmatrix} Q^1 & \cdots & Q^k \end{bmatrix}^T$, where T denotes the transpose. The resource consumption of the N classifiers may be aggregated into vectors as:

$$\begin{aligned} \mathbf{f}(\mathbf{Q}) &= \begin{bmatrix} \eta^1 Q^1 & \eta^2 Q^2 & \cdots & \eta^N Q^N \end{bmatrix}^T \\ \mathbf{h}(\mathbf{Q}, \mathbf{p_F}) &= \begin{bmatrix} w\gamma^1 Q^1 t^0 & \cdots & w\gamma^N Q^N t^{N-1} \end{bmatrix}^T. \end{aligned} \tag{4}$$

2.3. Distributed Classifier Cascade

In a distributed streaming system these N classifiers may be distributed across M servers. We define a location variable l^k with $l^k = m$ implying C^k is placed on server m[2]. Hence, we construct a location matrix, $\mathbf{A} = \{a_{i,j}\}_{M\times N}$ with $a_{i,j} = 1$ if $l^j = i$ and zero otherwise, to represent the location of the N classifiers across the M servers. If the available rate independent resources on server m are represented as $\mathcal{L}_m$ and the rate-dependent resources as $\mathcal{R}_m$, we may capture the resources available over M servers as $\mathbf{L}_{tot} = \begin{bmatrix} \mathcal{L}_1 & \cdots & \mathcal{L}_M \end{bmatrix}^T$, and $\mathbf{R}_{tot} = \begin{bmatrix} \mathcal{R}_1 & \cdots & \mathcal{R}_M \end{bmatrix}^T$.

2.4. Configuring Classifiers under Resource Constraints: Formulation

The problem we are trying to solve is to optimally configure (determine the model complexity Q^k and the operating point p_F^k for each classifier) the cascade of classifiers to maximize the utility given the distributed resource constraints. This combined optimization may be formulated as:

$$\begin{aligned} &max_{\mathbf{Q},\mathbf{p_F}} U(\mathbf{Q}, \mathbf{p_F}) \\ \text{s.t. } &\mathbf{Af}(\mathbf{Q}) \le \mathbf{L}_{tot} \text{ and } \mathbf{Ah}(\mathbf{Q}, \mathbf{p_F}) \le \mathbf{R}_{tot} \end{aligned} \tag{5}$$

The optimal configuration is selected from among the feasible set of complexities $\mathcal{Q}^k$ ($Q^k \in \mathcal{Q}^k$) and operating points $\mathcal{P}(Q^k)$ ($p_F^k \in \mathcal{P}(Q^k)$). Additionally, we use $\mathbf{x} \le \mathbf{y}$ to represent $x_i \le$

original data set, without assumptions on cascaded data filtering. The training of classifiers given the cascade structure is an important direction for future research.

[2]We assume a *forward flow* cascade, i.e. if $k_0 < k_1$, we have $l_{k_0} \le l_{k_1}$.

y_i with $1 \le i \le D$ for D dimensional vectors. A pure rate-independent (rate-dependent) constraint optimization may be solved for by setting $\mathbf{R}_{tot} = \mathbf{Inf}$ ($\mathbf{L}_{tot} = \mathbf{Inf}$).

In terms of the geometric interpretation mentioned earlier, these optimizations involve determining the optimal path in the operating performance space that satisfies the underlying resource constraints.

