An Anomaly Detection System based on Ensemble of Detectors with Effective Pruning Techniques

Amirreza Soudi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

December 2015

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis proposal prepared

By: **Amirreza Soudi**

Entitled: **An Anomaly Detection System based on Ensemble of Detectors with Effective Pruning Techniques**

and submitted in partial fulfilment of the requirements for the degree of

Master of Applied Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. VVVVVVVVV

_____ Dr. XXXXXX , Examiner, External

_____ Dr. YYYYYYY, Examiner

_____ Dr. ZZZZZZZZ, Supervisor

_____ Dr. WWWWWWWW, Supervisor

Approved by _____

Dr. XXXXXXX, Chair

Department of Electrical and Computer Engineering

_____

Dr. DDDDDDDDDD

Dean, Faculty of Electrical and Computer Engineering

ABSTRACT

Anomaly detection systems rely on machine learning techniques to model the normal behavior of the system. This model is used during operation to detect anomalies due to attacks or design faults. Ensemble methods have been used to improve the overall detection accuracy by combining the outputs of several accurate and diverse models. Existing Boolean combination techniques either require an exponential number of combinations or sequential combinations that grow linearly with the number of iterations, which make them difficult to scale up and analyze. In this paper, we propose PBC (Pruning Boolean Combination), an efficient approach for selecting and combining anomaly detectors. PBC relies on two novel pruning techniques that we have developed to aggressively prune redundant and trivial detectors. Compared to existing work, PBC reduces significantly the number of detectors to combine, while keeping similar accuracy. We show the effectiveness of PBC when applying it to a large dataset.

Dedicated to

My Family

ACKNOWLEDGEMENTS

None of this would have been possible without the love and patience of my family. My father, my mother, and my sister, to whom this thesis is dedicated, have been a constant source of love, concern, support and strength all these years. I would like to express my heart-felt gratitude to my family first.

My deepest gratitude is to my supervisor, Dr. Wahab Hamou-Lhadj. I have been amazingly fortunate to have a supervisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. His patience and support helped me overcome many crisis situations and finish this thesis. I am grateful to him for holding me to a high research standard and enforcing strict validations for each research result. Beside I owe a dept of gratitute to my supervisor's postdoce, Dr. Wael Khreich, who has been always there to listen and give advice. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work. His insightful comments and constructive criticisms at different stages of my research were thought-provoking and they helped me focus my ideas.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Intrusion Detection Systems (IDSs) have become important tools that help security administrators to identify and analyze unauthorized computer or network activities, from both outsider and insider attacks. The dominant detection methodologies are signature-based and anomaly-based approaches. Signature-based (or misuse) IDSs look for events that correspond to patterns of known attacks. They can provide a high level of accuracy, however are limited to detecting known attacks and vulnerable to polymorphic attacks (which are capable of changing their signatures as they propagate).

Anomaly detection systems (ADSs), on the other hand, monitor for significant deviations from normal system behavior. They are typically trained (using machine learning and data mining techniques) on datasets collected over a period of attack-free activities to learn the normal system behavior, and then deployed to detect deviations from the expected system behavior. These deviations are reported as anomalous events, although they are not necessarily malicious activities or attacks because they may also correspond to coding or configuration errors. An ADS can detect novel attacks (not known during training time) but generate large number of false alarms, because it is difficult to obtain complete descriptions of complex system

behavior, which may change over time.

IDSs can also be categorized depending on their location of deployment into network- and host-based detection systems (a more comprehensive taxonomy of IDSs can be found in [1]). Network-based IDSs monitor and analyze traffic to and from all devices on the network. They may be be located anywhere in the network; integrated in network devices (e.g., switches and routers) or as a standalone device. A host-based IDS runs on a host computer and monitors sensitive activities on this host, such as unauthorized access, modification of files, or system calls.

In this work, we focus on host-based anomaly detection using system calls – the gateway between user and kernel mode. Short sequences of system calls have been shown consistent with normal host operation, and can be used to detect attacks [2,3]. A large number of research studies have investigated different machine learning and data mining techniques for detecting anomalies in system call sequences [4]. Among these, techniques based on Hidden Markov Models (HMMs) – probabilistic models for sequential data – have been shown to produce a high level of detection accuracy [3, 5–13].

The success of an ADS depends largely on the model of normal behavior. A single HMM may not, however, provide adequate approximation of the underlying data distribution of a complex host system behavior, due to the many local maxima of the likelihood function [11]. Ensemble methods have been used to improve the overall system accuracy by combining the outputs of several *accurate and diverse* models [14–17]. In particular, combining the outputs from multiple HMMs, each trained with a different number of states, in the Receiver Operating Characteristics (ROC) space according to the Iterative Boolean Combination (IBC), has been shown to provide a significant improvement in the detection accuracy of system call anomalies [18].

The IBC is a general decision-level combination technique that attempts to select the decision thresholds (from each input detector) and the Boolean functions

that maximize the overall ROC convex hull of the combined ensemble [18]. As presented in Algorithm 1, given K soft detectors that assign scores or probabilities to the input samples, (which can be converted to a crisp detector by setting a threshold on the scores as further detailed in section 2.1) , the IBC algorithm starts by combining the outputs of the first two detectors (in the order of input), and then proceeds *sequentially* by combining the resulting combinations with the outputs of third detector, and so on, until the K^{th} detector is combined. It can then re-iterate to combine the resulting combinations with the original detectors.

The IBC technique provides a practical way for combining relatively large number of detectors while avoiding the exponential explosion of Boolean combinations. As detailed in Section 3.4, applying all Boolean functions using an exhaustive bruteforce search to determine optimal combinations leads to an exponential number of combinations, which is prohibitive even for a small number of detectors [19]. Even the *pairwise* Bruteforce Boolean Combination (BBC2), which is used as a baseline reference in our experiments, requires an exponential number of combinations, which is equal to the square of the number of detectors (see Algorithm 2 without the pruning mechanisms).

However, the sequential combinations of soft detectors according to IBC still faces challenges, when a large number of detectors (K) is presented for combinations. First, it produces a sequence of combination rules that grows linearly with K (and the number of iterations), which becomes difficult to analyse and understand. Furthermore, it makes the algorithm sensitive to the order in which the detectors are input for combinations, which increases the effort required to find best subset for operations. In any case, combining all available detectors without any pruning mechanism may be inefficient due to the redundancy in their outputs.

In this work, we propose an ADS based on a Pruned Boolean Combination (PBC) algorithm, which employs two novel pruning techniques to select a subset

of diverse and accurate detectors for combination, while discarding the remaining ones. Although both pruning techniques are based on Cohen's Kappa [20] measure, they differ in the way they compute the diversity and accuracy of the selected detectors for combinations. While the first technique relies only on Kappa measure (MinMax-Kappa), the second uses both Kappa and the ROC convex hull (ROCCH-Kappa). Therefore, our proposed PBC approach provides an efficient way to prune and combine large number of detectors, which avoids the exponential explosion of combinations of brute-force techniques, and reduces the sequence of combinations provided by IBC.

We evaluate our PBC-based ADS using ADFA Linux Dataset (ADFA-LD)[1], which has been recently made publicly available on the website of the University of New South Wales [21]. The performance of the PBC using both pruning techniques are compared to that of IBC and BBC2 in terms of ROC analysis and time complexity. The results show that PBC with both pruning techniques are capable of maintaining similar overall accuracy as measured by the ROC curves to that of IBC and BBC2. However, the time required for searching and selecting (or pruning) the subset of detectors from the same ensemble of detectors is on average three magnitudes lower for PBC with MinMax-Kappa pruning than that of BBC2. Furthermore, the experimental results show that PBC with both pruning techniques always provides two crisp detectors for combination (during the operations), while IBC provided an average of 11 detectors to achieve the same operating point.

1.1 System Calls

A system call is a request in a Unix-like operating system made via a software interrupt by an active process for a service performed by the kernel. Users' requests are made from higher level applications to the kernel via a platform-dependent set of

[1]http://www.cybersecurity.unsw.adfa.edu.au/ADFA IDS Datasets/

system calls. System calls provide an essential interface between a process and the operating system. Applications must invoke system calls to request kernel services, such as access to standard input/output, physical devices and network resources. Every system call has a unique number (known by the kernel) and a set of arguments. Some example of system calls used for file managements is as follows: open(), read(), write(), and close(). System call traces have different lengths that depends on the complexity and execution time of the process. Manifestations of a wide variety of attacks, including buffer overflows, symbolic link, decode and SYN floods may appear at the system call level and differ from normal behavior of privileged processes, ([2]; [22]; [23]; [24]; [3]). Furthermore, after gaining root privileges on a host, attackers will typically try to maintain administrative access on the compromised host ([25]), by for instance installing backdoors and rootkits. Attackers will also attempt to distribute Trojan horses, using for instance comprised email addresses or shared network folders, to compromise other systems. These malicious activities may also invoke different sequences of system calls than those generated during normal execution of a process.

1.2 HMM-based Anomaly Detection using System Call Sequences

The temporal order of system calls has been shown consistent with the normal behavior of a privileged process, while an unusual burst will occur during an attack [2, 3]. The authors proposed a host-based ADS using a simple sequence matching techniques for detecting anomalous system call sequences generated by UNIX privileged processes [2]. System call traces provided for training are first segmented into fixed-length *contiguous* sequences, using a fixed-size sliding window, shifted by one symbol to build the normal profile. These sequences are then stored in a database that represents the normal process behavior. During operations, a sliding window (having

the same size used to construct the normal profile) to scan the system calls generated by the monitored process for anomalies – sequences that are not found in the normal database.

Several statistical and machine learning techniques have been investigated over the last two decades for detecting system call anomalies [4]. Application of machine learning techniques include neural network [26], k-nearest neighbors [27], Markov models or n-grams [28, 29], Bayesian models [30]. Among these, techniques based on discrete HMMs have been shown to produce a high level of detection accuracy [3, 5–11, 31–33].

A discrete HMM is a stochastic process for sequential data [34,35]. An HMM is determined by two interrelated mechanisms – a latent Markov chain having a finite number of states N, and a set of observation probability distributions, each one associated with a state. Starting from an initial state $S_i \in \{S_1, ..., S_N\}$, determined by the initial state probability distribution π_i, at each discrete-time instant, the process transits from state S_i to state S_j according to the transition probability distribution a_{ij}. The process then emits a symbol v_k, from a finite alphabet $V = \{v_1, \ldots, v_M\}$ of size M symbols, according to the discrete-output probability distribution $b_j(v_k)$ of the current state S_j. HMM is commonly parametrized by $\lambda = (\pi, A, B)$, where the vector $\pi = \{\pi_i\}$ is the initial state probability distribution, matrix $A = \{a_{ij}\}$ is the state transition probability distribution, and matrix $B = \{b_j(v_k)\}$ is the state output probability distribution, $(1 \leq i, j \leq N$ and $1 \leq k \leq M)$.

A well trained HMM provides a compact detector that captures the underlying structure of a process based on the temporal order of system calls, and detects deviations from normal system call sequences with high accuracy and tolerance to noise. Training an HMM from a sequence (or a block) of observation symbols, $o_{1:T}$, aims at estimating HMM parameters λ to best fit the training data. Typically, parameters estimation consists of maximizing the likelihood of the training data over HMM

parameters space, $P(o_{1:T} \mid \lambda)$. Since this likelihood depends on the latent states, there is no known analytical solution to the learning problem. Iterative optimization techniques, such as the Baum-Welch (BW) algorithm [36], are applied to estimate the HMM parameters over several training iterations, until the likelihood function is maximized. During operation, the likelihood of a new observation sequence $o_{1:T}$ given a trained HMM λ, $P(o_{1:T} \mid \lambda)$ is typically evaluated by using the Forward-Backward (FB) algorithm [34,35]. Setting a threshold on the output probabilities of HMM, provides a decision whether the sequence is normal or anomalous.

Few researchers tried to investigate the effect of the number of states on the performance of HMM detectors, and found that N value may have a great impact on the overall performance [11,37]. In particular, combining the outputs from multiple HMMs, each trained with a different N value, in the ROC space according to the IBC technique (as described in the next section), has been shown to provide a significant improvement in the detection accuracy of system call anomalies [18].

1.3 Anomaly Detection Challenges

Most works assume being provided with a sufficiently representative amount of clean (attack-free) system call data for training the anomaly detectors, which is not correct most of the time. Even under this assumpotion these techiqnue have some problmes.

Designing and implementing every machine learning algorithms like HMM involves some estimation of parameters like number of hidden states N, initial matrix M. The value of N has a considerable impact not only on system accuracy, but also on HMM training time and memory requirements. The number of states is often chosen heuristically or empirically using validation data. In fact, HMMs trained with different number of states are able to capture different underlying structures of data. Therefore, a single best HMM will not provide a high level of performance

over the entire detection space. The other challenge is the size of the window, every window-based anomaly detection technique using system-call traces in form of window of system calls, the size of window affect performance and accuracy more or less in different technique. The smaller windows size is always desirable since it makes the technique much faster yet we need to have a limit on minimum size in order to be able to detect all of the anomalies [38]. In general, ADSs are designed to provide accurate results for a particular window size and anomaly size. Discriminative or sequence matching techniques, are only able to detect anomalies with sizes equal to that of the detector window.

As mentioned earlier first and foremost important challenges with anomaly detection techniques are unrepresentative data. in practice is very rare to find a clean data set which represents all of the system behavior so acquiring a sufficient amount of clean data that represents the normal behavior of complex realworld processes is a challenging task. Usually it's not possible to collect all the system behavior and it leads to have an incomplete view of the normal process behavior. Incomplete view of the normal behavior leads to misclassifying rare normal events as anomalous, as illustrated in Figure 1.1

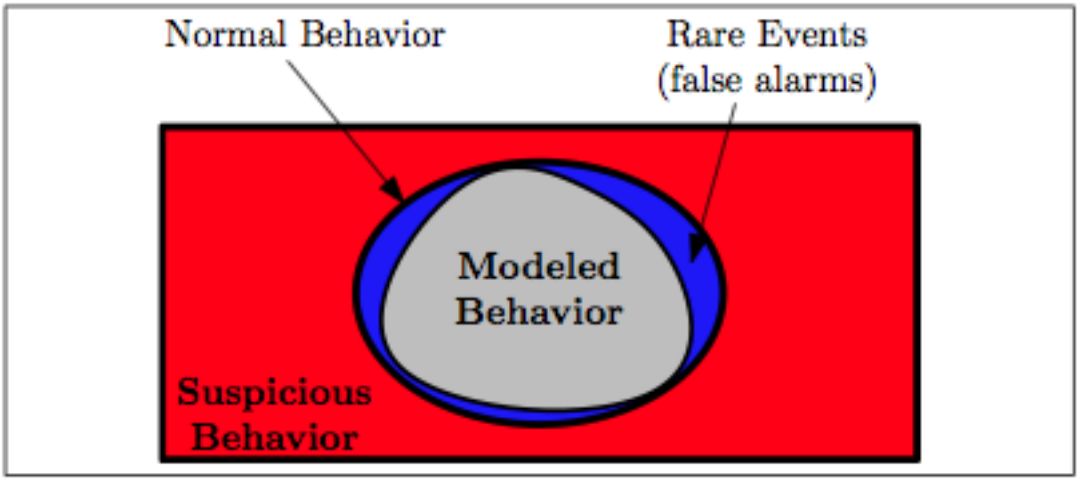


Figure 1.1: Illustration of normal process behavior when modeled with unrepresentative training data. Normal rare events will be considered as anomalous by the ADS, and hence trigger false alarms

Even if there is a way to model and collect all the system behavior system call these behaviors will change over the course of time (either by change of user usage behavior or by changes in monitored environment) and the training model and the clean data set should get updated. Changes to the underlying normal distribution lead the model to deviate, and generate both false positive and negative error as illustrated in Figure 1.2 Accordingly, the model must be updated from new data to account from this change.

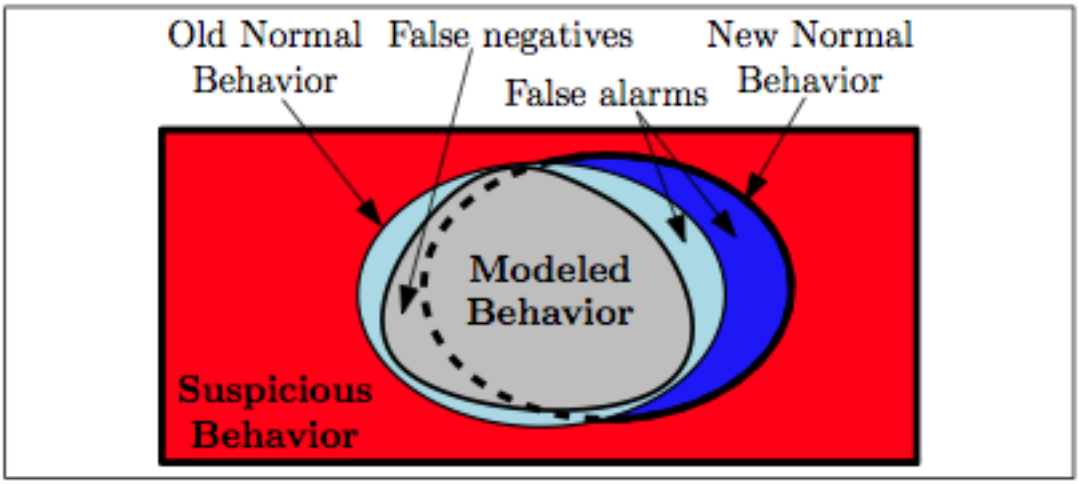


Figure 1.2: Illustration of changes in normal process behavior, due for instance to application update or changes in user behavior, and the resulting regions causing both false positive and negative errors

Chapter 2

Detector Combination

2.1 Boolean Combination of Detectors in the ROC Space

This section provides information about ROC analysis and summarizes the Boolean combination in the ROC space [18]. A crisp detector outputs a decision or a class label (e.g., normal or anomaly) while a soft detector such as HMM assigns scores to the input samples, which can be converted to a crisp detector by setting a decision threshold on the scores. Given the responses of a crisp detector on a validation set, the true positive rate (tpr) is the proportion of positives correctly classified over the total number of positive samples. The false positive rate (fpr) is the proportion of negatives incorrectly classified over the total number of negative samples. The positive (or target) class is typically the class of interest, which is the anomalous class for an ADS.

A ROC curve is a plot of tpr against fpr [39]. A crisp detector produces a single data point in the ROC space, while a soft detector produces a ROC curve by varying the decision thresholds. In practice, an empirical ROC plot is obtained by connecting the observed (tpr, fpr) pairs of a soft detector at each decision threshold. A point a is

superior to another point b in the ROC space, if $fpr(a) \leq fpr(b)$ and $tpr(a) \geq tpr(b)$. The ROC convex hull (ROCCH) is therefore the outer envelope connecting superior points in the ROC space. A ROC curve allows to visualize the performance of detectors and to select optimal operational points, without committing to a single decision threshold or to fixed error costs. The area under the ROC curve (AUC), or under the ROCCH, provides a general measure for evaluation and selection of detectors [39].

The IBC is a general decision-level combination technique that attempts to select the decision thresholds (from each input detector) and the Boolean functions that maximize the overall ROCCH of the combined ensemble [18]. The core of IBC (only for the first iteration) is described in Algorithm 1.

The IBC applies each Boolean function to combine the responses corresponding to each decision threshold from the first detector to those from the second detector. Fused responses are then mapped to vertices in the ROC space, and their ROC convex hull (ROCCH) is computed. Vertices that are superior to the ROCCH of original detectors are then selected. The set ($\mathcal{S}$) of decision thresholds from each detector 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 detector are then combined with the responses of each emerging vertex, and so on, until the last detector 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. Although not shown in Algorithm 1, the original IBC algorithm can iterate by re-combining the resulting combination (of all detectors) on the ROCCH with each of the original detectors (sequentially) until the ROCCH stops improving [18].

