Incremental Boolean Combination of Classifiers*

Wael Khreich[1], Eric Granger[1], Ali Miri[2], and Robert Sabourin[1]

[1] Laboratoire d'imagerie, de vision et d'intelligence artificielle
École de technologie supérieure, Montreal, QC, Canada
wael.khreich@livia.etsmtl.ca,
{eric.granger,robert.sabourin}@etsmtl.ca
[2] School of Computer Science, Ryerson University, Toronto, Canada
ali.miri@ryerson.ca

Abstract. The incremental Boolean combination (*incrBC*) technique is a new learn-and-combine approach that is proposed to adapt ensemble-based pattern classification systems over time, in response to new data acquired during operations. When a new block of training data becomes available, this technique generates a diversified pool of base classifiers from the data by varying training hyperparameters and random initializations. The responses of these classifiers are then combined with those of previously-trained classifiers through Boolean combination in the ROC space. Through this process, an ensemble is selected from the pool, where Boolean fusion functions and thresholds are adapted for improved accuracy, while redundant base classifiers are pruned. Results of computer simulations conducted using Hidden Markov Models (HMMs) on synthetic and real-world host-based intrusion detection data indicate that *incrBC* can sustain a significantly higher level of accuracy than when the parameters of a single best HMM are re-estimated for each new block of data, using reference batch and incremental learning techniques. It also outperforms static fusion techniques such as majority voting for combining the responses of new and previously-generated pools of HMMs. Pruning prevents pool sizes from increasing indefinitely over time, without adversely affecting the overall ensemble performance.

1 Introduction

In practice, pattern recognition systems are typically designed a priori using limited and imbalanced data acquired from complex changing environments. Various one- and two-class neural and statistical classifiers have been applied to detection tasks, for instance, to learn and detect normal or abnormal system behavior. Since the collection and analysis of representative training data for design an validation is costly, the classifier may represent an incomplete view of system behavior. Since new training data may become available after a classifier has originally been deployed for operations, it could be adapted to maintain or improve performance over time.

* This research has been supported by the Natural Sciences and Engineering Research Council of Canada.

C. Sansone, J. Kittler, and F. Roli (Eds.): MCS 2011, LNCS 6713, pp. 340–349, 2011.

Given a new block of training data, incremental re-estimation of classifier parameters raises several challenges. Parameters should be updated from new data without requiring access to the previously-learned data, and without corrupting previously-acquired knowledge [6]. State-of-the-art batch learning classifiers must accumulate new training data in memory, and retrain from the start using all (new and previously-accumulated) data. A number of classifiers in literature have been designed with the inherent ability to perform supervised incremental learning. However, the decline in performance caused by knowledge corruption remains an issue. Indeed, single classifier systems for incremental learning may not adequately approximate the underlying data distribution when there are multiple local maxima in the solution space [1].

Ensemble methods have been employed to overcome such limitations [6]. Theoretical and empirical evidence suggests that combining the responses of several accurate and diverse classifiers can enhance the overall accuracy and reliability of a pattern classification system [4,10]. Despite reducing information to binary decisions, combining responses at the decision level, in the Receiver Operating Characteristic (ROC) space, allows to combine across a variety of classifiers trained with different hyperparameters, feature subsets and initializations.

In this paper, a new ensemble-based technique called incremental Boolean combination (*incrBC*) is proposed for incremental learning of new training data according to a learn-and-combine approach. When a new block of training data becomes available, it is used to generate a new pool of classifiers by varying training hyperparameters and random initializations. The responses from the newly-trained classifiers are then combined to those of the previously-trained classifiers by applying Boolean combination in the ROC space. The proposed system allows to improve overall accuracy by evolving an ensemble of classifiers (EoCs) in which Boolean fusion functions and decision thresholds are adapted. Since the pool size grows indefinitely over time, *incrBC* integrates model management strategies to limit the pool size without significantly degrading performance.