3. SOLUTION: RATE-INDEPENDENT CONSTRAINTS

Consider, without loss of generality, a case with two classifiers on one machine with a given desired Θ. Consider two complexity configurations of these classifiers: a) $Q^1 = 0, Q^2 = \mathcal{L}_{tot}$ and b) $\bar{Q}^1 = \mathcal{L}, \bar{Q}^2 = \mathcal{L}_{tot} - \mathcal{L}$, i.e. operate classifier 1 as an all pass filter in scenario (a). Under these complexities, let the classifiers be tuned to the optimal operating points $(1, 1), (p_F^2, p_D^2)$ and $(\bar{p}_F^1, \bar{p}_D^1), (\bar{p}_F^2, \bar{p}_D^2)$ respectively. Note that because of the higher resource allocation to classifier 2 in case (a), we can easily show that $p_D^2 - \Theta p_F^2 \ge \bar{p}_D^2 - \Theta \bar{p}_F^2$. By writing out the respective matrices $\mathbf{T}^k$, we may compute the corresponding utilities U and $\bar{U}$ respectively. With some tedious algebra, and using these properties, we can then show that $U \ge \bar{U}$ for all values of Θ. Intuitively, this makes sense as the first classifier does not help alleviate the resource requirements for the second classifier, and only ends up erroneously discarding some correct data early. This idea can be extended to N exclusive classifiers, and we may conclude that under rate-independent constraints, it is always better to allocate all available resources to the last classifier in the chain.

4. SOLUTION: RATE-DEPENDENT CONSTRAINTS

Unlike for the rate-independent constraints, a cascade of classifiers is often necessary to meet rate-dependent constraints. This is because early simple (low-complexity) classifiers may be used to reduce the rate of data flow, thereby alleviating the resource requirements of future classifiers. The corresponding savings in resources, often outweigh any misclassifications by the early classifiers in the cascade. In this section, we use a Viterbi-like search strategy to configure the cascaded exclusive classifiers by selecting the operating points and model complexity, under rate-dependent resource constraints.

4.1. Viterbi-like Search Strategy

Note that the Viterbi decoding algorithm is useful when the feasible configuration set is finite and limited. At classifier C^k, there are $|\mathcal{Q}^k|$ feasible complexity points and $|\mathcal{P}(Q^k)|$ feasible operating points to select for each complexity Q^k, where $|\bullet|$ is the cardinality of the set. Over the entire chain, there are $\prod_{k=1}^{N} \sum_{Q^k \in \mathcal{Q}^k} |\mathcal{P}(Q^k)|$ combinations, which can grow prohibitively large. However, we may use the following proposition to limit the number of points that need to be searched in a Viterbi-like search strategy.

Proposition 1 *For classifier C^k the point $(\hat{t}^k, \hat{g}^k)$ cannot lead to better end-to-end utility than (t^k, g^k) if either of the two following conditions are satisfied:*
1) $\hat{t}^k \ge t^k$ and $\hat{g}^k \le g^k$
2) $0 \le t^k - \hat{t}^k \le g^k - \hat{g}^k$

Proof. Consider that the best end-to-end utility that can be achieved given point (t^k, g^k) is U (correspondingly $\hat{U}$ for the other point). This end-to-end utility may be written as:

$$U = \begin{bmatrix} -\Theta \\ 1+\Theta \end{bmatrix}^T \mathbf{T}^N \cdots \mathbf{T}^{k+1} \begin{bmatrix} t^k \\ g^k \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} t^k \\ g^k \end{bmatrix} \tag{6}$$

It is straightforward to show that $x \le 0$ and $y \ge 0$. If condition 1 holds, it is evident that $U \ge \hat{U}$ independent of the actual values of x and y. If condition 2 holds, then we can create an intermediate point $(t^k_{rem}, \hat{g}^k)$, with resulting best utility U_{rem} by removing purely good (i.e. correctly classified) data fraction $g^k - \hat{g}^k$ from (t^k, g^k). Note that, since we discard only correctly classified data, it is guaranteed that $U \ge U_{rem}$. Furthermore, we observe that

$$t^k_{rem} = t^k - (g^k - \hat{g}^k) \le t^k - (t^k - \hat{t}^k) = \hat{t}^k \tag{7}$$

directly from condition 2. Now we may use condition 1 to show that $U_{rem} \ge \hat{U}$. This means that $U \ge U_{rem} \ge \hat{U}$, thereby proving the proposition. □