However, as stated previously, combining all available detectors without pruning may be inefficient due to the redundancy in their outputs. The sequential combinations of soft detectors, illustrated in the loop in line 16 of Algorithm 1, produces a sequence of combination rules that grows linearly with K (and the number of iterations), which becomes difficult to track, analyse and understand when the value of K becomes large. In addition, the IBC algorithm is sensitive to the order in which the detectors are input for combinations, which increases the effort required to find best subset for operations.

2.2 Synthetic Data Set

In order to get a better insight about the reason of the combination and selecting the proper detectors among millions of trained detectors, I am going to provide the full example of training, pruning and combining detector on systetich data set. This way it is much easier to see and follow what happens from begging.

For this purpose I chose 2 famous synthetic data sets.

1. Lithuaninan
2. Banana (Moon)
3. Circle

unlike the real data set it's possible to plot these data set and shows the trained model over them, and this makes it easier to see what happens when we prune some unnecessary trained model and only keep those that have some meaning.

The following plots show the preliminary state of 3 above-mentioned data sets.

Another purpose of this example is to show that you can reach the same level of detection by combining very simple and naive model (detector) instead of spending time to find the proper model and wait to train that model on data set, which is usually the time consuming task. In this example, hundreds of linear detector (which

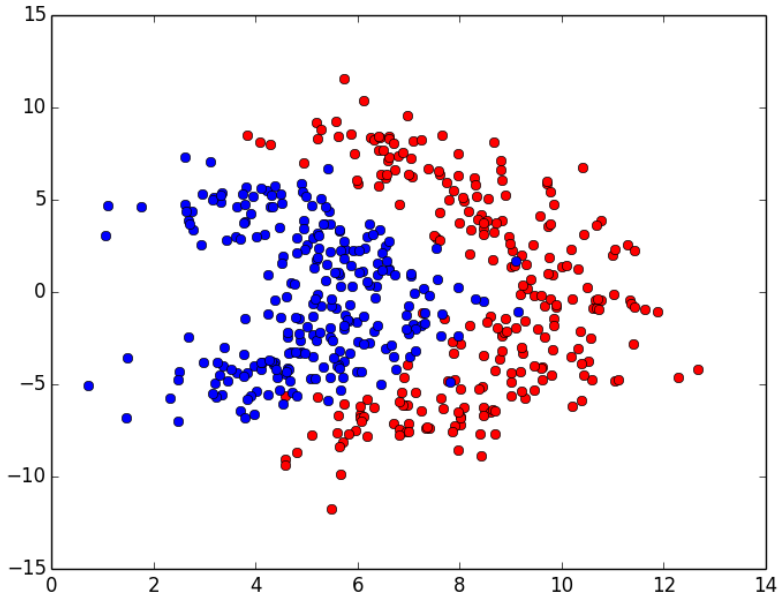


Figure 2.1: Lithuanian Data Set

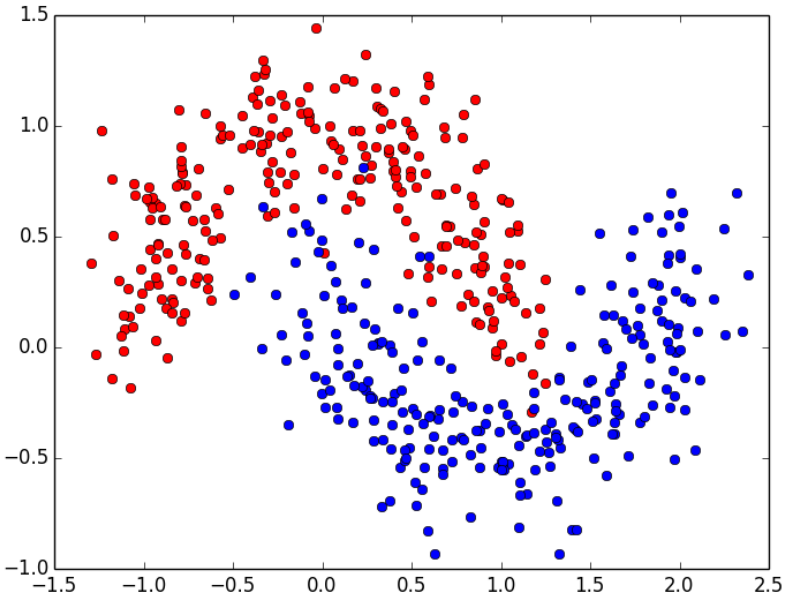


Figure 2.2: Banana (Moon) Data Set

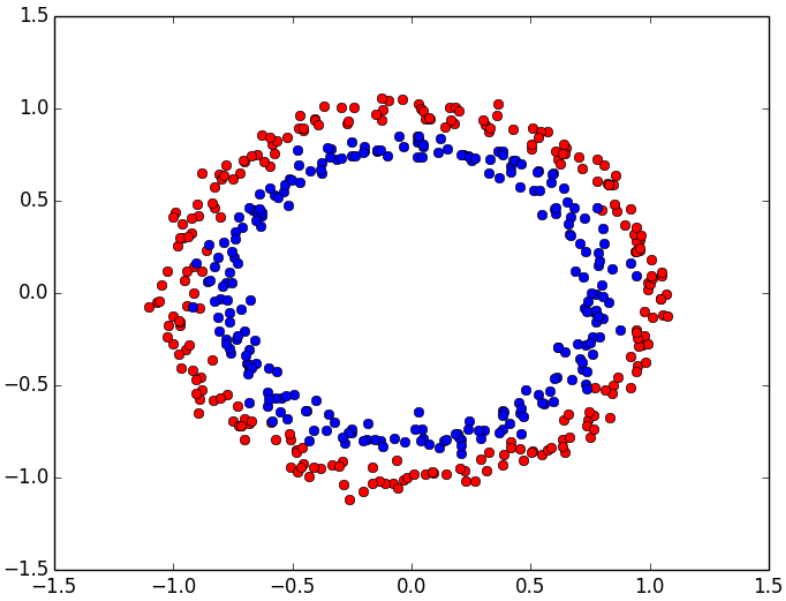


Figure 2.3: Circle Data Set

is very fast, but it's not accurate) are trained over a data set. In order to have diverse learner detectors we changed the ratio of data and their label during training to get as diverse detectors as possible.

As shown below, we have different trained detectors on these data sets which none of them can detect all the classed but as we will show later we can use them to better classify the data set by combining these simple detectors.

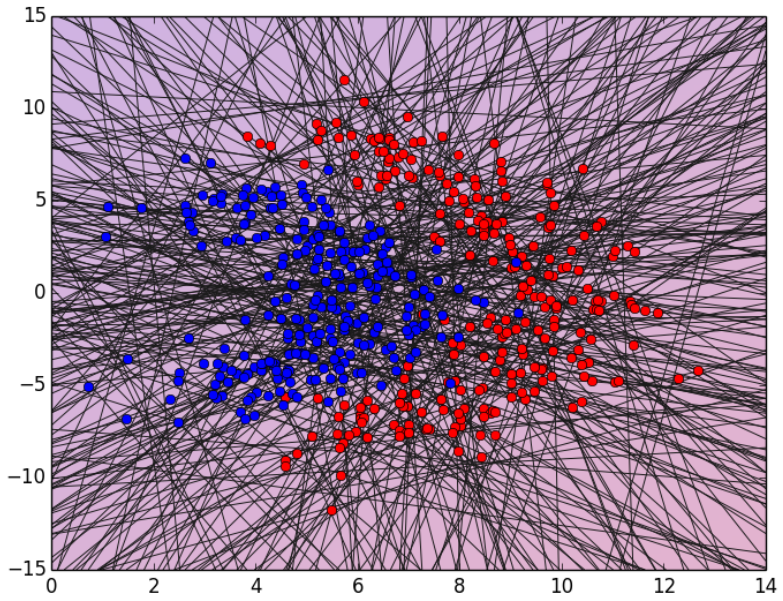


Figure 2.4: Linear detectors trained on the Lithuanian Data Set

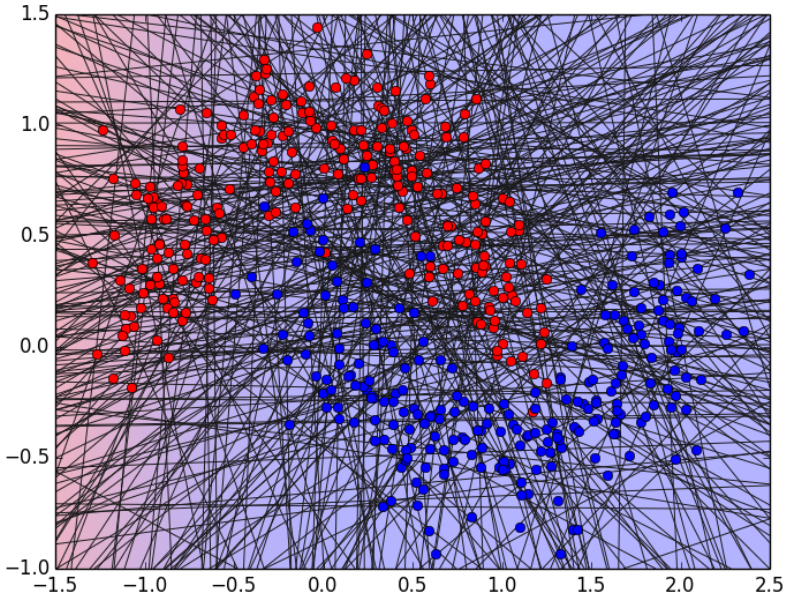


Figure 2.5: Linear detectors trained on the Banana (Moon) Data Set

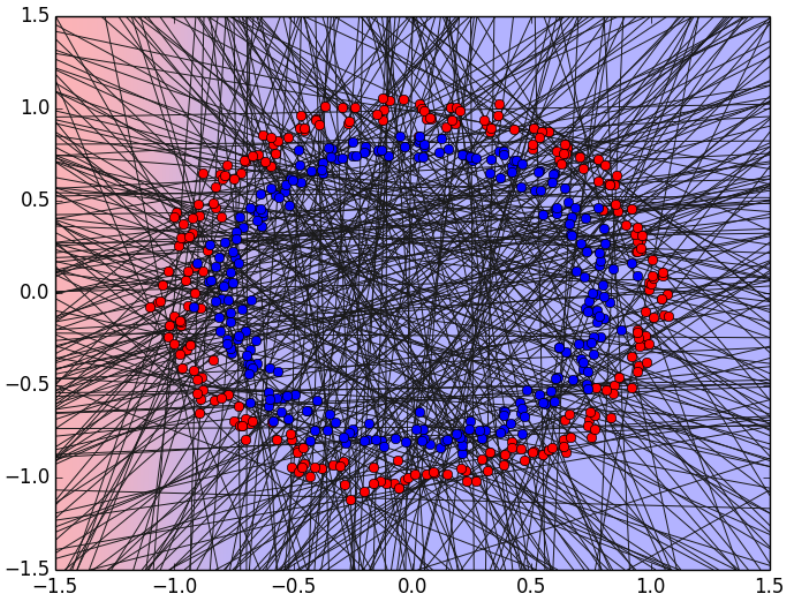
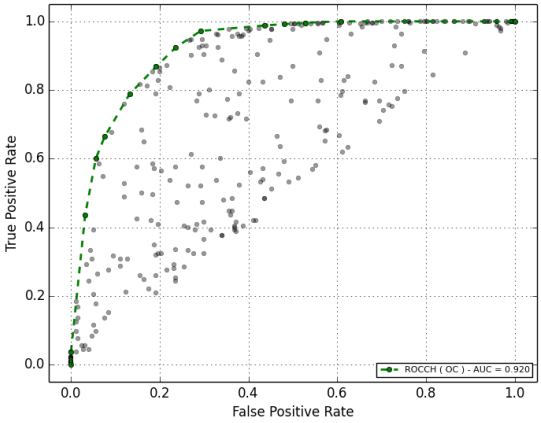
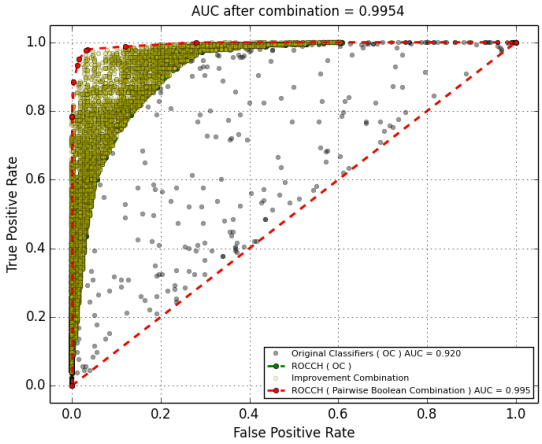


Figure 2.6: Linear detectors trained on the Circle Data Set

By selecting the right models and combine them together we can increase the detection rate, as below plots show; the plot on the left show all the original detectors and tehir ROC before combination, after selecting the proper detectors and combine them together (after just one iteration) we can reach the right plot which shows the combined detectors (in yellow) and their improved ROC curve, which shows more accurate result. The next step is to show why combination proivdes better result and how we can get the same result just with combining part of trained models.

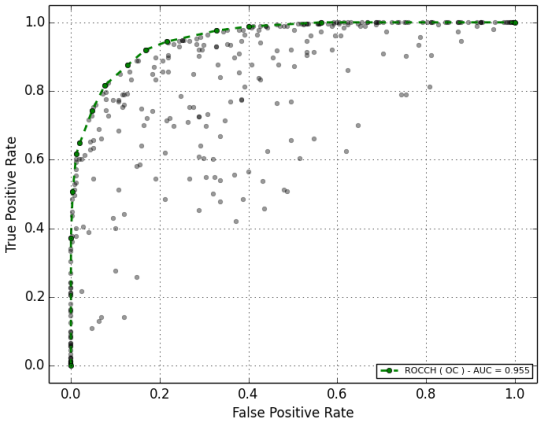


(a) ROC before combination

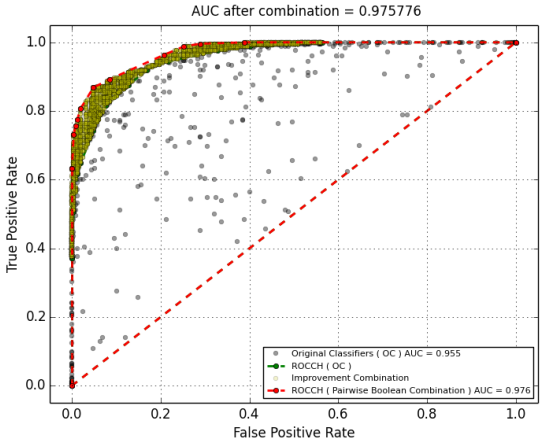


(b) ROC after combination

Figure 2.7: ROC comparison on Lithuanian dataset before and after first round of combination

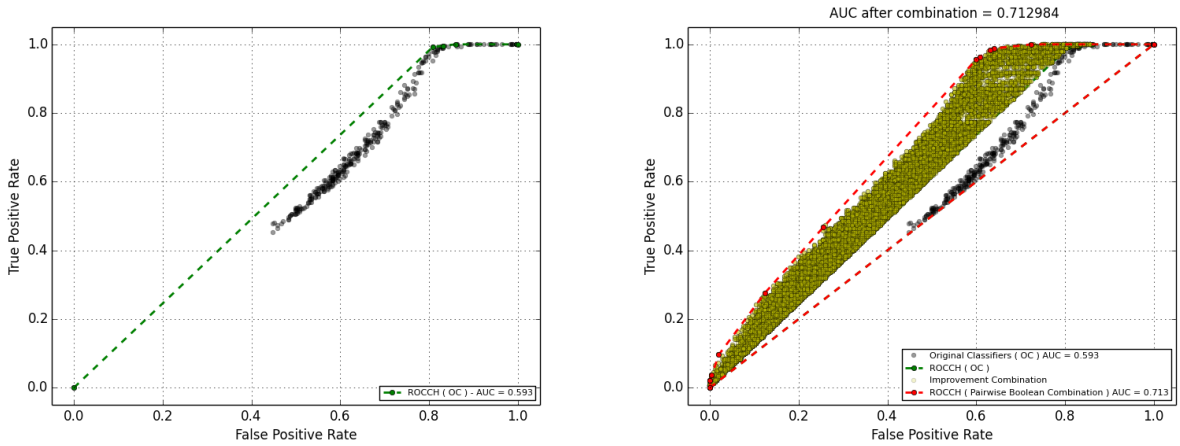


(a) ROC before combination



(b) ROC after combination

Figure 2.8: ROC comparison on Banana dataset before and after first round of combination



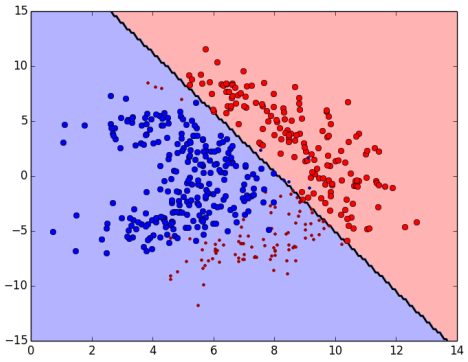
(a) ROC before combination

(b) ROC after combination

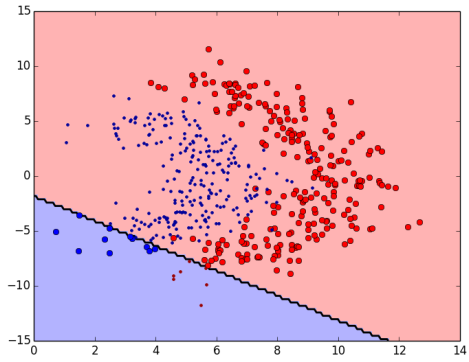
Figure 2.9: ROC comparison on Circle dataset before and after first round of combination

For each data set we are going to show there is no single detectors that can classify the dataset properly, yet there are at least two detectors that can complete each other and produce better results.

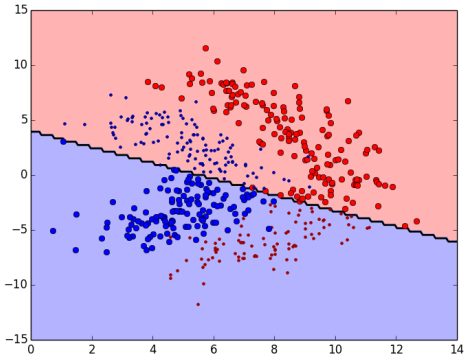
As it's shown in Figure 2.10 with just one LDA we cannot classify all the classes 100%. âĂŃBut with closer look at this data set and our LDA we can see that it's possible to choose two of these detectors and combine them and get a better result. As it's shown in Figure 2.11 we select detector from Firgure 2.10a and detector form 2.10d and combine them together.