For proof-of-concept, *incrBC* is applied to adaptive anomaly detection from system call sequences with Hidden Markov Models (HMMs). The experiments are conducted on both synthetically generated and sendmail data from the University of New Mexico [11]. Learning new data allows to account for rare events, and hence improve detection accuracy and reduce false alarms. The performance of the proposed system is compared to that of the reference algorithms for batch and incremental learning of HMM parameters. In addition, the performance achieved with Boolean fusion functions is compared to that of median (MED) and majority vote (VOTE) functions combining the outputs from pool of HMMs.

2 Learn-and-Combine Approach Using Incremental Boolean Combination

Boolean combination (BC) has recently been investigated to combine the decision of multiple crisp or soft one- or two-class classifiers in the ROC space [8]. These threshold-optimized decision-level combination techniques can outperform several techniques in the Neyman-Pearson sense, yet they assume that

the classifiers are conditionally-independent, and that their corresponding ROC curves are convex and proper. These assumptions are rarely valid in practice, where classifiers are designed using limited and imbalanced data. In previous research, the authors proposed BC techniques [3] for efficient fusion of multiple ROC curves using all Boolean functions, without any prior assumptions. These technique apply to batch learning of a fixed-size data set.

In this paper, an extension to the batch BC techniques (proposed in [3]) – called incremental BC (*incrBC*) technique – is proposed for incremental learning of new training data during operations. In response to a new block of data, this learn-and-combine approach consists in generating a new pool of classifiers, and then applying *incrBC* to combine ROC curves of the new pool with the ROCCH obtained with previously-obtained data.

As described in Algorithm 1, *incrBC* uses each Boolean function to combine the responses corresponding to each decision threshold from the first classifier to those from the second classifier. Fused responses are then mapped to vertices

Algorithm 1. $incrBC(\lambda_1, \lambda_2, \ldots, \lambda_K, \mathcal{V})$: Incremental Boolean combination of classifiers

input : K classifiers $(\lambda_1, \lambda_2, \ldots, \lambda_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$
output: ROCCH of combined classifiers where each vertex is the result of 2 to K combination of crisp classifiers. Each combination selects the best decision thresholds (λ_i, t_j) and Boolean function, which are stored in the set $(\mathcal{S})$

$n_k \leftarrow$ no. decision thresholds of λ_k using $\mathcal{V}$ // no. vertices on ROC(λ_k)
$BooleanFunctions \leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$
compute $ROCCH_1$ of the first two classifiers (λ_1 and λ_2)
allocate $\boldsymbol{F}$ an array of size: $[2, n_1 \times n_2]$ // temporary storage of combination results
foreach $bf \in BooleanFunctions$ **do**
    for $i \leftarrow 1$ **to** n_1 **do**
        $\boldsymbol{R_1} \leftarrow (\lambda_1, t_i)$ // responses of λ_1 at decision threshold t_i using $\mathcal{V}$
        for $j \leftarrow 1$ **to** n_2 **do**
            $R_2 \leftarrow (\lambda_2, t_j)$ // responses of λ_2 at decision threshold t_j using $\mathcal{V}$
            $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_1}, \boldsymbol{R_2})$ // combine responses using current Boolean function
            compute (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combined responses to ROC space
            push (tpr, fpr) onto $\boldsymbol{F}$
    compute $ROCCH_2$ *of all ROC points in* $\boldsymbol{F}$
    $n_{ev} \leftarrow$ number of emerging vertices
    $\mathcal{S}_2 \leftarrow \{(\lambda_1, t_i), (\lambda_2, t_j), bf\}$ // set of selected decision thresholds from each classifier and Boolean functions for emerging vertices
for $k \leftarrow 3$ **to** K **do**
    allocate $\boldsymbol{F}$ of size: $[2, n_k \times n_{ev}]$
    foreach $bf \in BooleanFunctions$ **do**
        for $i \leftarrow 1$ **to** n_{ev} **do**
            $\boldsymbol{R_i} \leftarrow S_{k-1}(i)$ // responses from previous combinations
            for $j \leftarrow 1$ **to** n_k **do**
                $\boldsymbol{R_k} \leftarrow (\lambda_k, t_j)$
                $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_i}, \boldsymbol{R_k})$
                compute (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$
                push (tpr, fpr) onto $\boldsymbol{F}$
    compute $ROCCH_k$ *of all ROC points in* $\boldsymbol{F}$
    $n_{ev} \leftarrow$ number of emerging vertices
    $\mathcal{S}_k \leftarrow \{\mathcal{S}_{k-1}(i), (\lambda_k, t_j), bf\}$ // set of selected subset from previous combinations, decision thresholds from the newly-selected classifier, and Boolean functions for emerging vertices
store $\mathcal{S}_k : 2 \leq k \leq K$
return $ROCCH_K$