We may use the above proposition to discard search candidates given the resource constraints. For each (t^k, g^k) we compute the rate-dependent resources consumed $\mathcal{R}^{con}_{l^k}$ on server l^k due to classifiers C^1 through C^k. We can then discard all $(\hat{t}^k, \hat{g}^k)$ that satisfy Proposition 1 and the condition $\mathcal{R}^{con}_{l^k} \le \hat{\mathcal{R}}^{con}_{l^k}$, i.e. all points that cannot lead to better utility while consuming more resources up to this machine[3]. Additionally, we also discard points $(\hat{t}^k, \hat{g}^k)$, if the point (t^k, g^k) is feasible in the *worst future case*, i.e. resource consumption up to C^k is such that $C^{k+1}..C^N$ can be operated at maximum complexity without violating the end-to-end resource constraint, i.e. $\mathbf{R}^{proj}_{max} \le \mathbf{R}_{tot}$, where $\mathbf{R}^{proj}_{max}$ is the projected resource consumption assuming the maximum complexity for $C^{k+1}..C^N$. Finally, we also discard points $(\hat{t}^k, \hat{g}^k)$ that violate the CPU constraint up to the current classifier. An illustration of the set of points that can be discarded is shown graphically in Figure 2 b. The designed strategy is shown in Algorithm 1.

5. EXPERIMENTAL RESULTS

We consider a speaker verification task, where we want to verify whether speech samples belong to a particular female speaker. We build a cascade of two classifiers for this task, with C^1 being a gender detector (i.e. forwards only female speech) and C^2 being the actual speaker verifier (tries to verify whether speech belongs to specific speaker). These classifiers are trained on real Switchboard telephony data with

[3]We do not consider servers $1..l^k - 1$ due to the forward flow assumption

Algorithm 1 Viterbi-Like Search Strategy

```
**Set** $(t^0, g^0) = (1, 1)$ and $k = 0$
**repeat**
    Forward. For each $(t^k, g^k)$ generate possible $(t^{k+1}, g^{k+1})$ and compute $\mathcal{R}^{con}_{l^{k+1}}$
    Pruning 1. Discard $(t^{k+1}, g^{k+1})$ if $\mathcal{R}^{con}_{l^{k+1}} > \mathcal{R}_{l^{k+1}}$
    Pruning 2. Discard $(\hat{t}^{k+1}, \hat{g}^{k+1})$ if $\exists$ $(t^{k+1}, g^{k+1})$ s.t. Proposition 1 is satisfied and $\mathcal{R}^{con}_{l^{k+1}} \leq \hat{\mathcal{R}}^{con}_{l^{k+1}}$ or $\mathbf{R}^{proj}_{max} \leq \mathbf{R}_{tot}$.
    $k \Leftarrow k + 1$
**until** $k = N$
From remaining points pick point that maximizes utility.
**return** $(\mathbf{Q}, \mathbf{p_F})$
```

176 female speakers, and 106 male speakers using the IBM speaker verification system [8] at complexities (number of Gaussians in the GMM) of $Q^1 \in \{8, 16, 32, 64\}$ and $Q^2 \in \{8, 16, 32, 64, 128\}$. Our experimental setup is described in more detail in [7]. We assume that the classifiers are located on one machine with $\gamma^1 = \gamma^2 = 1$, and we use $\Theta = 0.1$ corresponding to the cost-tradeoff between misses and false alarms in NIST testing scenarios. We vary the rate-dependent resource constraint $\mathcal{R}_1 \in \{40, 80, 160\}$ and show examples of the selected optimal operating point and classifier complexity (using the Viterbi-like strategy) in Figure 5. The first thing we