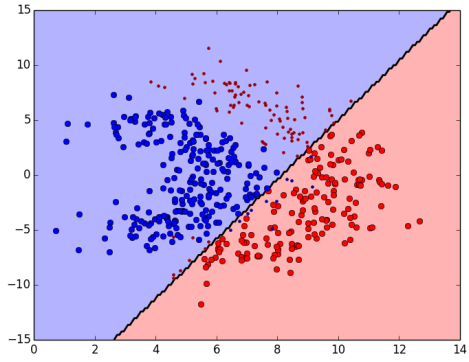
(a) First subfigure



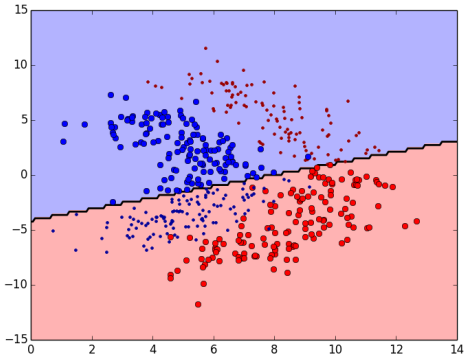
(b) Second subfigure



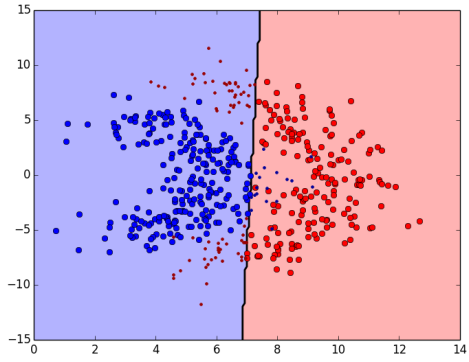
(c) Third subfigure



(d) Fourth subfigure

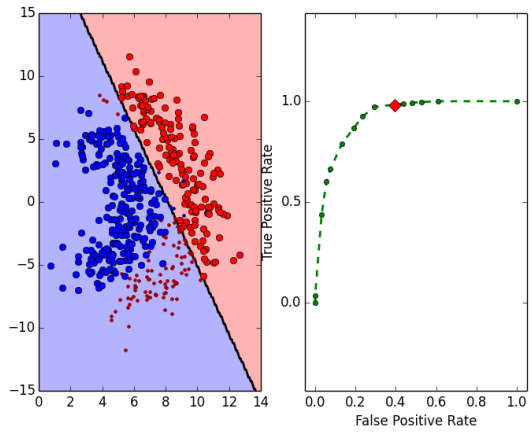


(e) Fifth subfigure

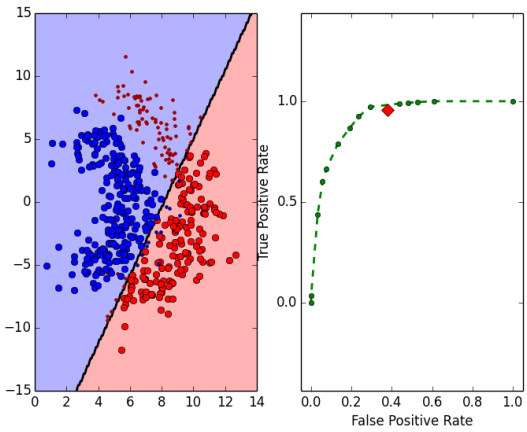


(f) Sixth subfigure

Figure 2.10: Some of best LDA (Linear Discriminant Analysis) that can be trained on Lithuanina data set



(a) First selected LDA



(b) Second selected LDA

Figure 2.11: Selected LDA and their respective point compare to current ROC

If we combine these two detectors with help of AND operation as you can see the result in Figure 2.12 we can get a better result. The big red point is the mapping of the combined detectors, and it's clear and it's improved since it's passed the current ROC and if we recalculate the ROC based on this new generated point we will have greater AUC.

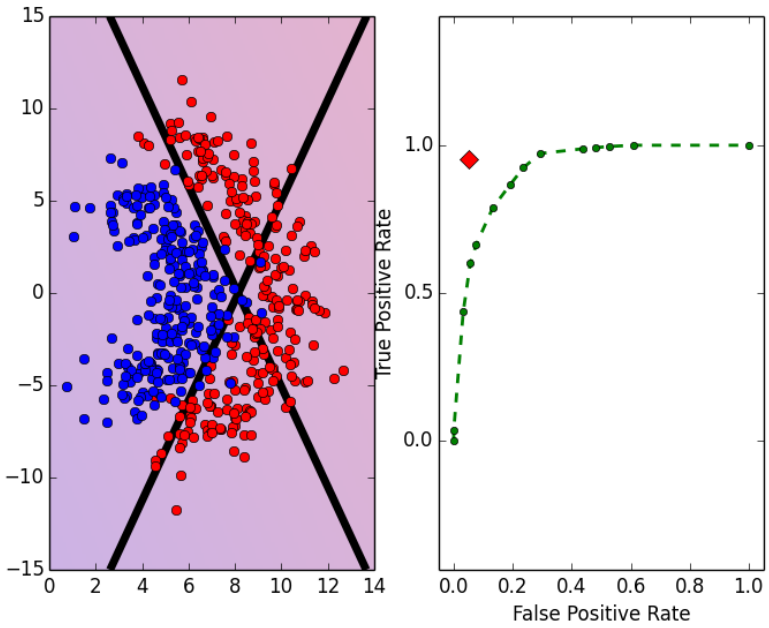
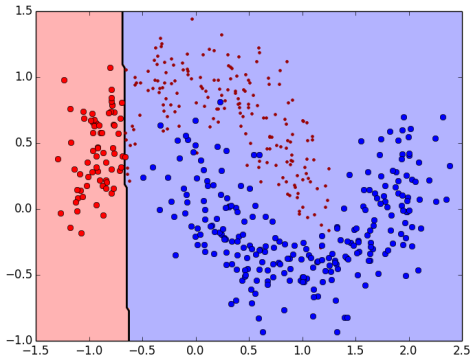


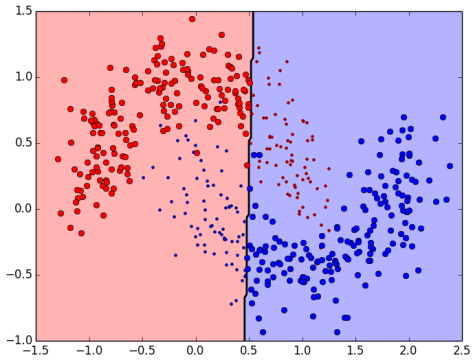
Figure 2.12: Selected detectors and respective combined point on ROC curve

The very similar approach can be applied to Banana data set with a little bit different. As you can see in Figure 2.13 like Lithuanianan data set, it is not possible to classify the data set completely but here we cannot classify the dataset completely even with help of 2 different detectors. In this specific example the minimum number should be 3 in order to classify data in an accurate way. That shows the best approach is to select and combine detectors in an irrational way so in every phase two of the best accurate and diverse detritus should be selected and get combined and insert the result back to the pool of detectors and make the whole process from the beginning till there is no major improvement.

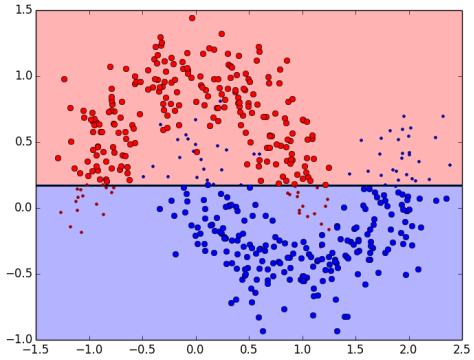
Based on the observation in this example it is better to select 3 detectors in order to be classify the data set, if we select detector in Figure 2.13a, Figure 2.13c and Figure 2.13f and combine them together we should expect to get the most accurate result among all other detectors. in figure 2.14 you can see all these detectors in one plot.



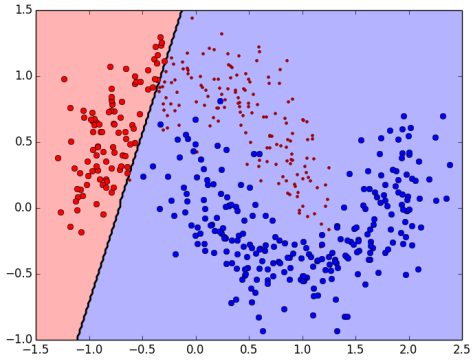
(a) First subfigure



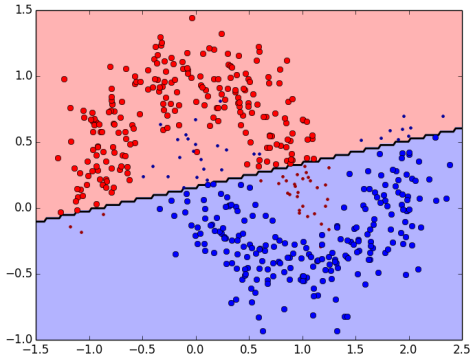
(b) Second subfigure



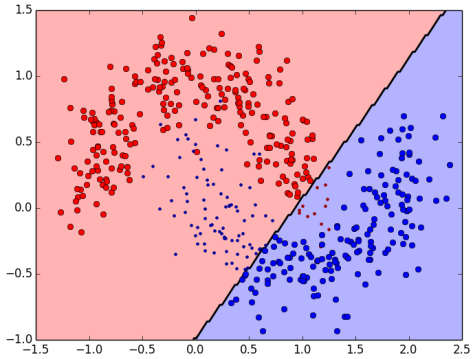
(c) Third subfigure



(d) Fourth subfigure



(e) Fifth subfigure



(f) Sixth subfigure

Figure 2.13: Some of best LDA (Linear Discriminant Analysis) that can be trained on Banana data set

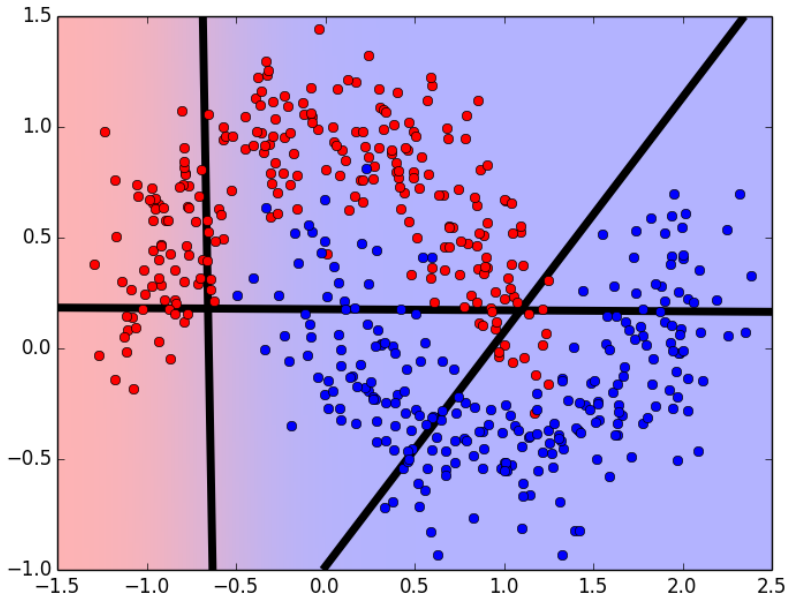
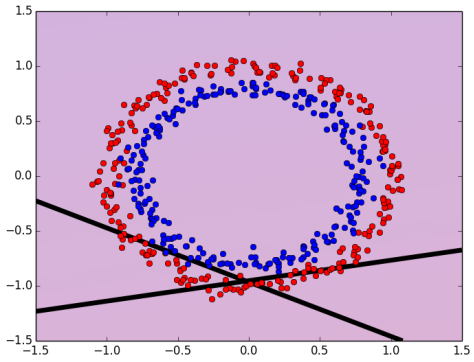


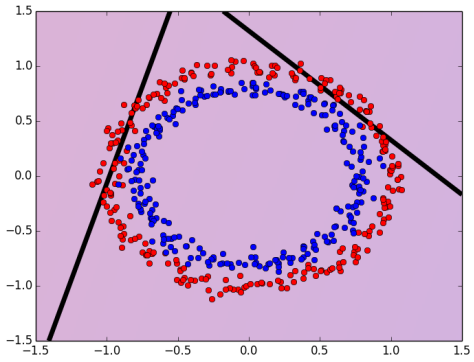
Figure 2.14: Best potential selected detectors on Banana data set

These examples show that it is not always necessary to train complex model ,which takes time and resources, over the data set to get the best result possible, the same result can be achieved by combining two or more simple models which can be trained much faster, and also by increasing number these simple models we are sure that we cover most of the cases and even outliers in the data set. Besides, even though we decide to train some complex model there is no guarantee that you can get the best result since every model has its own weakness. At the end some kind of combination (combining with the model that complement the complex model's error) will help to improve the result. In the next chapter our focus is to show how we can select those meaningful models that help to improve the result since it is not possible or even rational to train thousands of models and combine them all together. Also, we should keep in mind that in some cases we need to go for more than one iteration that means combining every single trained together models take more time or even it is impossible.

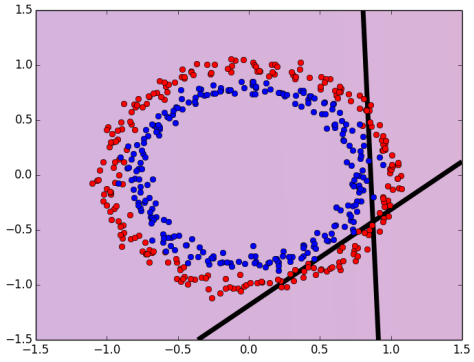
The Cricle data set is a good example that doing combination in an iterative way will improve the result, as it is shown in figure 2.15 There are not any two detectors that can be selected and combined together in order to classify the whole data set completely, yet if we select all of these good detectors and start by combining two of them and for next iteration combine the result with the next one as you can see in figure 2.16 we can classify the whole data set with the help of few simple detectors.



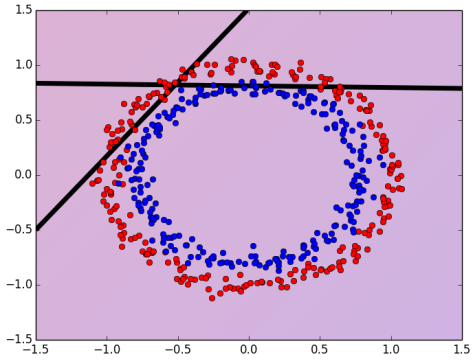
(a) First subfigure



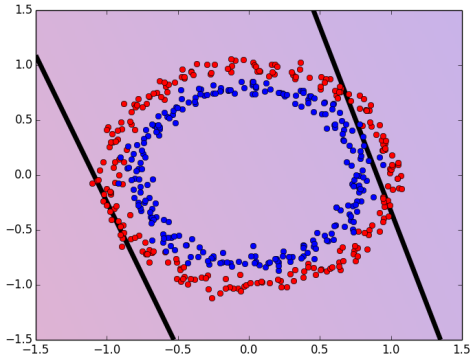
(b) Second subfigure



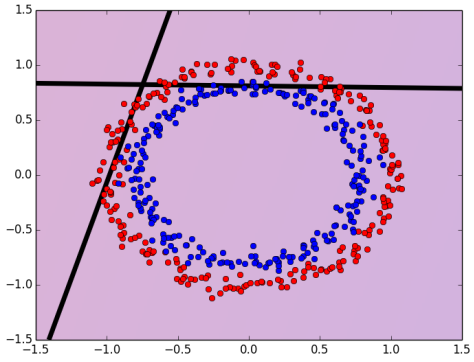
(c) Third subfigure



(d) Fourth subfigure



(e) Fifth subfigure



(f) Sixth subfigure

Figure 2.15: Series of 2 selected LDA (Linear Discriminant Analysis) that can be trained and combined on Circle data set

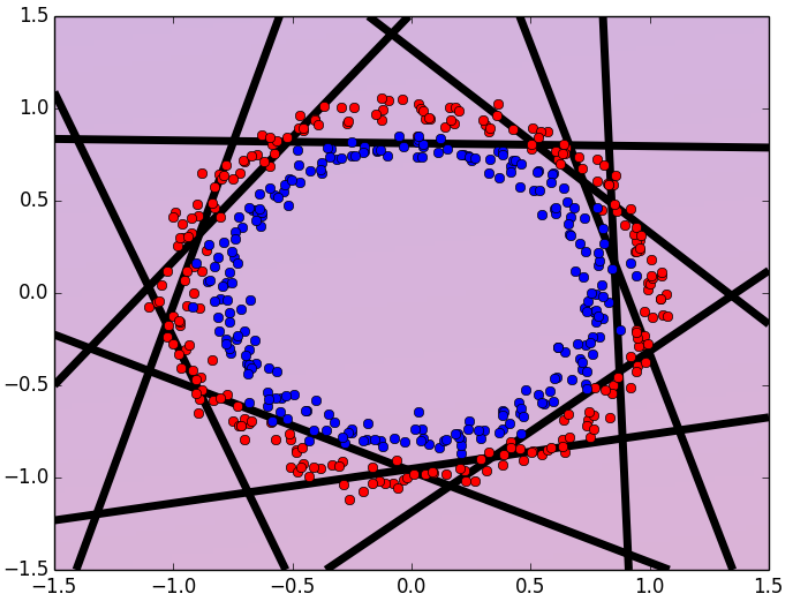


Figure 2.16: Best potential selected detectors on Circle data set

Algorithm 1: IBC($D_1, D_2, \ldots, D_K, \mathcal{V}$): Iterative Boolean Combination

input : K soft detectors $(D_1, D_2, \ldots, D_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$

output: ROCCH of combined detectors.

- Each vertex is the result of 2 to K combination of crisp detectors.
- Each combination selects the best decision thresholds from different detectors (D_i, t_j) and Boolean function (stored in the set $\mathcal{S}$)

$n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_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 detectors (D_1 and D_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**

    $\boldsymbol{R_1} \leftarrow (D_1, t_i)$ // responses of D_1 at decision threshold t_i using $\mathcal{V}$.

    for $j \leftarrow 1$ **to** n_2 **do**

      $R_2 \leftarrow (D_2, t_j)$ // responses of D_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 func.

      compute (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combination to ROC plane

      push (tpr, fpr) onto $\boldsymbol{F}$

  compute $ROCCH_2$ *of all ROC points in* $\boldsymbol{F}$

  $n_{ev} \leftarrow$ number of emerging vertices

  $\mathcal{S}_2 \leftarrow \{(D_1, t_i), (D_2, t_j), bf\}$ // set of selected decision thresholds from each detector 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**

        $\boldsymbol{R_k} \leftarrow (D_k, t_j)$

        $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_i}, \boldsymbol{R_k})$

        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

  $\mathcal{S}_k \leftarrow \{\mathcal{S}_{k-1}(i), (D_k, t_j), bf\}$

  // $\mathcal{S}_k$ is the set of the selected subsets from the previous combinations; the decision thresholds from the newly-combined detector; and the Boolean functions that yields to the emerging vertices on the ROCCH.

store $\mathcal{S}_k : 2 \leq k \leq K$

return $ROCCH_K$

Algorithm 2: PBC($D_1, D_2, \ldots, D_K, \mathcal{V}$): Pruned Boolean Combination

input : K soft detectors $(D_1, D_2, \ldots, D_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$

output: ROCCH of combined detectors.