in the ROC space, and their ROC convex hull (ROCCH) is computed. Vertices that are superior to the ROCCH of original classifiers are then selected. The set ($\mathcal{S}$) of decision thresholds from each classifier and Boolean functions corresponding to these vertices is stored, and the ROCCH is updated to include emerging vertices. The responses corresponding to each decision threshold from the third classifier are then combined with the responses of each emerging vertex, and so on, until the last classifier in the pool is combined. The BC technique yields a final ROCCH for visualization and selection of operating points, and the set of selected thresholds and Boolean functions, $\mathcal{S}$, for each vertex on the composite ROCCH to be applied during operations. For a pool of K soft classifiers each comprising n crisp classifiers, the worst-case time complexity required for combinations (during the design phase) is $\mathcal{O}(Kn^2)$ Boolean operations.

Selecting crisp detectors from all HMMs in the pool leads to unnecessarily high computational and memory complexity since the pool size grows as new blocks of data become available. In addition, HMMs are combined according to the order in which they are stored in the pool. An HMM trained on new data may capture different underlying data structure and could replace several previously-selected HMMs. Model management mechanisms are therefore required for ensemble selection and pruning less relevant members of the pool.

Model Selection. Ensemble selection is performed at each new block of data, and the best ensemble of classifiers is selected from the pool based on different optimization criteria and selection strategies, each one seeking to increase accuracy and reduce the computational and memory complexity [9]. Optimization criteria include ensemble accuracy, entropy and diversity. Ensemble selection strategies include ranking- and search-based techniques. In ranking-based techniques, the pool members are ordered according to some performance measure on a validation set, and the top classifiers are then selected to form an ensemble. Search-based techniques first combine the outputs of classifiers and then select the best performing ensemble evaluated on a validation set.

The proposed ensemble selection algorithm, BC_{search}, (see Algorithm 2) employs both rank- and search-based selection strategies to optimize the area under the ROCCH (AUCH)[1]. In contrast with existing techniques, BC_{search} exploits the monotonicity of the *incrBC* algorithm for an early stopping criterion. Furthermore, the final composite ROCCH is always stored which allows for visualization of the whole range of performance and adaptation to changes in prior probabilities and costs of errors. This can be achieved by adjusting the desired operational point, which activates different classifiers, decision thresholds and Boolean functions.

As described in Algorithm 2, the BC_{search} algorithm employs an incremental search strategy. It starts by ranking all classifiers in decreasing order of their AUCH accuracy on a validation set and then selects the classifier with the largest AUCH value. Next, it applies the *incrBC* algorithm to combine the selected

[1] Other ROC-based measures, such as the partial AUCH or the true positive rate at a required false positive rate, can also be measured for a region-specific accuracy.

Algorithm 2. $BC_{search}(\mathcal{P}, \mathcal{V})$: Boolean combination – incremental search

input : Pool of classifiers $\mathcal{P} = \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ and a validation set $\mathcal{V}$
output: Ensemble of base classifiers (E) selected from the pool $\mathcal{P}$