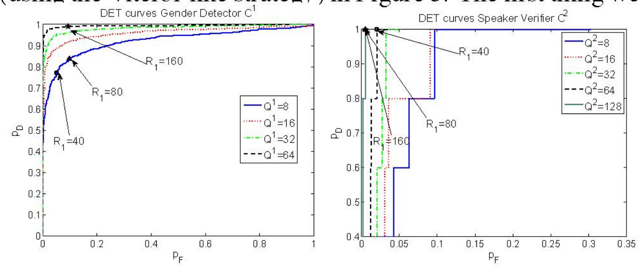

Fig. 3. Experimental Results: Speaker Verification

observe is that, as the resource constraint is relaxed, the classifiers expectedly use higher complexity, and improve their utility. It is more important to notice that by providing a small amount of resources to (simple) C^1, C^2 can operate at very high model complexity, while satisfying resource constraints. We highlight this, by comparing the utility derived from the cascade against using only the speaker verifier in Table 1.

Table 1. Utility ($U \times 100$) Under Resource Constraints.

Classifier Topology	$\mathcal{R}_1 = 40$	$\mathcal{R}_1 = 80$	$\mathcal{R}_1 = 160$
Cascade: $C^1 \rightarrow C^2$	0.17	0.27	0.32
No Cascade: C^2	0.03	0.15	0.30

Clearly, under resource constraints, a classifier cascade (even with independently trained classifiers), outperforms a single classifier. As the available resources increase, the gains from using a cascade decrease, as expected. We also compare the complexity of our Viterbi-like strategy against a full search, and observe savings (in terms of the number of points in the search space) of a factor of 10-100 for this two classifier cascade. We also explored a more general topology with 6 simulated classifiers (over 3 machines), and observe savings of up to a factor of 10^6 over the exhaustive strategy, while achieving the optimal utility. Note that the actual savings depend on the classifier characteristics, the topology, and the resource constraints (with higher savings under tight resource constraints).

6. CONCLUSION

In this paper we design an algorithms for optimally configuring cascaded exclusive classifiers, on distributed systems, under resource constraints. We show that a classifier cascade degenerates into a single classifier under rate-independent constraints. We then propose a Viterbi-like search strategy to determine the optimal operating points and classifier complexities under rate-dependent constraints. We examine the performance of this strategy on a cascade of two classifiers (gender detector followed by a speaker verifier) in a real speaker verification system. We show that under resource constraints, this cascade outperforms a single classifier. The computational complexity of the Viterbi-like strategy is 10-100 times lower than an exhaustive search, while guaranteeing solution optimality. Under simulated scenarios with 6 classifiers, we have observed complexity savings of up to a factor of 10^6. There are several directions for future research. We would like to extend these ideas for more complex topologies and also consider the construction of such topologies, including the training of classifiers specific to a cascaded structure. We would also like to examine the cascade performance in dynamically varying scenarios.

7. REFERENCES

[1] L. Amini *et al*, "The stream processing core," Tech. Rep. RSC 23798, IBM T. J. Watson Research Center, Nov. 2005.

[2] T. C. W. Landgrebe *et al*, "Optimising two-stage recognition systems," in *International Workshop on Multiple Classifier Systems*, June 2005.

[3] T. E. Senator, "Multi-stage classification," in *IEEE ICDM*, Nov 2005.

[4] P. Frossard, O. Verscheure, and C. Venkatramani, "Signal processing challenges in distributed stream processing systems," in *IEEE ICASSP*, May 2005.

[5] Y. Chi *et al*, "Loadstar: A load shedding scheme for classifying data streams," in *SIAM SDM*, Oct 2004.

[6] A. Jain, E. Y. Chang, and Y-F Wang, "Adaptive stream resource management using kalman filters," in *ACM SIGMOD*, 2004.

[7] D. S. Turaga *et al*, "Resource management for networked classifiers in distributed stream mining systems," in *IEEE ICDM*, Dec 2006.

[8] G. N. Ramaswamy *et al*, "The ibm system for the nist 2002 cellular speaker verification evaluation," in *IEEE ICASSP*, April 2003.