- Each vertex is the result of exact 2 combination of crisp detectors.
- Each combination selects the best decision thresholds from different detectors (D_i, t_j) and Boolean function (stored in the set $\mathcal{S}$)

$n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_k).

let $n = \sum_{k=1}^{K} n_k$

$BooleanFunctions \leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$

allocate $\boldsymbol{C}$ an array of size: $[|\mathcal{V}|, n]$ // storage of all crisp detectors' decisons.

convert soft detectors to crisp detectors

for $i \leftarrow 1$ **to** K **do**

  for $j \leftarrow 1$ **to** n_i **do**

    $\boldsymbol{R} \leftarrow (D_i, t_j)$ // responses of D_i at decision threshold t_j using $\mathcal{V}$.

    push $\boldsymbol{R}$ onto $\boldsymbol{C}$

---

choose Pruning Technique {MinMax-Kappa, ROCCH-Kappa}

reduce n to U // $U \ll n$: is a user defined max number of detectors

return $\mathcal{C}_{\text{selected}} \leftarrow \boldsymbol{C}$ - Pruned Detectors

// Subset of size U detectors selected from all original detectors and returned for combination

---

allocate $\boldsymbol{F}$ an array of size: $[2, U^2 \times size(BooleanFunctions)]$ // temporary storage of combination results.

foreach $bf \in BooleanFunctions$ **do**

  for $i \leftarrow 1$ **to** U **do**

    $\boldsymbol{R_1} \leftarrow \mathcal{C}_{\text{selected}}[i]$ // Retrieve Decision Vector

    for $j \leftarrow 1$ **to** U **do**

      $\boldsymbol{R_2} \leftarrow \mathcal{C}_{\text{selected}}[j]$

      $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_1}, \boldsymbol{R_2})$ // combine responses using current Boolean func.

      compute (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combination to ROC plane

      push (tpr, fpr) onto $\boldsymbol{F}$

compute *ROCCH of all ROC points in* $\boldsymbol{F}$

$n_{ev} \leftarrow$ number of emerging vertices

$\mathcal{S} \leftarrow \{(D_1, t_i), (D_2, t_j), \ldots, (D_k, t_k), bf\}$ // set of selected decision thresholds from each detector and Boolean functions for emerging vertices.

store $\mathcal{S}$; **return** *ROCCH*

Chapter 3

Pruning

3.1 Introduction

Pruning is a very popular practice to decrease the cost and time of operation. By pruning the pool of thousand or even millions of detectors and reduce the number to some feasible number combination technique become practical. There are mostly two important challenges when it comes to pruning, first in which level should prune the detectors, and secondly, how this pruning should be applied, what are the criteria?

Generally speaking, combination and so on pruning can be done at four different levels: sensor level, feature level, matching score level, and decision level [40], as illustrated in Fig. 3.1

Pruning at matching score level is the most popular way of pruning, offering the best tradeoff between information content and ease of pruning [40]. In this thesis main focus of pruning etectors is at decision level, however there is some scenario that needed to prune at the matching score level. When there are thousands of detecros at matching score level, it doesn't make sense to convert them to few response vector and combine everything together, because first of all we will have too many redundant response vectors and makes the pruning process longer.

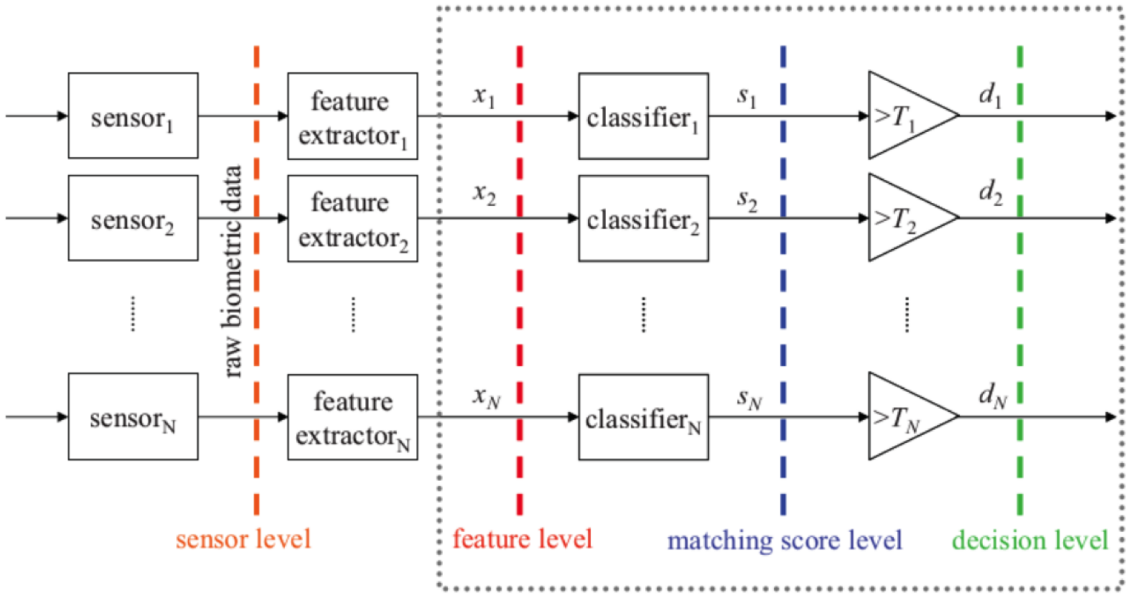


Figure 3.1: Different levels of combination or pruning: sensor level, feature level, matching score level, and decision level [40].

After seeing the different levels that pruning can be applied, the question is how to prune the pool of available detectors and make a basket of reasonable detectors. To have a reasonable basket of remaining detectors mostly two criteria should be considered.

Firstly, there should be some accurate detectors in the basket, with this we guarantee that the minimum accuracy we get after pruning and the combination is not worse than any of the detectors we had in the pool. Also by choosing and combining the accurate one we increase chances of getting better results than combining two random or naive detectors.

Secondly, we need to be sure that we have a diverse basket of detectors, if very similar detectors are selected in the basket we shouldn't expect to see a major improvement after combining. Other thing that helps to increase the diversity of the basket is to try to find and select detectors which complement the errors of already

selected detectors. Since we have already selected accurate detectors in the basket, it is a good practice to try to find the detectors that complement the errors of them.

To see the importance of diversity and how it can help improving the result of the combination, below example shows the step of selecting initial detectors and respective detectors compare to it that tries to complement its errors, and at the end give us the very promising result. As it's shown in figure 3.2 among all of the trained detecros one accurate one is selected, and in figure 3.3 it is shown that a detector is selected to increase diversity by completing first detector's errors.

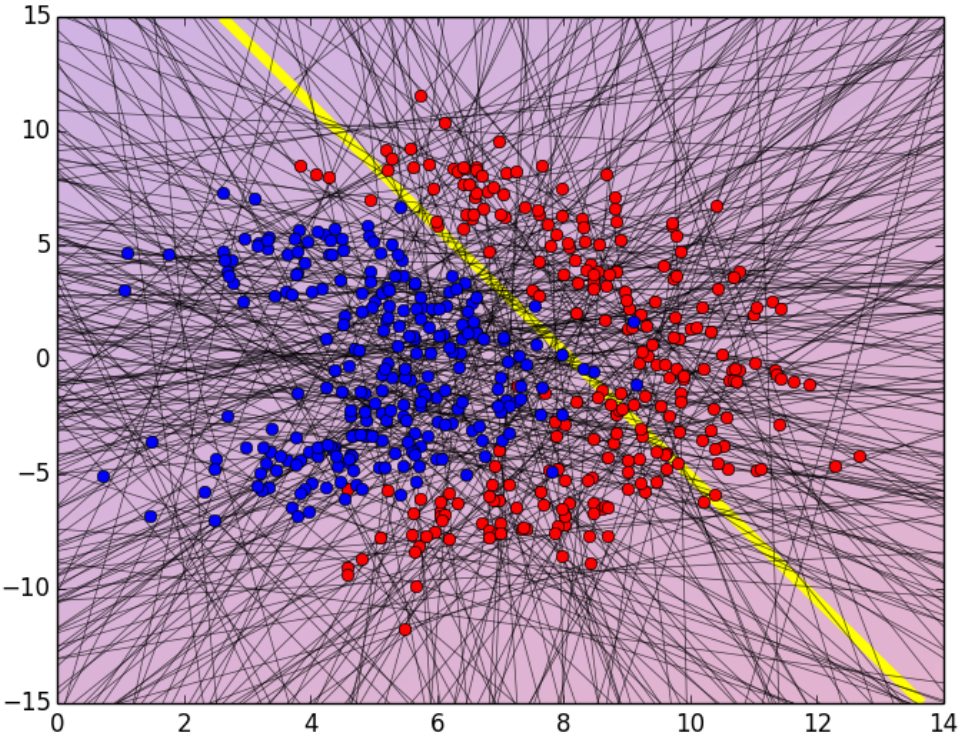


Figure 3.2: Selecting accurate detectors

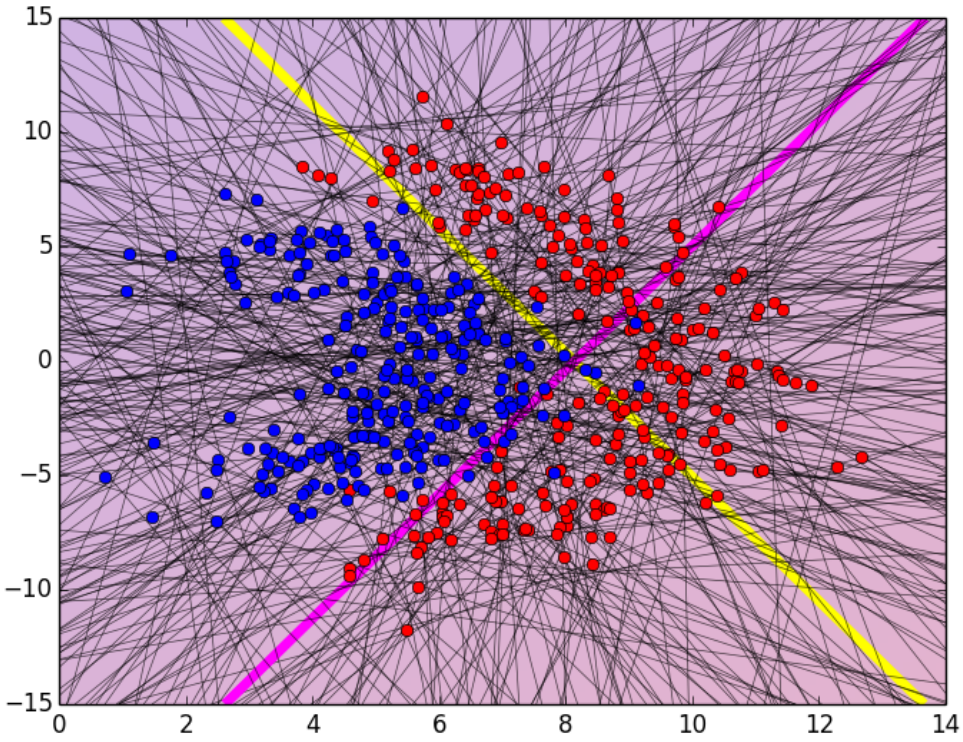


Figure 3.3: Selecting diverse detector compares to previously selected

3.2 Pruning Decision Level

Sometimes it is needed to prune earlier than match-score level, imagine there are too many trained models, as we had when we worked with ADFA data set, and since our technique work with response vector every one of these scores will convert to 100 new detectors, that is leading to having an even bigger pool of detectors. In order to prevent a large number of combinations and reduce chances of having redundant or very similar detectors it is a good idea to prune at score levels when there are lots of trained models.

As it is shown in fig. 3.4 1000 different HMM are trained on ADFA data set, with different initial matrix and different number of states, and as it is clear on the

figure; many of them are very similar regarding having different number of states or... . That forces us to run pruning at decision level to improve the performance.

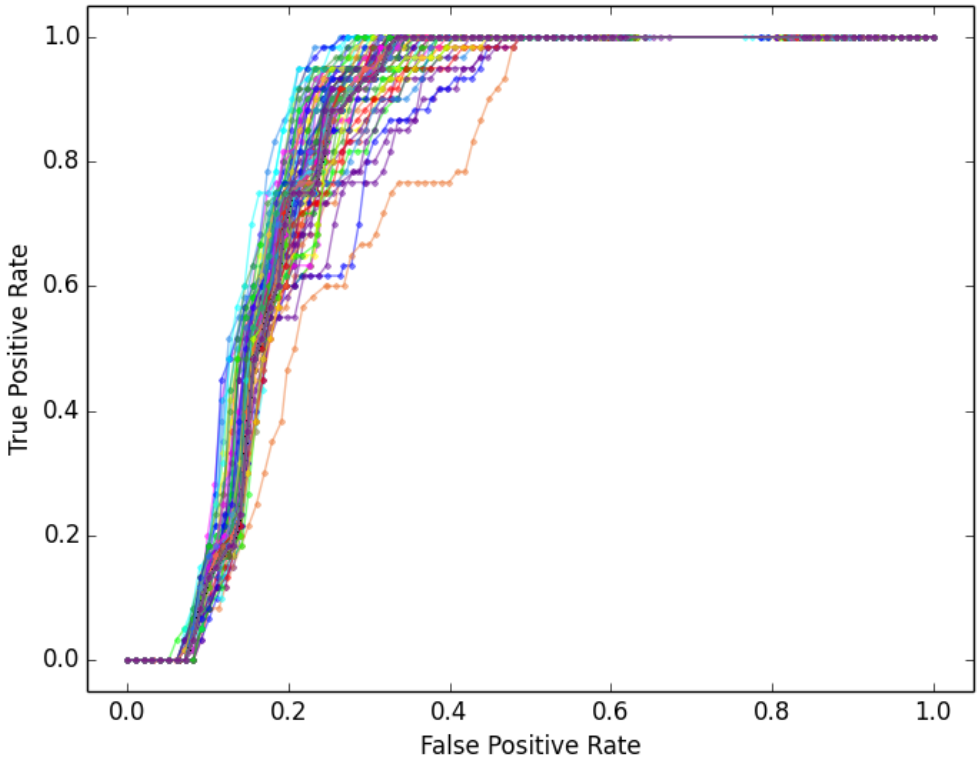


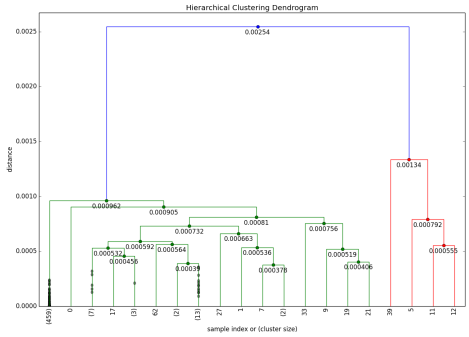
Figure 3.4: 1000 different HMM are trained on ADFA

For pruning at this level we are using hierarchical clustering, Hierarchical clustering organizes objects into a dendrogram whose branches are the desired clusters. Hierarchical clustering is a widely used method for detecting clusters in genomic data. Clusters are defined by cutting branches off the dendrogram. A common but inflexible method uses a constant height cutoff value; this method exhibits suboptimal performance on complicated dendrograms [41].

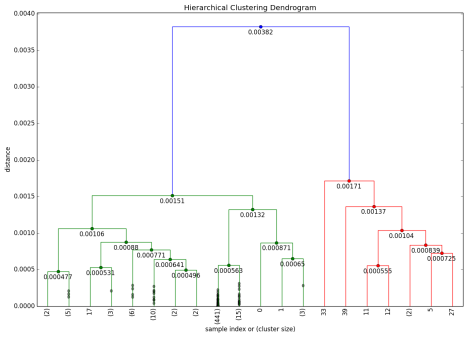
There are lots of different metrics (Euclidean, Cosine,...) and methods (Signle, Complete, Average, Ward, Median,...) that can be used to create the clusters. Since the purpose is to reduce the 1000 scores (that comes form training diffrenet model

on dataset) to 20 unique clusters and pick final 20 scores from each of the cluster, we can use any combination of above mentioned metrics and methods, and specific method or metric doesn't change the result significantly.

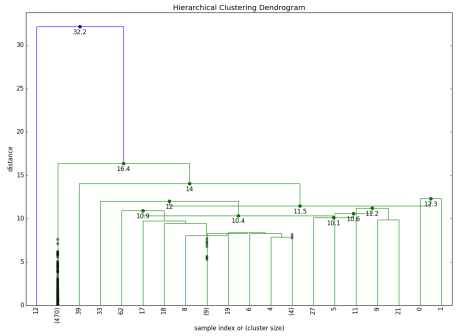
In fig. 3.5 there are 6 different dendrograms by setting cutoff to 20.



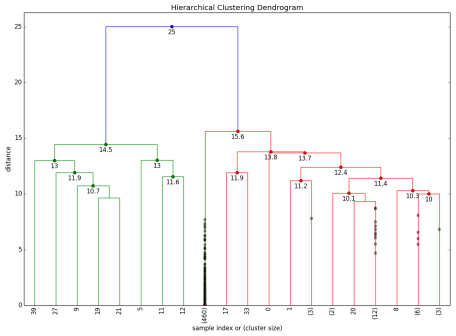
(a) Metric=Cosine, Method=Average



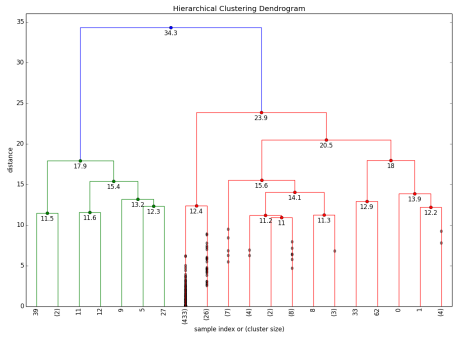
(b) Metric=Cosine, Method=Complete



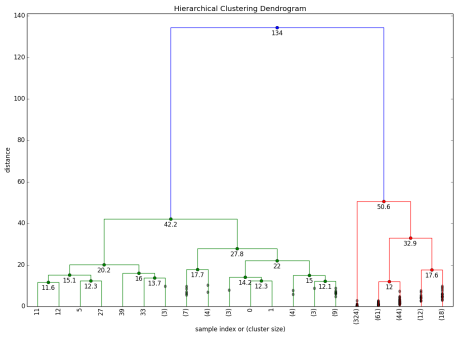
(c) Metric=Euclidean, Method=Centroid



(d) Metric=Euclidean, Method=Average



(e) Metric=Euclidean, Method=Complete



(f) Metric=Euclidean, Method=Ward

Figure 3.5: Dendrograms with cutoff as 20 with different metrics and methods on 1000 different model

In fig. 3.6 the final 20 selected scores that are chosen based on dendrogram with 20 cutooff are shown.

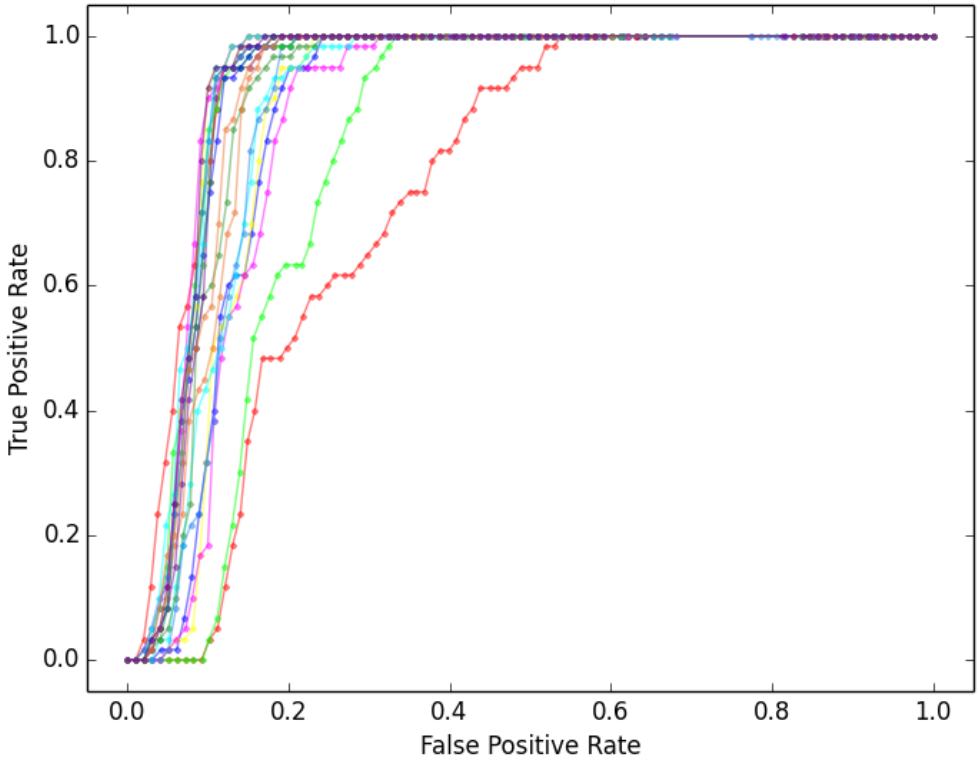


Figure 3.6: 20 selected scores based on hierarchical clustering

The scores that are shown in fig. 3.6 are our starting points for next level of pruning. In Section 3.3 we will see how we convert these scores to number of respons vectors and apply our technique to them to reduce the pool of detectors and combination cost.