set tol // set the tolerance value for AUCH
foreach $\lambda_k \in \mathcal{P}$ **do**
  compute ROC curves and their $ROCCH_k$, using $\mathcal{V}$
sort classifiers $(\lambda_1, \ldots, \lambda_K)$ in descending order of $AUCH(ROCCH_k)$ values
$\lambda_1 = \arg\max_k \{AUCH(ROCCH_k) : \lambda_k \in \mathcal{P}\}$
$E \leftarrow \lambda_1$ // select classifier with the largest AUCH value
foreach $\lambda_k \in \mathcal{P} \backslash E$ **do** // remaining classifiers in $\mathcal{P}$
  $ROCCH_k$=*incrBC* $((\lambda_1, \lambda_k), \mathcal{V})$
$\lambda_2 = \arg\max_k \{AUCH(ROCCH_k) : \lambda_k \in \mathcal{P} \backslash E\}$
$E \leftarrow E \cup \lambda_2$ // select classifier with largest AUCH improvement to E
$ROCCH_1 \leftarrow ROCCH_2$ // update the convex hull
$\mathcal{S}_2 \leftarrow \{(\lambda_1, t_i), (\lambda_2, t_{i'}), bf\}$ // Set of selected decision thresholds from each classifier and selected Boolean functions (from *incrBC* algorithm)
$j \leftarrow 3$
repeat
  foreach $\lambda_k \in \mathcal{P} \backslash E$ **do**
    $ROCCH_k$=*incrBC* $((\mathcal{S}_{j-1}, \lambda_k), \mathcal{V})$
  $\lambda_j = \arg\max_k \{AUCH(ROCCH_k) : \lambda_k \in \mathcal{P} \backslash E\}$
  $E \leftarrow E \cup \lambda_j$
  $ROCCH_{j-1} \leftarrow ROCCH_j$
  $\mathcal{S}_j \leftarrow \{\mathcal{S}_{j-1}(i), (\lambda_j, t_{i'}), bf\}$ // Set of selected subset from previous combinations, decision thresholds from the newly-selected classifiers, and Boolean functions (derived from *incrBC* algorithm)
  $j \leftarrow j + 1$
until $AUCH(ROCCH_j) \leq AUCH(ROCCH_{j-1}) + tol$ // no improvement
store $\mathcal{S}_j$, $2 \leq j \leq K$
return E

classifier with each of the remaining classifiers in the pool. The classifier that most improves the AUCH accuracy of the EoCs is then selected. The cumulative EoCs are then combined with each of the remaining classifiers in the pool (using *incrBC*), and the classifier that provides the largest AUCH improvement to the EoCs is selected, and so on. The algorithm stops when the AUCH improvement of the remaining classifiers is lower than a user-defined tolerance value, or when all classifiers in the original ensemble are selected. Given a pool of K classifiers, the worst-case time complexity of this selection algorithm is $O(K^2)$ w.r.t. the number of Boolean combination of *incrBC*. However, this complexity is only attained when the algorithm selects all classifiers (zero tolerance). In practice, the computational time is typically lower depending on the tolerance value.

Model Pruning. As new blocks of data are learned incrementally, pruning less relevant models is essential to limit the pool size $|\mathcal{P}|$ (and memory resources) from growing indefinitely. With *incrBC*, classifiers that go unselected over time are discarded. A counter is therefore assigned to each classifier in the pool indicating the number of blocks for which an classifier was not selected as an ensemble member. A classifier is then pruned from the pool, according to a user-defined life time (LT) expectancy value of unselected models. For instance, with an $LT = 3$ all classifiers that have not been selected after receiving three blocks of data, as indicated by their counters, are discarded from the pool.

3 Simulation Results

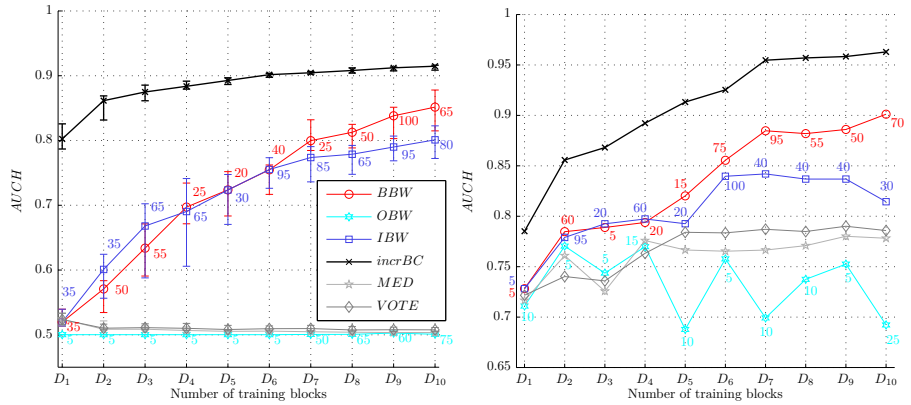
Host-based intrusion detection systems applied to anomaly detection (AD) typically monitor for significant deviations in system call sequences, since system calls are the gateway between user and operating system's kernel mode. Among various neural and statistical classifiers, techniques based on HMMs have been shown to produce a high level of performance [11], although standard techniques for re-estimating HMM parameters involve batch learning [2]. Designing an HMM for AD involves estimating HMM parameters and the number of hidden states (N) from the training data. *incrBC* allows to adapt AD to newly-acquired data based on a learn and combine approach.

Proof-of-concept simulations are conducted on both synthetically generated data and sendmail process data collected at the University of New Mexico (UNM)[2] [11]. The synthetic data generator is based on the conditional relative entropy (CRE) [3,7], which controls the irregularity of the generated data ($CRE = 0$, perfect regularity and $CRE = 1$, complete irregularity). The synthetically generated data simulate a complex process, with an alphabet $\Sigma = 50$ symbols and $CRE = 0.4$. The sizes of injected anomalies (AS) are assumed equal to the detector window (DW) sizes. The training is conducted on ten successive blocks D_k, for $k = 1, \ldots, 10$, of normal system call sequences, each of length $DW = 4$ symbols. For the synthetic data, each block comprises 500 sub-sequences, whereas the validation ($\mathcal{V}$) and test ($\mathcal{T}$) sets are comprised of $2,000$ and $5,000$ labeled sub-subsequences (normal or anomalous). For sendmail data, each block comprises 100 sub-sequences, and each $\mathcal{V}$ and $\mathcal{T}$ comprise 450 sub-sequences. In both cases, the anomaly size is $AS = 4$ symbols and the ratio of normal to anomalous sequences is $4 : 1$.

For each D_k, 20 new base HMMs are generated and appended to the pool ($\mathcal{P}$). These ergodic HMMs are trained with 20 different number of states ($N = 5, 10, \ldots, 100$) according to the Baum-Welch (BW) algorithm. For each N value, 10-fold cross-validation and ten different random initializations are employed to select the HMM (λ_N^k) that gives the highest AUCH on $\mathcal{V}$. The HMMs from the pool (of increasing size $|\mathcal{P}| = 20, 40, \ldots, 200$ HMMs) are then provided for incremental combination according to the *incrBC* technique. The same training, validation and selection procedures are applied to the other techniques. However, for the reference batch Baum-Welch (BBW) the training is conducted on cumulative data blocks ($D_1 \cup D_2 \ldots \cup D_k$), and both online Baum-Welch (OBW) [5] and IBW [2] algorithms resume the training from the previously-learned HMMs using only the current block of data (D_k).

Incremental BC without model management. The performance of the proposed *incrBC* technique is first presented without employing the model management strategy. For a given block D_k, all available HMMs in $\mathcal{P}$ were selected and combined according to *incrBC*. As illustrated in Figure 1, the average AUCH accuracy achieved by applying this technique is highest overall. It is significantly