3.3 Pruning Boolean Combination (PBC) Approach

In this section we describe our Pruned Boolean Combination (PBC) algorithm, which is based on two novel pruning techniques that select a subset of diverse and accurate detectors for combination, while discarding the remaining ones, this happened at decision level. PBC can avoid both the exponential explosion of combinations provided by the brute-force approach and the sequential combination of IBC. The

main difference is that the PBC algorithm proceeds by thresholding all available soft detectors into crisp ones. This step is, in fact, not essential for PBC because thresholding could be done outside the algorithm and the crisp detectors are directly input for pruning. In contrast with IBC, all available crisp detectors are input to one of the proposed pruning techniques, MinMax-Kappa or ROCCH-Kappa described in Section 3.3.3 and 3.3.4, to select the best subset of crisp detectors for Boolean combination and control its size U (i.e., the number of combined detectors). Without pruning brute-force pairwise combination of all available detectors may not be feasible for large number of detectors, $\mathcal{O}(N^2)$ (where N is number of available detectors), for further details see Section 3.4.

After seeing the main pruning idea in Section 3.3.3 and 3.3.4 we introduce a different distance correlation (measure of agreement between vecotrs or dependence coefficient) 3.3.2 rather than kappa to show that the results of PBC are consistent, and as long as a proper measure is used we can get promising result either using it with MinMax- or ROCCH-. In Section 3.3.5 we explained how we used the MinMax and ROCCH technique again and just changed the measure of agreement from kappa to Dcor or Dcov to prune the pool of detectors.

The boost in performance achieved with an ensemble of detectors is often attributed to the concept of diversity. While it is generally accepted that an ensemble should contain diverse models to improve the performance, there is no clear definition of diversity neither a consensus about the measure of diversity and its computation. In practice, several measures of diversity have been proposed to quantify the level of agreement or measure the dependency among the ensemble members [42].

3.3.1 Kappa Measure

Cohen's Kappa statistic (or simply Kappa hereafter) is one of the well-known and widely used measure of agreement between raters [20]. Kappa has lately gained some

popularity for ensemble combination, especially the Kappa-error diagrams which help visualizing individual accuracy and diversity in a two dimensional plot [43, 44]. Our pruning techniques, described in the next sections, are based on Kappa and inspired by the Kappa-error diagram only for visualization of Kappa against the false positives and true positives (as shown in Figures 3.7 and 3.8).

Table 3.1: Contingency table between two detector decisions

	D_2 correct	D_2 wrong
D_1 correct	a	b
D_1 wrong	c	d

Consider the contingency table of two detectors, D_1 and D_2, presented in Table 3.1, where, for instance, a is the number of examples on which both detectors agree. The sum of all element in Table 3.1 is equal to the size of validation set, $a+b+c+d=|\mathcal{V}|$. The Kappa (κ) measure between two detectors is therefore computed based on the element of the contingency table according to Equation 3.1.

$$\kappa = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{3.1}$$

Kappa takes on values between -1 and 1; lower values means a high level of disagreement or more diverse opinions, while higher values indicate a high level of agreement or similarity in responses between detectors. When both detectors provide the same vector of decisions (they agree on every example) then $\kappa = 1$. On the other hand, when $\kappa = 0$ the detectors are independent (any agreement is totally due to chance). Negative Kappa values can be interpreted as both detectors agrees less than what would be expected just by chance. More importantly, negative values account for negative correlations, which can be useful for combination, but this rarely occurs in practice [43].

3.3.2 Distance Covariance (dcov) and Distance Correlation (dcor) Measure

Distance correlation is a measure of statistical dependence between two vectors of arbitrary. The measure of dependence is zero if and only if the random variables are statistically independent. Distance correlation is a new measure of dependence between random vectors introduced by [45].

In order to understand the definition of distance covariance and distance correlation we should first start with distance covariance.

Assume we have two random vecors, the vecotrs are X_n and Y_n, $k = 1, 2, ..., n$. Fristly we should compute all piarwise distances:

$a_{j,k} = ||X_j - X_k||$

$b_{j,k} = ||Y_j - Y_k||$

By this, we computed the n by n distance matrices (aj, k) and (bj, k). After this we need to calculate all doubly centered distances:

$$A_{j,k} := a_{j,k} - \bar{a}_{j.} - \bar{a}_{.k} + \bar{a}_{..} \tag{3.2}$$

$$B_{j,k} := b_{j,k} - \bar{b}_{j.} - \bar{b}_{.k} + \bar{b}_{..} \tag{3.3}$$

In equation 3.2 $\bar{a}_{j.}$ is the j-th row mean, $\bar{a}_{.k}$ is the k-th column mean, and $\bar{a}_{..}$ is the grand mean of the distance matrix of the X sample. The exact same notation applies on b too. One thing to note is that in the matrices of centered distances $A_{j,k}$ and $B_{j,k}$ sum of all rows and all columns equal to zero.

By having all of these value we can calculate the distance covariance like this:

$$dCov_n^2(X, Y) := \frac{1}{n^2} \sum_{j,k=1}^{n} A_{j,k} B_{j,k} \tag{3.4}$$

Interesting point about distance coveriance is that we can define it with help of PearsonâĂŹs covariance.

$$dCov^2(X,Y) := cov(||X - X'||, ||Y - Y'||) - 2cov(||X - X'||, ||Y - Y''||) \tag{3.5}$$

After calculating the distance covariance we can use it to calculate distance correlation, The distance correlation of two random vectors is calculated by dividing their distance covariance by the product of their distance standard deviations.

$$dCor(X,Y) := \frac{dCov(X,Y)}{\sqrt{dVar(X)dVar(Y)}} \tag{3.6}$$

3.3.3 MinMax-Kappa Pruning

The proposed pruning technique starts by computing the Kappa values between each detector's decision vector and the true decision labels (or ground truth), and then sorting them in ascending order. Detectors with large Kappa values ($\kappa \approx \kappa_{max}$) are accurate and hence should be selected; however, they provide less diverse decisions among themselves. Therefore, the technique attempts to select complementary detectors by choosing those with Kappa values close to $\kappa \approx \kappa_{min}$. In practice, κ_{max} and κ_{min} values depend on the data and the detectors. Trivial detectors (providing always either positive or negative decisions) also reside at $\kappa_{min} \approx 0$. In such cases, these are filtered out before selected the complementary detectors. When detectors with negative Kappa values exist, it is always beneficial to select them since they provide detectors with negative correlations or complementary errors compared those with κ_{max}.

The MinMax-Kappa pruning technique takes two parameters, the total number of detectors and the ratio of detectors to be selected close to κ_{max} and κ_{min}. In our experiments, we experimented with different parameters sets and presented

the results for 50 selected detectors with a ration of 50% (which means half of the detectors are selected from the region with higher values of Kappa, while the remain half are chosen from the region with lower values). In fact, these are user-defined parameters that are constrained by the resources available during operations. Our experiments showed that the results of the pruning algorithms are not very sensitive to small changes in these parameters

Figure 3.7 shows an example of the selected detectors from our experiment, according to the MinMax-Kappa technique. The Kappa values (on the X-axis) for the same selected detectors are plotted against false positive (Figure 3.7a) and true positive (Figure 3.7b) rates. These points (large, blue) are the selected detectors, $\mathcal{C}_{\text{selected}}$ in Algorithm 2. All remaining detectors (small points) are pruned. For illustration, Figure 3.9a map the selected point to the ROC space, which could also be compared to our second pruning technique.

3.3.4 ROCCH-Kappa Pruning

This technique also tries to select accurate detectors and then find a set of detectors with complimentary errors. In contrast with MinMax-Kappa, this technique considers the detectors that are on the facet of the ROCCH in the ROC space. These detectors are selected since they are the most accurate available detectors that covers the whole range of fpr and tpr trade-off. The Kappa measure is then used to select diverse detectors with reference to those on ROCCH. The technique computes the Kappa value of each detector on the ROCCH and all the remaining detectors, sort the resulting values in ascending order. Detectors with larger values of Kappa are discarded, since they provide similar decisions (or errors) to those provided by the select detectors (on the ROCCH).

The most diverse detectors are those that provide negative or close to zero Kappa values ($\kappa \leq 0$), however are not trivial detectors (providing always either

positive or negative decisions). The only parameter of the ROCCH-Kappa is the number of detectors to be selected for each detector on the ROCCH. In practice, the number of detectors on the ROCCH is small ([3-20] detectors). Figure 3.8 shows an example of the selected detectors from our experiment, according to the ROCCH-Kappa technique, where the Kappa values (on the X-axis) for the selected detectors are plotted against false positive (Figure 3.8a) and true positive (Figure 3.8b) rates. Similarly, these selected points (large, blue) are the $\mathcal{C}_{\text{selected}}$ in Algorithm 2, while all remaining detectors (small points) can be pruned. Figure 3.9b map the detectors selected by ROCCH-Kappa to the ROC space, which could be compared to those selected by MinMax-Kappa in Figure 3.9a.

3.3.5 MinMax-, ROCCH- Pruning with dCor and dCov

These techniques keep using the logic we proposed in section 3.3.3 and section 3.3.4, the novelty behind them is to use newer and stronger distance correlation, which is described in section 3.3.2. With the help of these two new metric and integrating them with MinMax and ROCCH technique we can have more confidence in our PBC method that are working well and it is stable all the time and since you use it with a rational measure of agreement you can expect to get the reasonable results.

Here in both dCor and dCov when the result of comparing two different vectors is equal to 0 it means that they are surely independent of each other; and when the result is equal to 1 it means they are exactly similar to each other.

3.3.6 Expected Value of Boolean Combination

Expected Value of Boolean Combination (EVBC) Our other approach for pruning classifiers is using Expected value. To do that we calculate expected values of combinations of each classier with every other classifiers. This is an iterative method that would be applied on every classifier we have. First we select a classifier, classifier

X. To calculate expected value of a combination we need to choose another classifier such as classifier Y from our pool of classifiers. Since we have 10 different Boolean operations we could produce 10 different new classifiers by combining classifier X and Y. Here, we explain our formula for one of the Boolean operation but we could easily extend it for all the rest.

As we explained earlier we could identify each classifier by its TPR-FPR on ROC curve, therefore we want to calculate expected value of TPR-FPR of the new classifier. To calculate TPR and FPR of the combination (tpr_C, fpr_C) from Classifier X with (TPR_X, FPR_X) and classifier Y with (TPR_Y, FPR_Y) we use the equations below:

$$TPR_C = B1 \times TPR_X \times TPR_Y + B2 \times TPR_X \times (1 - TPR_Y) + B3 \times (1 - TPR_X) \times TPR_Y + B4 \times (1 - TPR_X) \times (1 - TPR_Y) \quad (3.7)$$

$$FPR_C = B1 \times FPR_X \times FPR_Y + B2 \times FPR_X \times (1 - FPR_Y) + B3 \times (1 - FPR_X) \times FPR_Y + B4 \times (1 - FPR_X) \times (1 - FPR_Y)) \quad (3.8)$$

Which B1, B2, B3, B4 is binary representation of our Boolean operation (And = '1000', OR = '1110', XOR = '0110', etc.) For example for XOR we would have:

$$TPR_C = 0 \times TPR_X \times TPR_Y + 1 \times TPR_X \times (1 - TPR_Y) + 1 \times (1 - TPR_X) \times TPR_Y + 0 \times (1 - TPR_X) \times (1 - TPR_Y) \quad (3.9)$$

$$FPR_C = 0 \times FPR_X \times FPR_Y + 1 \times FPR_X \times (1 - FPR_Y) + 1 \times (1 - FPR_X) \times FPR_Y + 0 \times (1 - FPR_X) \times (1 - FPR_Y)) \quad (3.10)$$

Now we have 3 classifiers X, Y and the combination of them using XOR operation (Classifier C) as it's shown in the picture below. Now we need to evaluate the goodness of classifier C compared to X and Y. Area Under the Curve (AUC) is a function, which can be used to evaluate the goodness. Based on the coordination of each classifier on ROC curve we can compute AUC of the classifier with the following formula:

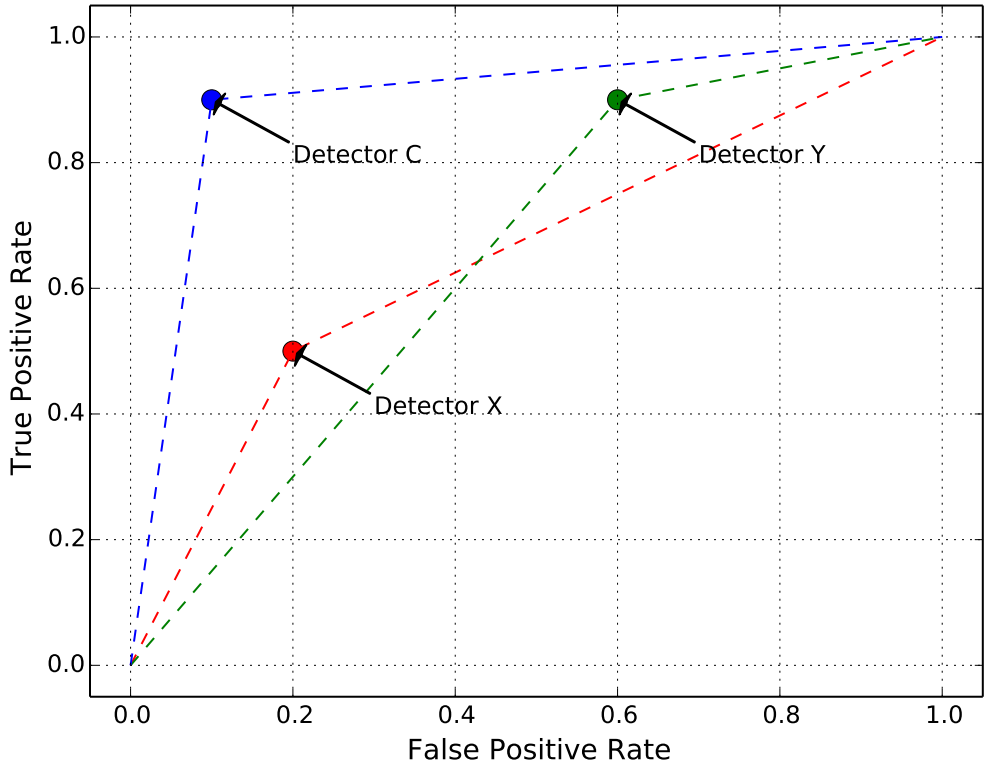


Figure 3.10: Combination of classifier X and Y lead to C

We compute AUC for classifier X, Y and C and keep classifier C only if its AUC is greater than AUC of both classifiers X and Y. Greater AUC means that our classifier is much closer to the ground truth (which will give us the TPR = 1 and FPR = 0). If AUC_C is greater than AUC_X and AUC_Y we keep the difference in AUC as the improvement of that classifier with the specific Boolean operation used. The variable that keeps the improvement for classifier X is IMP as defined below: $IMP_X = AUC_C$âĂŞ$Max(AUC_X, AUC_Y)$ We calculate IMP_X for all 10 Boolean

operations and add them to the previous value. At the end, IMP_X has the expected value of all of the combination possible between classifiers X and Y using 10 Boolean operations. We repeat these steps for classifier X in combination with the rest of classifiers, choosing one each time as a new classifier Y and update IMP_X. The final IMP_X value has the total expected value of the improvement with respect to all the combinations if we select classifier X. After we compute IMP_X for classifier X and the rest of classifiers we will update the value in our table in a TPR_X FPR_X position with IMP_X. We repeat all these steps for every classifier we have and update their IMP in the table and then we can create a table shown below with each cell representing the expected value of the classifier combinations getting improved. We select that classifier with the most improvement.

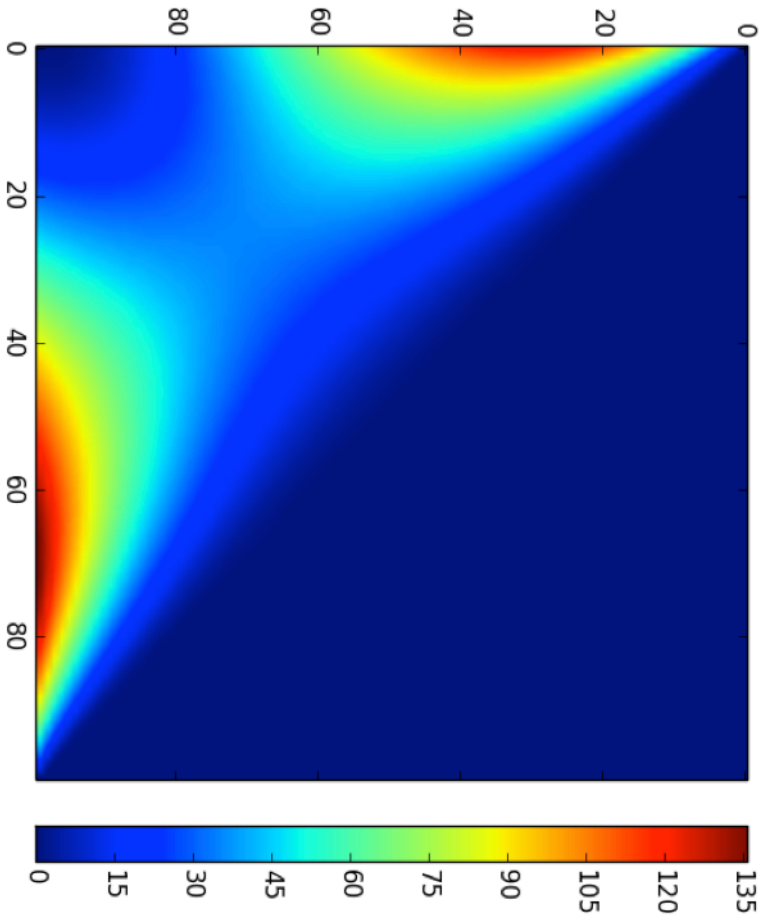


Figure 3.11: expected value of all classifiers in the ROC space

As shown in Fig. 3.11 the representation of that table if we use color plotting. The red region shows classifiers with higher IMP and the blue region shows classifier with low IMP. That means if we select our basket of classifiers from red region we would have a higher chance to get better classifiers after combination of those pruned classifiers. Our assumption for creating above table was that we have all the possible classifiers with 2 digits precision and after computing all possible combination for

all the classifiers we set the value of each cell below minor diameter to zero (all the classifiers below random line) since want to evaluate goodness of classifiers better than random guess. We can re-produce the same table by computing the expected value of combination just for the classifiers that already we have; by doing that we will get the below table (plot) and we can prune our classifier based on this new table.

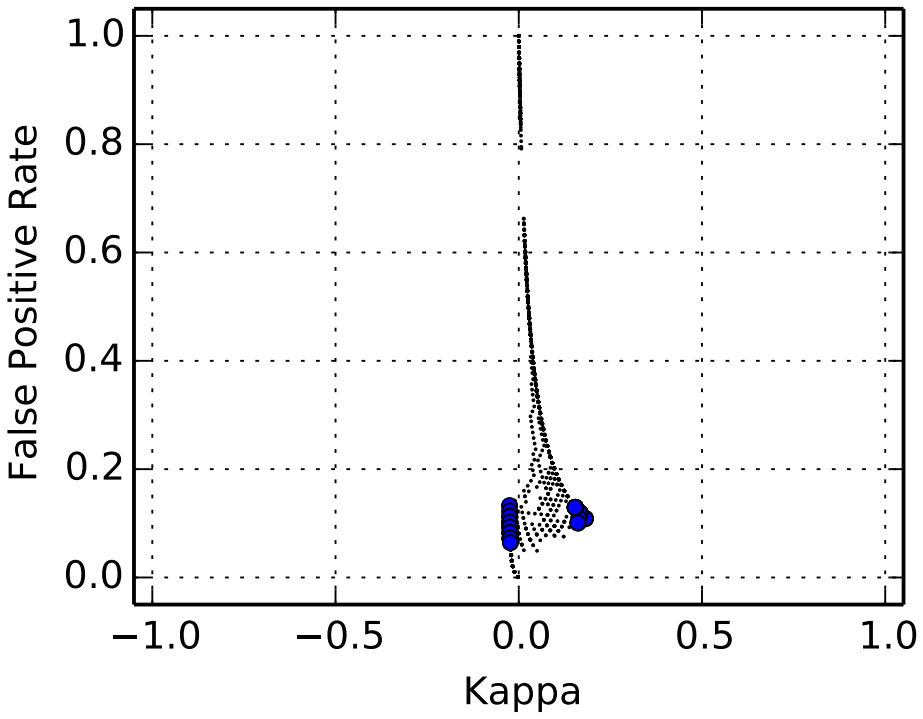
3.4 Complexity Analysis

Given K soft detectors, let n_i be the number of decision thresholds or crisp detectors produced by each of the soft detector D_i, $i = 1, \ldots, K$, on the validation set $\mathcal{V}$. Let $n = \sum_{i=1}^{K} n_i$ the total number of crisp detectors in the ensembles, and $n_{avg} = n/K$ the average number of crisp detectors produced by soft detectors.

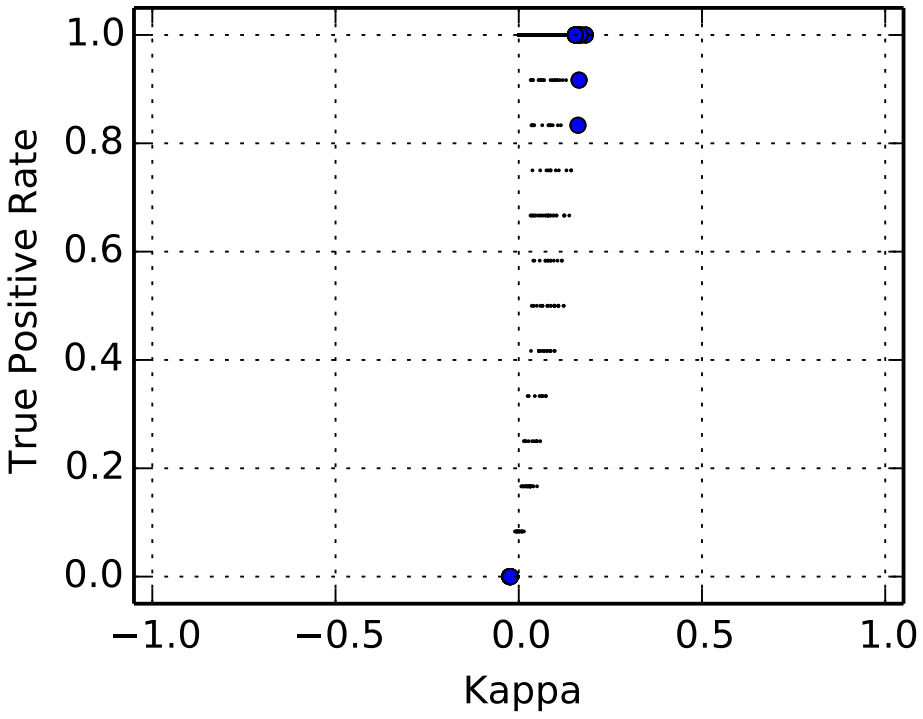
A brute-force search for optimal combination is infeasible in practice due to the doubly exponential combinations. In fact, for n crisp detectors there are 2^n possible outcomes that can be combined in 2^{2^n} ways, which makes the brute-force combination impractical even for small n values [19]. Even only pairwise combination of n crisp detectors, which requires $\mathcal{O}(n^2)$ Boolean operations, may not be feasible in practise for large n values. The sequential combination of the IBC algorithm reduces its worst-case time complexity to $\mathcal{O}(n_{avg}^2 + Kn_{avg})$ Boolean operations.

Both pruning techniques are capable of reducing the size of the selected subset of detectors (for Boolean combination) up to a user defined maximum number (U). The worst-case time complexity required by MinMax-Kappa technique to select U crisp detectors (and prune the rest) is $\mathcal{O}(n(\log n+1)+U^2)$. It requires about $n(\log n+1)$ operations for computing and sorting the Kappa values for all crisp detectors, and U^2 for the pairwise Boolean combinations of the U retained detectors. For ROCCH-Kappa technique however the worst-case time complexity is of the order $\mathcal{O}(n(\log n+$

$n_{ev}) + U^2)$, where n_{ev} is the number of emerging vertices which is typically around ten. The additional n_{ev} factor is due to the computation of Kappa is repeated n_{ev} times for each emerging point on the ROCCH. For numerical comparison, Table 4.2 shows the average time for each combination and pruning technique used in our experiments.

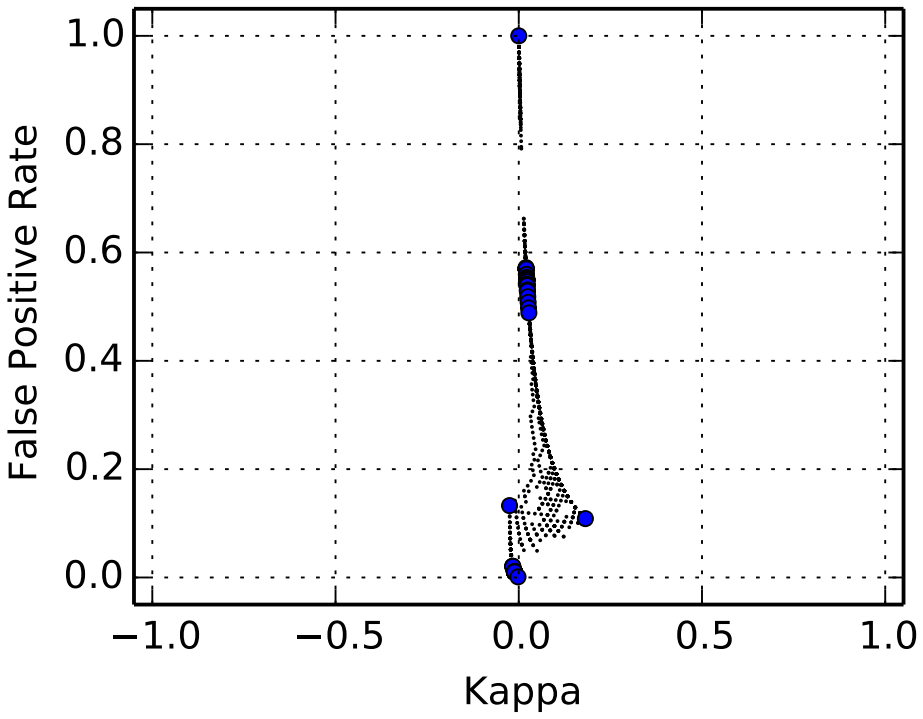


(a) κ-fpr diagram

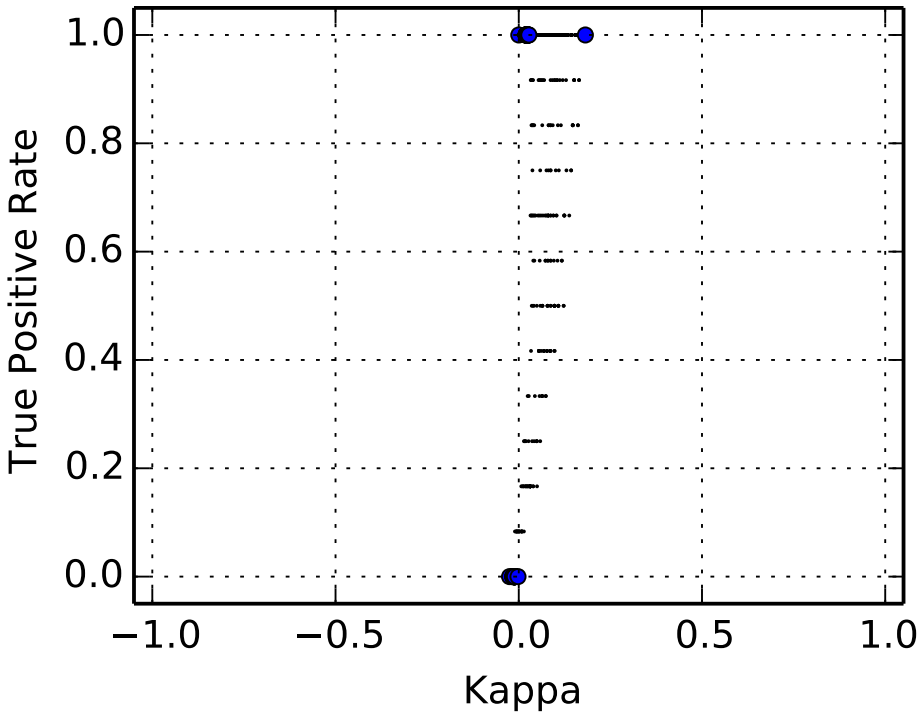


(b) κ-tpr diagram

Figure 3.7: Illustration of selected detectors (large blue circles) based on MinMax-Kappa pruning technique. All remaining detectors (small black dots) are pruned.

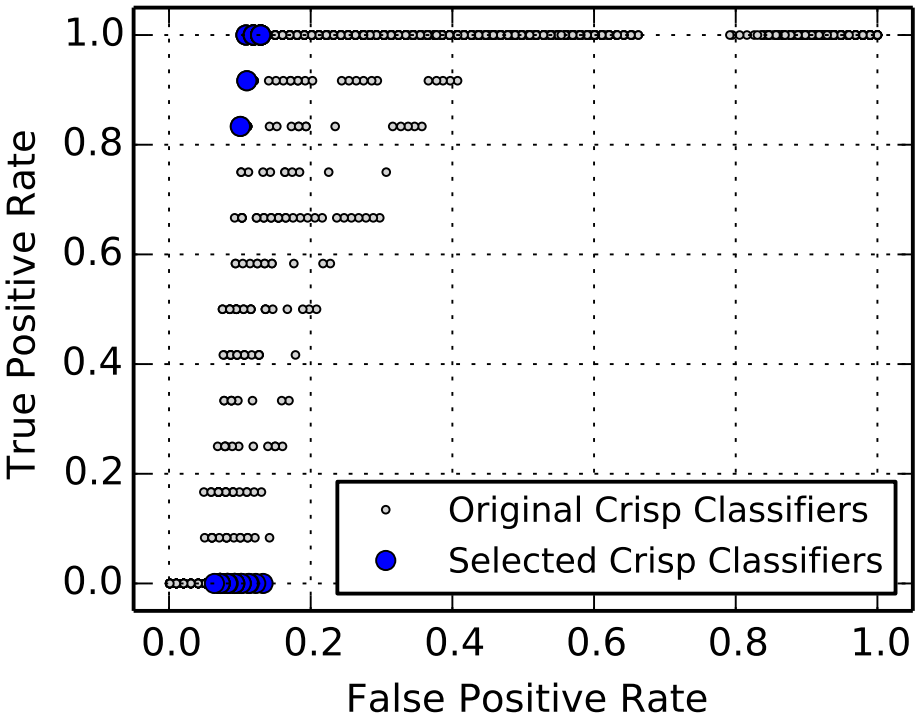


(a) κ-fpr diagram

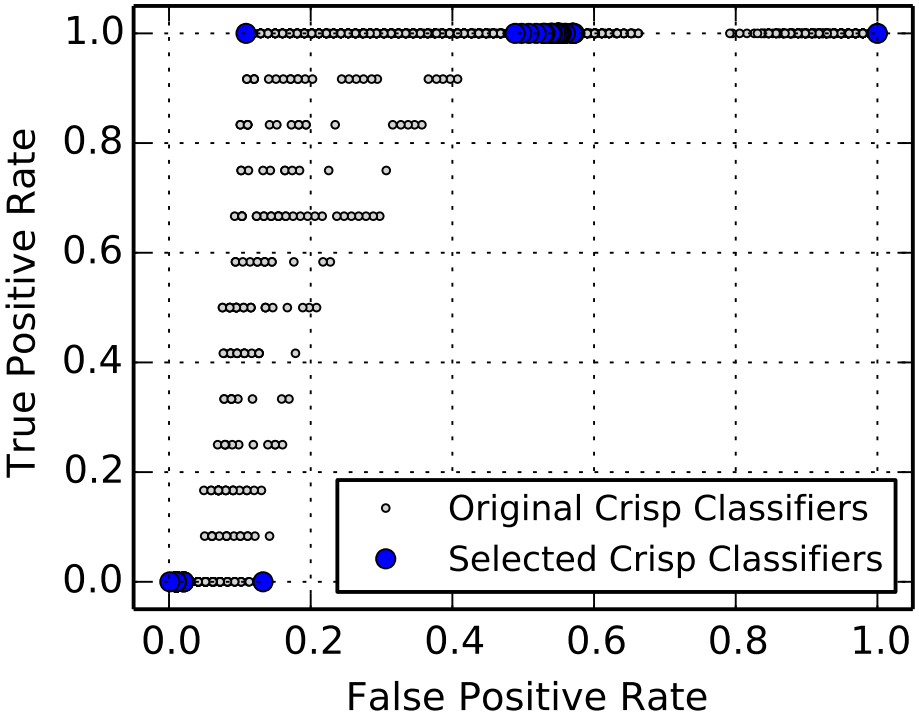


(b) κ-tpr diagram

Figure 3.8: Illustration of selected detectors (large blue circles) based on ROCCH-Kappa pruning technique. All remaining detectors (small black dots) are pruned.



(a) MinMax-Kappa pruning



(b) ROCCH-Kappa pruning

Figure 3.9: Illustration of the selected detectors for combination mapped onto the ROC space (large blue circles). All other detectors (small black dots) can be pruned.

Chapter 4

Experiment and Results

4.1 Experiments on System Call Dataset

We evaluate the accuracy and efficiency of our pruning techniques using a modern system call dataset, called ADFA Linux Dataset (ADFA-LD), which has been recently made publicly available on the website of the University of New South Wales [21]. The ADFA-LD dataset is generated by exploiting various security vulnerabilities in a Ubuntu operating system (OS) hosting a web server. The systems consists of a fully patched Ubuntu Linux 11.04 OS with an Apache 2.2.17 web server, PHP 5.3.5 server side scripting engine, TikiWiki 8.1 content management system, FTP server, MySQL 14.14 database management system and an SSH server. First they collect normal system call traces by letting users perform basic operations, such as web browsing and Latex document preparations under controlled situations. The anomalous system call traces are collected while the system is being under six types of attack vectors resulting in a total of 60 attacks. These attacks were launched by a certified penetration tester against the system and included web-based exploitation, simulated social engineering, poisoned executable, remotely triggered vulnerabilities, remote password brute-force attacks and system manipulation. The authors of the

ADFA-LD have organized the dataset into 833 normal traces for training the anomaly detectors, and 4373 normal traces and 60 anomalous traces for testing.

In our experiments we used the following experimental setup. First, we trained 20 HMMs with different numbers of states (i.e., $K = 20$ soft detectors), using the 833 normal traces provided for training in the ADFA-LD dataset. On average, the output of each HMM is thresholded into $n_{avg} = 100$ thresholds or crisp detectors, which provides $n = K \times n_{avg} = 20 \times 100 = 2000$ crisp detectors in the ensemble.

The objective is to choose the most accurate subset from this ensemble for combination while pruning the remaining detectors according to our MinMax-Kappa or ROCCH-Kappa technique. Then, the ROC curves and AUC results are compared to those of IBC and the pairwise Bruteforce Boolean Crisp combination (BBC2). BBC2 is another baseline, which is the PBC Algorithm 2, but without any pruning mechanism.

For evaluation of performance, a 5-fold cross-validation (5FCV) is applied to the test set comprising the 4373 normal and the 60 anomalous traces. Since the number of anomalous traces is relatively small with reference to the normal ones, we applied the cross validation to partition the normal and anomalous sets separately, such that we keep the same ratio (normal to anomalous) and guarantee that all folds include the anomalies. Accordingly, each fold contains 874 traces selected at random from the 4373 normal traces and 12 attacks traces selected at random from the 60 attack traces.

In contrast with the standard way of applying the 5FCV, we used one fold for computing the Boolean combination according to each of the combination technique, and the remaining four folds (i.e., 3498 normal traces and 48 attack traces) are used for evaluating and benchmarking the detection performance. This is because we wanted to test the boost in performance while using a small number of anomalies during combination. The reuslt are shown in section 4.2.

For second data set we used Canali data set [46], the data generated from 10 different machines. Our models , which are 4 HMMs and 5 SVMs, are trained on the normal data from: anubis-good + machine1 + ... + machine9, and the scores that we used are computed on the test data: machine10 that consist of (23 normal traces) + malware (5855 anomalous traces).

With this data set, in contrast with ADFA that we showed the boost in result with a small number of anomalies we want to show even when anomalies are the majority of the test set with PBC technique there is a boost in result. Here again we applied the PBC technqie with help of different distance correlation metric (Kappa, Distance Covariance, Distance Correlation) and compare the result with BBC2 and IBC. For Canali we used 5FCV approach as well. The reuslt are shown in section 4.3.

4.2 Results (ADFA)

Figure 4.1 shows the ROC curves and the AUC performance for both pruning techniques proposed in this paper (MinMax- and ROCCH-Kappa) compared to those of IBC and BBC2 (PBC with no pruning mechanism). And figure 4.2 shows the same ROC (the same fold) for both MinMax- and ROCCH- technique with different distance correlation as discussed in section 3.3.5.

These results are for one of the 5FCV experiments (the combinations are computed on one fold and evaluated on four) as described in Section 4.1. As shown in the figure the results are comparable both in term of AUC values and the shape of the ROC curves. This is also confirmed in Table 4.1, where the AUC values of each combination technique is averaged over the 5FCV experiments.