[2] `http://www.cs.unm.edu/~immsec/systemcalls.htm`



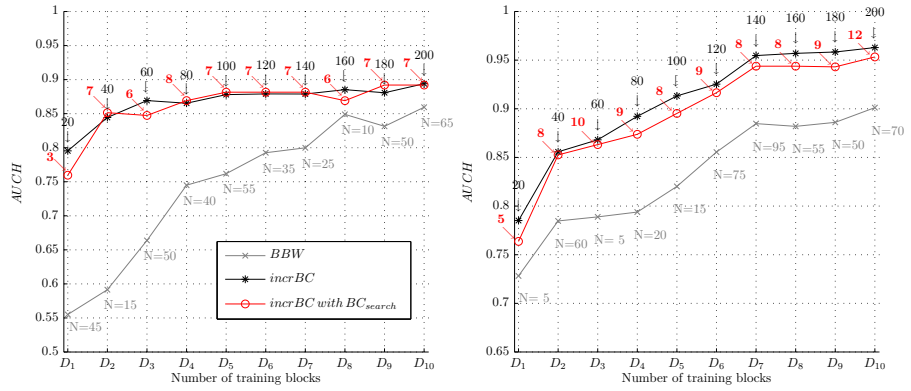
(a) Synthetic data. (b) Sendmail data.

Fig. 1. Average AUCH of techniques used for incremental learning over 10 successive blocks of data D_k. For each block, the HMMs are trained according to each technique with 20 different states ($N = 5, 10, \ldots, 100$), providing a pool of size $|\mathcal{P}| = 20, 40, \ldots, 200$ HMMs. Numbers above points are the N values for BBW, OBW and IBW that achieved the highest AUCH accuracy on each block. Error bars in (a) are lower and upper quartiles over ten replications.

higher than that of the reference BBW, most notably when provided with limited training data in the first few blocks. The *incrBC* effectively exploits the complementary information provided from the pool of HMMs trained with different number of states and different initializations, and using newly-acquired data. Not surprisingly, OBW leads to the lowest level of accuracy, as one pass over limited data is insufficient to capture the underlying data structure. IBW outperforms OBW since it re-estimates HMM parameters iteratively over each block using a fixed learning rate [2].

The MED and VOTE fusion functions do not improve accuracy with respect to the Boolean functions produced with *incrBC*, which reflects their inabilities to exploit the complementary information. The MED function directly combines HMM likelihood values for each sub-sequence during operations, while VOTE considers the crisp decisions from HMMs at optimal operating thresholds (equal error rates). In contrast, *incrBC* applies ten Boolean functions to the crisp decisions provided by the thresholds of each HMM. Then, it selects the decision thresholds and Boolean functions that improve the overall ROCCH on the validation set $\mathcal{V}$. Exploring all decision thresholds before selection may increase ensemble diversity, and improve overall system accuracy.

Model selection. Figure 2 presents the impact on accuracy of using the BC_{search} algorithm proposed for selection of ensembles in the *incrBC* technique (compared to that of *incrBC* from Figure 1). The results in Figure 2a (using synthetic data) correspond to the first replication of Figure 1a. As shown in Figure 2, BC_{search} maintains a AUCH accuracy that is comparable to that of

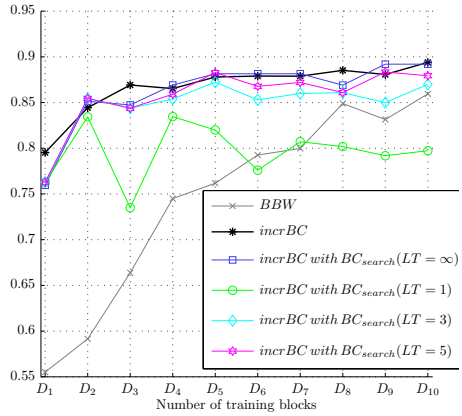


(a) Synthetic data ($tolerance = 0.01$) (b) Sendmail data ($tolerance = 0.003$)

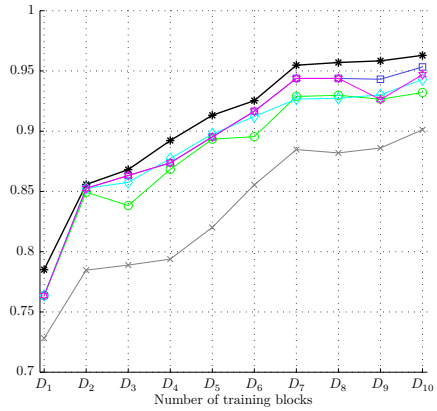
Fig. 2. Average AUCH accuracy achieved when the *incrBC* technique employs BC_{search} for model selection, over 10 successive blocks of data D_k. For each block, values on the arrows correspond to the number of models (EoHMM size) selected by each technique from the overall pool of size $|\mathcal{P}| = 20, 40, \ldots, 200$ HMMs. The performance of the BBW used to train a single best HMM, where the best state value is selected at each block, is also shown for reference.