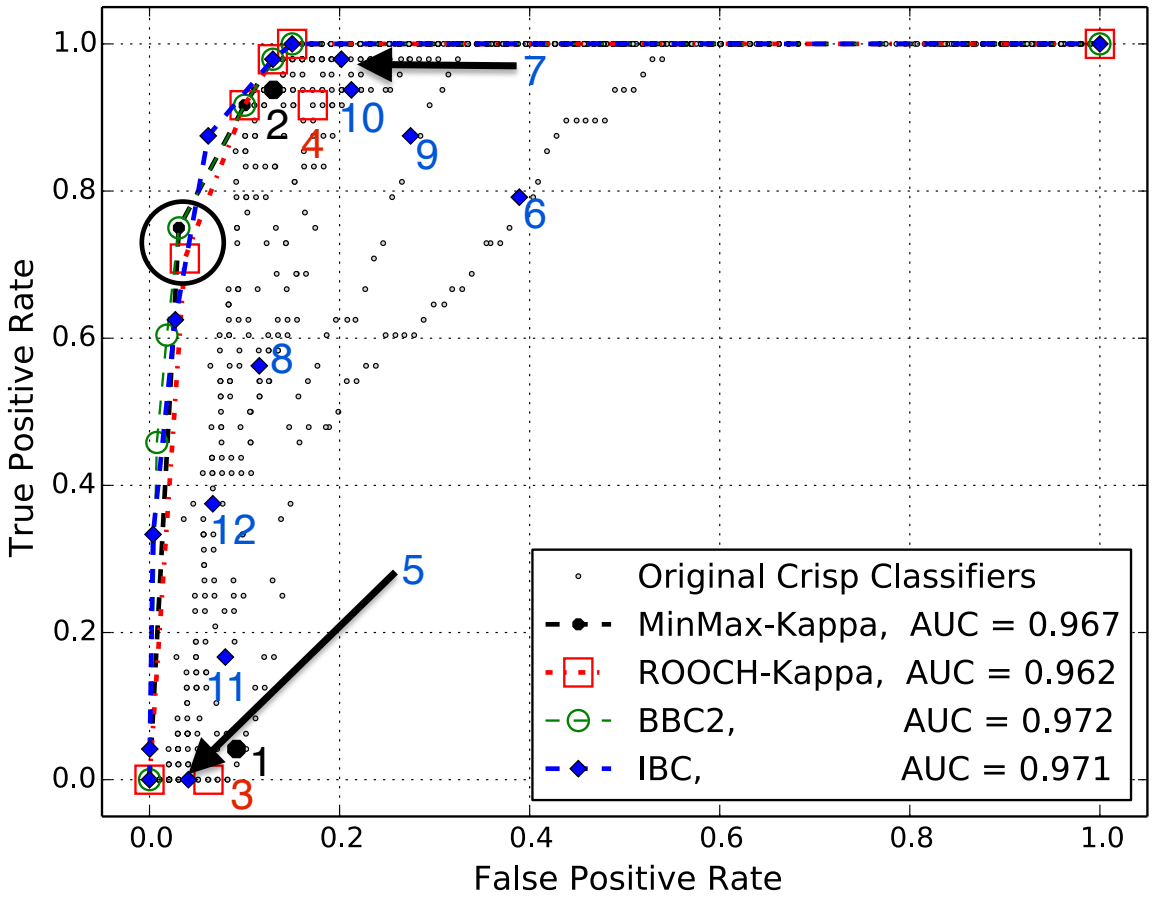


Figure 4.1: One of the ROC curves results of 5-folds cross-validation of Boolean combination on one fold and evaluated on four folds. The numbers represent the crisp detectors selected for combination (by each technique) to achieve about the same operational point as denoted by the large black circle

Table 4.2 shows the average over the 5FCV of the pruning time (for MinMax-Kappa, ROCCH-Kappa, MinMax-dCor, MinMax-dCov, ROCCH-dCor, ROCCH-dCov), combination time, and number of Boolean operations required by each technique to achieve the final ROCCH (as shown in figure 4.1 figure 4.2 for one fold).

We set the maximum number of selected detectors to $U = 50$ for all the above-mentioned techniques (this can be further optimized, but gave good trade-off between performance and time complexity). Therefore, the input to these pruning techniques

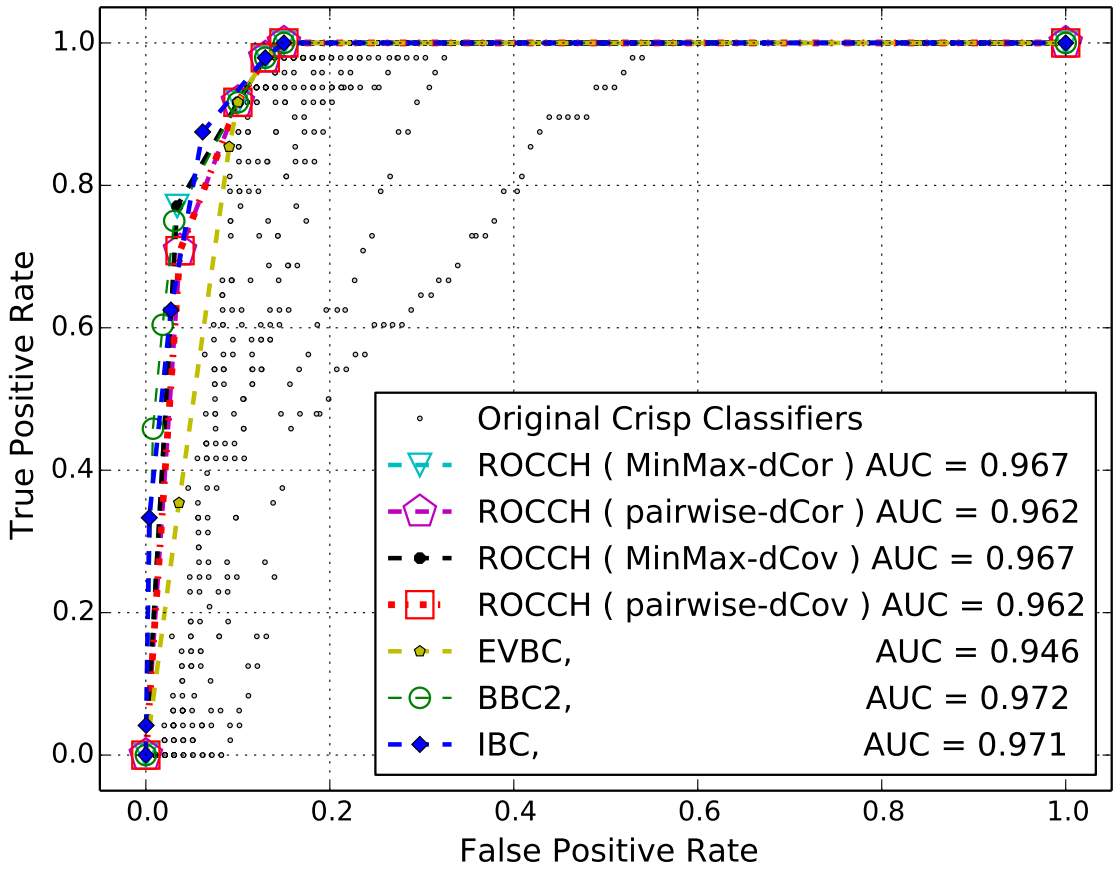


Figure 4.2: The same ROC curves results of 5-folds cross-validation of Boolean combination on one fold and evaluated on four folds that was used in fig. 4.1 with other distance correlation metrics

is $n = 2000$ crisp detectors, while the output is a subset of a maximum size of 50 detectors provided for Boolean combination (in PBC Algorithm). MinMax-Kappa took about 1.6 seconds in average to select the best subset, while ROCCH-Kappa took about ten times more due the n_{ev} factor, which is described in Section 3.4, to prune the ensemble of $n = 2000$ detectors. Furthermore, MinMax-Kappa is able to select a smaller number of detectors than $U = 50$ on average and computes the Kappa values once, which explains the reduction in combination time and number of Boolean operations compared to those of ROCCH-Kappa, as shown in Table 4.2. Thereby, although both techniques provide similar AUC performance, MinMax-Kappa is slightly

preferred due to its improved efficiency compared to ROCCH-Kappa.

Table 4.2 also shows that our MinMax-Kappa techniques was able to achieve the same AUC performance of BBC2 (the pairwise Boolean combination of the 2000 detectors), by selecting less than 50 detectors out of the 2000 ones. More interestingly, MinMax-Kappa achieved these results with an average time of three magnitudes lower than that of BBC2, and about 200 times fewer Boolean operations.

Compared to IBC, MinMax-Kappa requires, on average, slightly more combination time and a larger number of Boolean operations to achieve the same AUC performance. However, the number of selected detectors and Boolean functions required to realize each vertex on the ROCCH is on average five times more according to our experiments. For instance, to achieve the final operating points denoted with a large black circle on Figure 4.1, MinMax-Kappa uses two detectors and one Boolean function according to the following formula:

$$\neg D_1 \oplus D_2$$

Note that in Figure 4.1, we only shown the number of crisp detectors not to clutter the figures (i.e., 1 means D_1)
Similarly, ROCCH-Kappa uses the following detectors and Boolean function to achieve similar operating point:

$$\neg D_3 \oplus D_4$$

However to achieve similar point of operations, IBC requires eight detectors combined according to the following Boolean operations:

$$(((((((D_5 \oplus D_6) \wedge \neg D_{11}) \wedge D_7) \oplus D_8) \equiv D_9) \vee D_{10}) \wedge D_{12})$$

The average number of combined detectors on the final ROCCH over the 5FCV is about 10 detectors when using IBC compared to two detectors when using PBC with MinMax-Kappa pruning as shown in the last column of Table 4.2. This sequence of combination rules grows linearly with the number of soft detector K, which makes

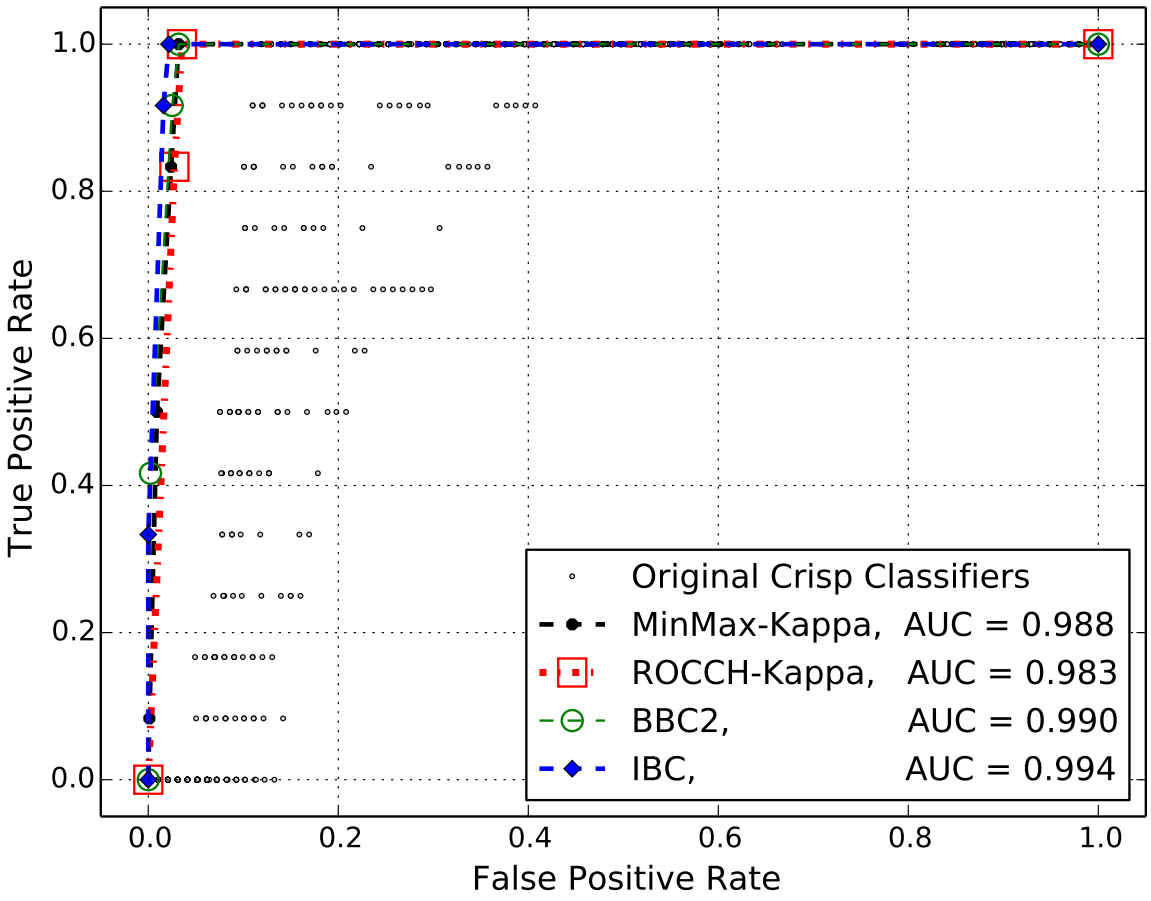


Figure 4.3: One of the ROC curves results of 5-folds cross-validation of Boolean combination on four folds and evaluated on one fold.

IBC results difficult to analyse and understand for large K values. In contrast to IBC, the combination of two detectors according to combination of PBC with MinMax-Kappa are insensitive to order in which detectors are input to the algorithm, which makes the search for the best subset of detectors easier. However, MinMax-Kappa requires an optimization of the maximum number of detectors U that trades off the complexity and the accuracy. Setting an ADS based on two HMMs into operations, requires less time and memory resources to provide the output probabilities of the input system call sequences. In addition, operating a small number of detectors becomes critical in application of anomaly detection mobile security, due to the constraint on power resources.

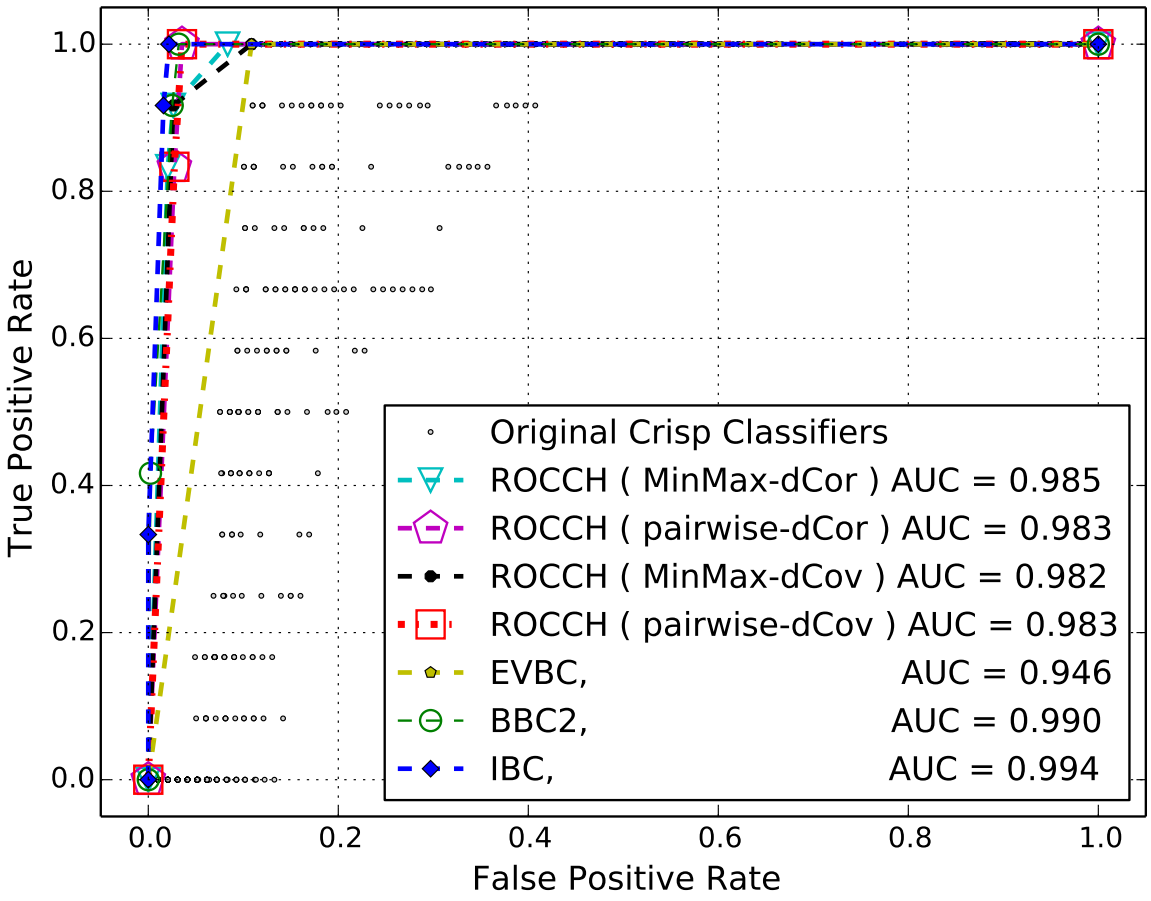


Figure 4.4: The same ROC curves results of 5-folds cross-validation of Boolean combination on four folds and evaluated on one fold that was used in fig. 4.3 with other distance correlation metrics.

We conducted an alternative case study to check the impact on ROC and AUC performance of all Boolean techniques, when the number of anomalies is increased during the design phase. Therefore, instead of using one fold (comprising 12 attack traces and 874 normal traces) for selecting the crisp detectors and the corresponding Boolean functions, as described in Section 4.1, we used 4 folds (48 attack traces and 3498 normal traces) and one fold for evaluation of performance.

Figure 4.3 and figure 4.4 show the ROC curves and the AUC performance for all techniques, PBC with MinMax-Kappa, ROCCH-Kappa, MinMax-dCor, MinMax-dCov, ROCCH-dCor, ROCCH-dCov, IBC and BBC2. Again, the presented results

are for one of the 5FCV experiments (but the combinations are computed on four folds and evaluated on one). As shown in the figure, all techniques provide comparable results; however, with a large improvement in detection accuracy over those presented in Figure 4.1. For instance, for detecting all attacks ($tpr = 100\%$) the false positive rate is now $fpr \approx 2\%$ compared to $fpr \approx 16\%$ in Figure 4.1. This boost in performance can be also seen in Table 4.3 in terms of average AUC values for each combination technique over the 5FCV experiments. The results of this experiments show, as expected, that when the system is provided with more normal or attack traces, the overall performance of all Boolean combination techniques improves. In such cases, there is no need to retrain the original detectors (HMMs in our case), which is a time consuming process, but the design phase of Boolean combination techniques must be repeated. This provides an advantage for IBC and PBC, since they are efficient in selecting the detectors for final operations. However, our PBC approach will always provide two detectors for each emerging point on the ROCCH, which is less costly to operate and easier to analyse in real-world setting.

4.3 Results (Canali)

Figure 4.5 shows the ROC curves and the AUC performance for both pruning techniques proposed in this paper (MinMax- and ROCCH-Kappa) compared to those of IBC and BBC2 (PBC with no pruning mechanism) on Canali dataset in order to see if the PBC technique is consistent regardless of dataset. And figure 4.6 shows the same ROC (the same fold) for both MinMax- and ROCCH- technique with different distance correlation as discussed in section 3.3.5.

These results are for one of the 5FCV experiments (the combinations are computed on one fold and evaluated on four) as described in Section 4.1. As shown in the figure the results are comparable both in term of AUC values and the shape of

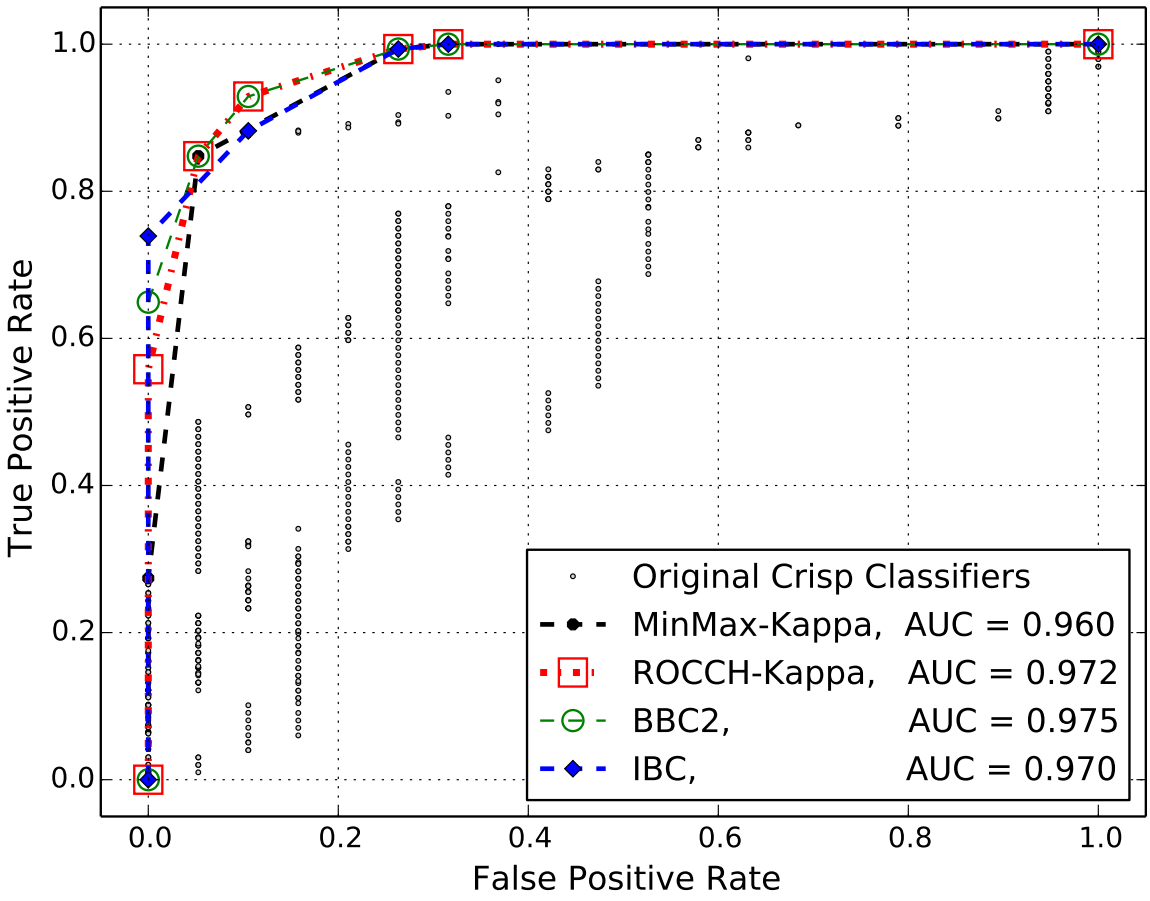


Figure 4.5: One of the ROC curves results of 5-folds cross-validation of Boolean combination on four folds and evaluated on one fold on Canali.

the ROC curves. This is also confirmed in Table 4.4, where the AUC values of each combination technique is averaged over the 5FCV experiments.

In this experminet like what we did for ADFA 4.2 we set the maximum number of selected detectors to $U = 50$ for all the above-mentioned techniques. Therefore, the input to these pruning techniques is $n = 900$ crisp detectors (4 different HMMS and 5 SVMs each thresholded 100 times), while the output is a subset of a maximum size of 50 detectors provided for Boolean combination (in PBC Algorithm).

Compared to ADFA's 4.2 result here we can see that ROCCH-Kappa generate even better result than IBC, however the difference is not significant yet it persuade us more to use PBC; Since by using PBC we are guaranteed that our final formula is

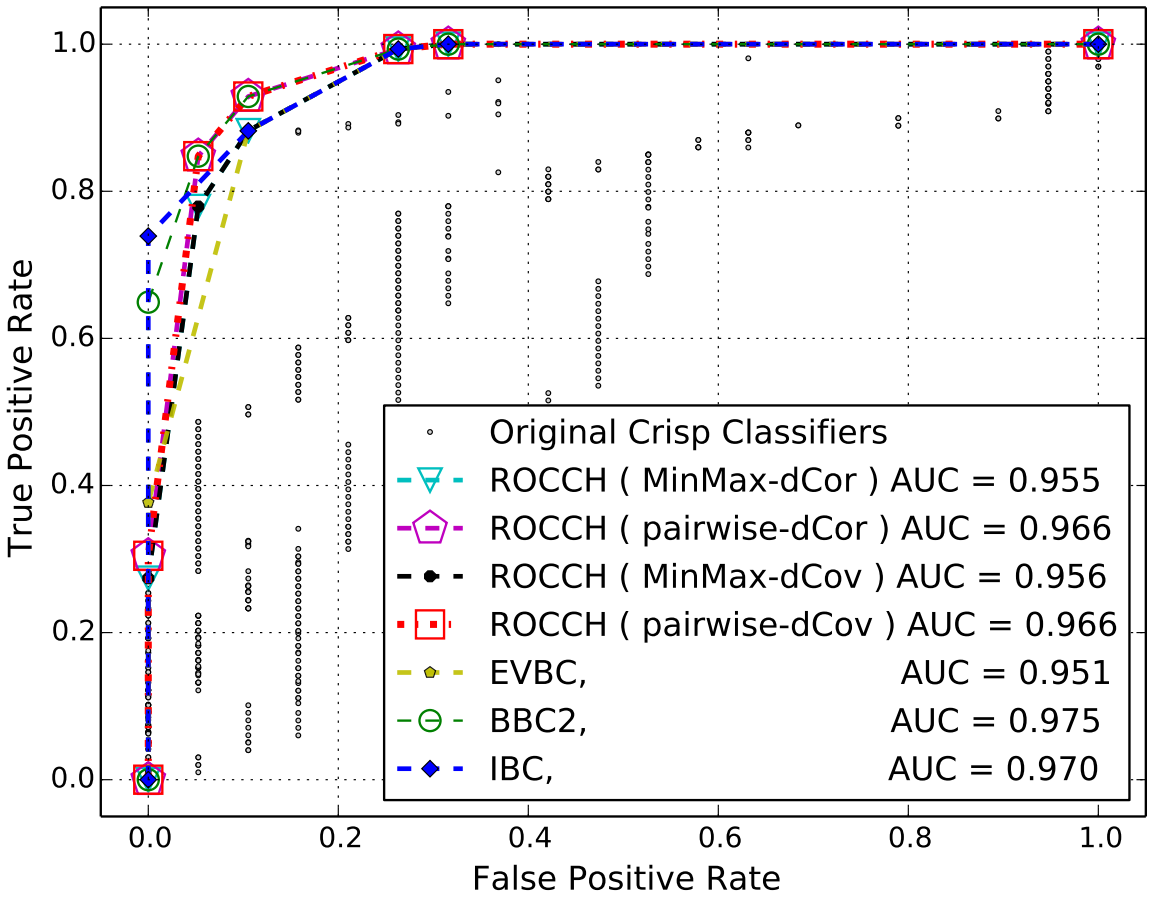


Figure 4.6: The same ROC curves results of 5-folds cross-validation of Boolean combination on one fold and evaluated on four folds that was used in fig. 4.5 with other distance correlation metrics

shorter and we have much fewer number of combinations that leads to less calculation time which saves CPU cycles (and get the result very much faster).

Running PBC as discussed 3.3 with complettly new data set like Canali (which is for Widnows) and getting promising result is a proof that PBC technique is applicable on various data sets as long as having a diverse detectors trained on the them.

The key for PBC to work well is to have as many diverse detectors as possible. It doesn't matter if it's very complex detectors or simple one, we should have diverse pool of detctors; then with help of PBC we prune the pool to create the basket of

detectors for combination. PBC with reasoble distance correlation tries to select few sample from each set of detectors such that it represnts the original pool.

4.4 Threats to Validity

A threat to internal validity exists in the implementation of the IBC, BBC2 and PBC algorithms as well as in conducting the experiments for anomaly detection. We have mitigated this threat by manually verifying the outputs.

We have conducted experiments using only one system call dataset derived from the Linux operating system, which consists a threat to external validity of this study. More experiments are therefore required to generalize the presented results to other datasets, operating systems and other software vulnerabilities.

Evasion attacks could also pose a threat to validity. For instance, *mimicry* attacks try to mimic the normal system behavior will go undetected with the ADSs that are based on individual system calls or their temporal order [47]. Mimicry attacks could be conducted by an attacker who is able to launch his attack without tempering the normal order of system calls by, for instance, replacing foreign system call sequences (which can be easily detected) with normal ones or by using system call arguments [47].

The manifestations of such mimicry attacks could be detected by including additional features, such as system call arguments [48], return values extracted from the call stack information [49], and the user identity [50]. An added advantage of multiple detector systems, combined without our PBC, is that they can combine different detectors trained on various features, such as system call arguments, return values and other information flow features to help mitigating such evasion attacks.

Table 4.1: Average AUC values and their standard deviations over the 5FCV for each techniques. Design on one fold and evaluated on four folds.

Method Name	Mean	Std
BBC2	0.97426	0.001
IBC	0.97276	0.001
MinMax-Kappa	0.97074	0.003
ROCCH-Kappa	0.97051	0.003
EVBC	0.94472	0
MinMax-dCor	0.9673	0.006
ROCCH-dCor	0.97067	0.005
MinMax-dCov	0.96515	0.004
ROCCH-dCov	0.97051	0.005

Table 4.2: Comparison of pruning and combination time (seconds) and number of Boolean operations required to achieve the final ROCCH during the design phase, and the number of selected detectors during the operational phase. All values are averaged over 5FCV.

	Design Phase			**Operations**
Method Name	Pruning Time	Combination Time	# Boolean operations	# Combined detectors
BBC2	N/A	16364	4,000,000	2
IBC	N/A	11	11,000	11
MinMax-Kappa	1.6	15	19,701	2
ROCCH-Kappa	11.8	38	37,701	2
EVBC	370	13	...	2
MinMax-dCor	74	16	19,701	2
ROCCH-dCor	190	30	37,701	2
MinMax-dCov	64	17	19,701	2
ROCCH-dCov	185	37	37,701	2

Table 4.3: Average AUC values and their standard deviations over the 5FCV for each techniques. Design on four folds and evaluated on one fold.

Method Name	Mean	Std
BBC2	0.98177	0.00636
IBC	0.98003	0.01127
MinMax-Kappa	0.97806	0.00640
ROCCH-Kappa	0.97578	0.00585
EVBC	0.94472	0
MinMax-dCor	0.9725	0.012
ROCCH-dCor	0.9752	0.005
MinMax-dCov	0.96268	0.011
ROCCH-dCov	0.9753	0.005

Table 4.4: Average AUC values and their standard deviations over the 5FCV for each techniques. Design on four folds and evaluated on one fold on Canali.

Method Name	Mean	Std
BBC2	0.97398	0.008
IBC	0.97141	0.006
MinMax-Kappa	0.95884	0.005
ROCCH-Kappa	0.9716	0.008
EVBC	0.95074	0.006
MinMax-dCor	0.95426	0.006
ROCCH-dCor	0.96451	0.006
MinMax-dCov	0.9616	0.011
ROCCH-dCov	0.96451	0.006

Chapter 5

Conclusion

In this paper, we proposed PBC, an efficient approach for selecting and combining anomaly detectors, which relies on two novel pruning techniques. During the design phase, the PBC is able to select a small subset of diverse and accurate detectors for Boolean combinations, while discarding the remaining ones. The pruning techniques we developed at the core of PBC rely on Kappa measure (MinMax-Kappa) and on the ROC convex hull (ROCCH-Kappa) to aggressively prune redundant and trivial detectors.

The results on ADFA-LD system call datasets show that PBC with both pruning techniques are capable of maintaining similar overall accuracy as measured by the ROC curves to that of IBC and BBC2. Therefore, our proposed PBC-based ADS is able to prune and combine large number of detectors without suffering from the exponential explosion in number of combinations provided with the pairwise brute-force Boolean combination techniques. This has been shown analytically (in the time complexity analysis) and confirmed in the experimental results.

During the operational phase, PBC with both pruning techniques always provides two crisp detectors for each combination, while IBC requires an average of 11 detectors to achieve the same operating point (in terms of true and false positive

rates). The proposed PBC approach is also general and can be applied to combine any soft or crisp detectors or two-class classifiers in a wide range of applications that requires combination of decisions.

Future work involves conducting more experiments using different real-world datasets. Another interesting direction is to investigate the potential improvement by re-combining the resulting combinations with the selected detectors, and explore other measures of diversity. We also intend to implement the new techniques in TotalADS [51], a tool we have developped to support multiple anomaly detectors. Finally, we need to investigate how we can reduce the size of traces to enable better scalability. Example of trace abstraction techniques are presented in [52, 53].

Bibliography

[1] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16 – 24, 2013.

[2] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, "A sense of self for unix processes," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pp. 120–128, May 1996.

[3] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pp. 133–145, 1999.

[4] S. Forrest, S. Hofmeyr, and A. Somayaji, "The evolution of system-call monitoring," in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pp. 418–430, Dec 2008.

[5] Y. Du, H. Wang, and Y. Pang, "A hidden markov models-based anomaly intrusion detection method," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 5, pp. 4348–4351, June 2004.

[6] B. Gao, H.-Y. Ma, and Y.-H. Yang, "Hmms (hidden markov models) based on anomaly intrusion detection method," in *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, vol. 1, pp. 381–385 vol.1, 2002.

[7] X. Hoang and J. Hu, "An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls," in *Networks, 2004. (ICON 2004). Proceedings. 12th IEEE International Conference on*, vol. 2, pp. 470–474 vol.2, Nov 2004.

[8] J. Hu, "Host-based anomaly intrusion detection," in *Handbook of Information and Communication Security*, pp. 235–255, 2010.

[9] W. Wang, X.-H. Guan, and X.-L. Zhang, "Modeling program behaviors by hidden markov models for intrusion detection," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 5, pp. 2830–2835, Aug 2004.

[10] X. Zhang, P. Fan, and Z. Zhu, "A new anomaly detection method based on hierarchical hmm," in *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pp. 249–252, Aug 2003.

[11] W. Khreich, E. Granger, R. Sabourin, and A. Miri, "Combining hidden markov models for improved anomaly detection," in *Communications, 2009. ICC '09. IEEE International Conference on*, pp. 1–6, June 2009.

[12] A. Sultana, A. Hamou-Lhadj, and M. Couture, "An improved hidden markov model for anomaly detection using frequent common patterns," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 1113–1117, June 2012.

[13] S. Murtaza, A. Sultana, A. Hamou-Lhadj, and M. Couture, "On the comparison of user space and kernel space traces in identification of software anomalies," in *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 127–136, March 2012.

[14] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 226–239, Mar 1998.

[15] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, pp. 1–15, 2000.

[16] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[17] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC Press, 2012.

[18] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "Boolean combination of classifiers in the roc space," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 4299–4303, Aug 2010.

[19] M. Barreno, A. Cardenas, and D. Tygar, "Optimal roc curve for a combination of classifiers.," *Advances in Neural Information Processing Systems (NIPS), MIT Press*, pp. 57–64, 2008.

[20] W. W. Cohen, "Fast effective rule induction," in *In Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115–123, Morgan Kaufmann, 1995.

[21] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pp. 4487–4492, April 2013.

[22] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, pp. 151–180, 1998.

[23] A. P. Kosoresow and S. A. Hofmeyr, "Intrusion detection via system call traces," *IEEE Software*, vol. 14, pp. 35–42, 1997.

[24] A. B. Somaayaji, *Operating System Stability and Security through Process Homeostasis*. Phd thesis, University of New Mexico, 2002.

[25] W. L. S. Kevin D. Mitnick, *The Art of Intrusion: The Real Stories Behind the Exploits of Hackers, Intruders and Deceivers*. Wiley, 2 2005.

[26] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection," in *Proceedings of the 1st Conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1*, ID'99, pp. 6–6, 1999.

[27] Y. Liao and V. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection1," *Computers & Security*, vol. 21, no. 5, pp. 439 – 448, 2002.

[28] S. Jha, K. Tan, and R. A. Maxion, "Markov chains, classifiers, and intrusion detection," in *Proceedings of the Computer Security Foundations Workshop*, pp. 206–219, 2001.

[29] C. Marceau, "Characterizing the behavior of a program using multiple-length n-grams," in *Proceedings of the 2000 Workshop on New Security Paradigms*, pp. 101–110, 2000.

[30] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pp. 14–23, Dec 2003.

[31] F. Gao, J. Sun, and Z. Wei, "The prediction role of hidden markov model in intrusion detection," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 2, pp. 893–896 vol.2, May 2003.

[32] Y.-S. Chen and Y.-M. Chen, "Combining incremental hidden markov model and adaboost algorithm for anomaly intrusion detection," in *Proceedings of the*

ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, pp. 3–9, 2009.

[33] P. Wang, L. Shi, B. Wang, Y. Wu, and Y. Liu, "Survey on hmm based anomaly intrusion detection using system calls," in *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pp. 102–105, Aug 2010.

[34] "A survey of techniques for incremental learning of {HMM} parameters," *Information Sciences*, vol. 197, pp. 105 – 130, 2012.

[35] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[36] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[37] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognition*, vol. 36, no. 1, pp. 229 – 243, 2003.

[38] K. Tan and R. Maxion, ""why 6?" defining the operational limits of stide, an anomaly-based intrusion detector," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pp. 188–201, 2002.

[39] T. Fawcett, "An introduction to {ROC} analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006.

[40] Q. Tao and R. Veldhuis, "Threshold-optimized decision-level fusion and its application to biometrics," *Pattern Recogn.*, vol. 42, pp. 823–836, May 2009.

[41] P. Langfelder, B. Zhang, and S. Horvath, "Defining clusters from a hierarchical cluster tree," *Bioinformatics*, vol. 24, pp. 719–720, Mar. 2008.

[42] L. Kuncheva, "That elusive diversity in classifier ensembles.," in *Lecture Notes in Computer Science*, vol. 2652 of *Lecture Notes in Computer Science*, pp. 1126–1138, Springer, 2003.

[43] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pp. 211–218, 1997.

[44] L. I. Kuncheva, "A bound on kappa-error diagrams for analysis of classifier ensembles.," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 3, pp. 494–501, 2013.

[45] G. J. Székely, M. L. Rizzo, *et al.*, "Brownian distance covariance," *The annals of applied statistics*, vol. 3, no. 4, pp. 1236–1265, 2009.

[46] D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "A quantitative study of accuracy in system call-based malware detection," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pp. 122–132, 2012.

[47] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pp. 255–264, 2002.

[48] S. Bhatkar, A. Chaturvedi, and R. Sekar, "Dataflow anomaly detection," in *Security and Privacy, 2006 IEEE Symposium on*, pp. 15 pp.–62, May 2006.

[49] H. Feng, O. Kolesnikov, P. Fogla, W. Lee, and W. Gong, "Anomaly detection using call stack information," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pp. 62–75, May 2003.

[50] U. Larson, D. Nilsson, E. Jonsson, and S. Lindskog, "Using system call information to reveal hidden attack manifestations," in *Security and Communication*

Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on, pp. 1–8, May 2009.

[51] S. Murtaza, A. Hamou-Lhadj, W. Khreich, and M. Couture, "Total ads: Automated software anomaly detection system," in *Source Code Analysis and Manipulation (SCAM), 2014 IEEE 14th International Working Conference on*, pp. 83–88, Sept 2014.

[52] A.Hamou-Lhadj, "The concept of trace summarization," in *Proceedings of the 1st International Workshop on Program Comprehension through Dynamic Analysis (PCODA'05)*, pp. 43–47, 2005.

[53] S. Murtaza, W. Khreich, A. Hamou-Lhadj, and M. Couture, "A host-based anomaly detection approach by representing system calls as states of kernel modules," in *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pp. 431–440, Nov 2013.