incrBC. For each block however, the size of selected EoHMMs, $|E|$, is reduced significantly compared to the original pool of size $|\mathcal{P}| = 20, 40, \ldots, 200$ HMMs. For instance, at the 10^{th} data block in Figure 2a, BC_{search} selects an ensemble of size $|E| = 7$ from the generated pool of size $|\mathcal{P}| = 200$ HMMs (selected with *incrBC*). BC_{search} provides compact and accurate ensembles by exploiting the order in which HMMs are combined and the benefit achieved by cumulative EoHMMs before selecting a new ensemble member.

Model pruning. Figure 3 presents the AUCH accuracy of the full *incrBC* technique, employing BC_{search} and the pruning strategy according to various life time (LT) expectancy values. An HMM is pruned if it is not selected during a LT corresponding to 1, 3 or 5 data blocks. Figure 3a illustrates the impact on AUCH accuracy of pruning the pool of HMMs in Figure 2a (synthetic data), while Figure 3b illustrates the impact of pruning the pool of HMMs in Figure 2b (sendmail data). As shown in Figure 3a, the level of AUCH accuracy achieved with *incrBC* decreases when LT varies from ∞ to 1. Early punning ($LT = 1$) of HMMs that have not improved ensemble performance during a given learning stage, may lead to knowledge corruption, and hence a decline in system performance. These HMMs may provide complementary information to newly-generated HMMs, depending on the data. This is illustrated in Figure 3b, where the decline in AUCH for $LT = 1$ is relatively smaller for sendmail data, which incorporates more redundancy than the synthetically-generated data. As shown in Figures 3a and 3b, the performance achieved with a delayed pruning of HMMs (e.g., $LT = 3$ and 5) compares to that of retaining all generated HMMs in the pool ($LT = \infty$). For fixed tolerance and LT values, *incrBC* is capable of selecting small EoHMMs and further reducing the



(a) Pruning the pool shown in Figure 2a



(b) Pruning the pool shown in Figure 2b

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
$BC_{search}(LT = \infty)$										
$\|E\|$	3	7	6	8	7	7	7	6	7	7
$\|\mathcal{P}\|$	20	40	60	80	100	120	140	160	180	200
$BC_{search}(LT = 1)$										
$\|E\|$	6	6	7	7	3	3	7	7	3	3
$\|\mathcal{P}\|$	20	22	24	23	25	23	25	25	24	25
$BC_{search}(LT = 3)$										
$\|E\|$	6	6	7	7	7	7	3	3	7	7
$\|\mathcal{P}\|$	20	40	52	57	49	50	50	46	47	52
$B_{search}(LT = 5)$										
$\|E\|$	6	6	7	7	7	7	8	8	8	8
$\|\mathcal{P}\|$	20	40	60	80	92	97	92	93	94	92

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}
$BC_{search}(LT = \infty)$										
$\|E\|$	5	8	10	9	8	9	8	8	9	12
$\|\mathcal{P}\|$	20	40	60	80	100	120	140	160	180	200
$BC_{search}(LT = 1)$										
$\|E\|$	5	5	10	10	3	3	8	8	9	9
$\|\mathcal{P}\|$	20	25	30	26	29	30	31	31	30	31
$BC_{search}(LT = 3)$										
$\|E\|$	5	5	8	8	8	8	9	9	8	8
$\|\mathcal{P}\|$	20	40	46	53	55	56	55	54	54	52
$B_{search}(LT = 5)$										
$\|E\|$	5	5	8	8	10	10	9	9	8	8
$\|\mathcal{P}\|$	20	40	60	80	86	97	101	99	95	92

Fig. 3. Average AUCH accuracy achieved when the *incrBC* technique uses BC_{search} and pruning of HMM pools on (a) synthetic and (b) UNM sendmail data. The size of selected EoHMMs and that of the pool are presented for each data block below the graphs. The performance of the BBW algorithm, of *incrBC* without model management (Figure 1), and of *incrBC* when HMMs are combined without any pruning, $LT = \infty$ (Figure 2), are also shown for reference.

size of the pool. A fixed-size pool may be maintained by tuning the tolerance and LT values upon receiving new blocks of data.

4 Conclusions

This paper presents a new ensemble-based technique called incremental Boolean combination (*incrBC*) for incremental learning of new training data according to a learn-and-combine approach. Given a new block of training data, a diversified pool of base classifiers is generated from the data, and their responses are combined with those of previously-trained classifiers in the ROC space. A BC_{search} algorithm selects accurate ensemble of classifiers for operations. This technique allows to adapt Boolean fusion functions and decision thresholds over time, while punning redundant base classifiers. The proposed system is capable

of changing its desired operating point during operations, and hence allow to account for changes in prior probabilities and costs of errors.

During simulations conducted on both synthetic and real-world host-based intrusion detection data using HMMs, the proposed system has been shown to achieve a higher level of accuracy than when parameters of a single best HMM are estimated, using reference batch and incremental learning techniques. It also outperforms ensemble techniques that use the median and majority vote fusion functions to combine new and previously-trained HMMs. The system has shown to form compact ensembles for operations, while maintaining or improving the overall system accuracy. Pruning has been shown to limit the pool size from increasing over time, without negatively affecting the overall ensemble accuracy.

The robustness of the proposed learn-and-combine approach depends on maintaining a representative validation set over time, for selection of base classifiers, and for adaptation of decision thresholds and Boolean functions. Future work involves applying *incrBC* to real-world problems that feature heavily imbalanced data sampled from dynamically-changing environments.

References

1. Connolly, J.F., Granger, E., Sabourin, R.: An adaptive classification system for video-based face recognition. Information Sciences (2010) (in Press)
2. Khreich, W., Granger, E., Miri, A., Sabourin, R.: A comparison of techniques for on-line incremental learning of HMM parameters in anomaly detection. In: Proc. 2nd IEEE Int'l Conf. on Computational Intelligence for Security and Defense Applications, Ottawa, Canada, July 2009, pp. 1–8 (2009)
3. Khreich, W., Granger, E., Miri, A., Sabourin, R.: Iterative Boolean combination of classifiers in the ROC space: An application to anomaly detection with HMMs. Pattern Recognition 43(8), 2732–2752 (2010)
4. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, Chichester (2004)
5. Mizuno, J., Watanabe, T., Ueki, K., Amano, K., Takimoto, E., Maruoka, A.: Online estimation of hidden Markov model parameters. In: Proc. 3rd Int'l Conf. on Discovery Science, vol. 1967, pp. 155–169 (2000)
6. Polikar, R., Upda, L., Upda, S., Honavar, V.: Learn++: An incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man and Cybernetics, Part C 31(4), 497–508 (2001)
7. Tan, K., Maxion, R.: Determining the operational limits of an anomaly-based intrusion detector. IEEE Journal on Selected Areas in Communications 21(1), 96–110 (2003)
8. Tao, Q., Veldhuis, R.: Threshold-optimized decision-level fusion and its application to biometrics. Pattern Recognition 41(5), 852–867 (2008)
9. Tsoumakas, G., Partalas, I., Vlahavas, I.: An ensemble pruning primer. Applications of Supervised and Unsupervised Ensemble Methods 245, 1–13 (2009)
10. Tulyakov, S., Jaeger, S., Govindaraju, V., Doermann, D.: Review of classifier combination methods. In: Marinai, S., H.F (eds.) Studies in Comp. Intelligence: ML in Document Analysis and Recognition, pp. 361–386. Springer, Heidelberg (2008)
11. Warrender, C., Forrest, S., Pearlmutter, B.: Detecting intrusions using system calls: Alternative data models. In: Proc. IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, pp. 133–145 (1999)