Mahkonen *et al. EURASIP Journal on Image and Video Processing* (2018) 2018:61
https://doi.org/10.1186/s13640-018-0303-9

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Cascade of Boolean detector combinations

Katariina Mahkonen*, Tuomas Virtanen and Joni Kämäräinen

Abstract
This paper considers a scenario when we have multiple pre-trained detectors for detecting an event and a small dataset for training a combined detection system. We build the combined detector as a Boolean function of thresholded detector scores and implement it as a binary classification cascade. The cascade structure is computationally efficient by providing the possibility to early termination. For the proposed Boolean combination function, the computational load of classification is reduced whenever the function becomes determinate before all the component detectors have been utilized. We also propose an algorithm, which selects all the needed thresholds for the component detectors within the proposed Boolean combination. We present results on two audio-visual datasets, which prove the efficiency of the proposed combination framework. We achieve state-of-the-art accuracy with substantially reduced computation time in laughter detection task, and our algorithm finds better thresholds for the component detectors within the Boolean combination than the other algorithms found in the literature.

Keywords: Binary classification, Classification cascade, Boolean combination

1 Introduction

Detection and binary classification are fundamental tasks in many intelligent computational systems. They may be considered as the same problem, where an input sample is to be determined into one of two groups, either one of two predefined classes, or as having some property or not. In the field of computer vision, face detection, pedestrian detection, and car detection are canonical examples that have received a lot of attention [1, 2]. Event detection from audio signal is of wide interest [3]. Detection tasks with multiple measurement modalities available are present, e.g., in biometric identity verification [4] and for medical decisions [5].

For detection of observation from a certain category, i.e., a class, many different types of detectors, trained with different data with different statistics—possibly even from different measurement modalities—are often available. Most of the detectors reported in the literature output a score, which denotes the likelihood of the existence of the quested target class, in the input data. A threshold value is then used to provide the classification "target" or "no target" for the input. Thus, a threshold value may be used to control the false negative-false positive trade-off, i.e., an operating point of the detector.

The different detectors may have very different performance, and the scores given by them are not fully correlated. Therefore, the combination of their outputs provides an opportunity to obtain a combined detector with performance superior to any of the components.

The cost of classification in terms of time and computational power, besides accuracy, is an important factor in many detection problems. Some of the detectors are very fast to execute while others are computationally heavy. An effective way to reduce the cost of classification is to use a sequential decision making process which asks for new resources only if needed for required accuracy.

We propose a new method for combining multiple sensitivity tunable detectors, i.e., detectors which output likelihood scores, to form a computationally efficient binary classification cascade. The component detectors are not restricted to be based on a single feature set, but may even operate on different measurement modalities. They have preferably been trained with different datasets to introduce uncorrelatedness in their output scores. For

*Correspondence: katariina.mahkonen@tut.fi
Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland

combining the available sensitivity tunable detectors, we propose to utilize a monotone Boolean function built using **AND** ($\wedge$) and **OR** ($\vee$) operators in disjunctive normal form (DNF). A Boolean function (BF) is said to be *monotone*, if changing the value of any of the input variables from 0 to 1 cannot decrease the value of the function from 1 to 0. For continuous data binarization, we use similar procedure as presented in [6]. Thus, a monotone BF on this data performs a monotone partition of the space of measurement values.

A BF lends itself naturally to sequential evaluation, which is an integral property of a decision process of a classification cascade. Also, by utilizing a BF of thresholded detector scores, we avoid inferring class probabilities from the scores, which would be error prone while having only a small dataset for combined system training. In the proposed **OR** of **AND**s function (BOA), each detector score is compared to multiple threshold levels, which allows formulating any monotonic decision boundary while making the classification decision in a computationally efficient way. The BOA cascade detector itself is trained to be sensitivity controllable as well.

The contributions of the paper are (1) a monotone Boolean **OR** of **AND**s (BOA) binary classification function to build a cascaded combination of multiple sensitivity tunable detectors, (2) an algorithm to train a BOA combination, and (3) utilizing a cascaded decision making process for audio-visual detection task.

For evaluating the proposed BOA detector cascade and the training algorithm to set its parameters, we use two audiovisual databases for two detection tasks, namely MAHNOB laughter dataset [7] for laughter detection task and CASA dataset [8] for video context change detection task. In the laughter detection task, we show that the accuracy of detection with a BOA cascade is superior to the other detection accuracies reported in the literature, while the computation time of detection is remarkably reduced compared to the other solutions. With three component detectors for the video context change detection, we show that the proposed BOA training algorithm outperforms alternative Boolean combination training algorithms found in the literature.

In the following section, we introduce the work related to Boolean detector combinations and algorithms for training Boolean combination parameters, as well as the work on cascaded detectors presented in the literature. The proposed Boolean OR of ANDs combination, and the algorithm to set its parameters are presented in Section 3. The experimental setup and the results obtained are presented in Section 4.

2 Related work

This paper proposes combining multiple tunable detectors robustly utilizing a monotone DNF-BF, named BOA, the evaluation of which is formulated as a computationally efficient classification cascade. Thus, we first review the literature on Boolean detector combinations and BFs in general. Then, we review the algorithms suitable for training a Boolean combination. Finally, we discuss the literature on classification cascades.

2.1 Boolean detector combinations

Using a Boolean conjunction or a Boolean disjunction for combining multiple detectors has been proposed in several studies, for example in [9–11]. Sensitivity tunable detector functions $f_m : \boldsymbol{x} \rightarrow \mathbb{R}$ for $m = 1 \ldots M$ are utilized within a combination. Each detector function $f_m(\boldsymbol{x})$ produces a score l_m, which denotes likelihood of the target appearing in the sample $\boldsymbol{x}$. The Boolean conjunction of M sensitivity tunable detectors is

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigwedge_{m=1}^{M} \left(f_m(\boldsymbol{x}) \geq \theta_m^{\mathrm{AND}} \right), \tag{1}$$

and the Boolean disjunction is

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{m=1}^{M} \left(f_m(\boldsymbol{x}) \geq \theta_m^{\mathrm{OR}} \right), \tag{2}$$

where $\boldsymbol{\theta}$ denotes all the thresholds θ_m^* used within the combination. All of the studies [9–11] report that either a conjunctive or a disjunctive Boolean combination of detectors do improve the detection accuracy over component detectors, provided that the thresholds $\theta_1{}^*, \theta_2{}^*, \ldots, \theta_M^*$ are set appropriately.

Mixtures of **AND** and **OR** operators within a Boolean combination have been investigated in [12]. Utilizing notation, where the detector function $f_m(\boldsymbol{x})$ identifiers m are listed in vectors z_q, $q = 1 \ldots Q$, each z_q containing M_q identifiers, this kind of Boolean **OR** of **AND**s combination is

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{q=1}^{Q} \left[\bigwedge_{i=1}^{M_q} \left(f_{z_q(i)}(\boldsymbol{x}) \geq \theta_{z_q(i)} \right) \right]. \tag{3}$$

As a big limitation of (3) proposed in [12], compared to the BOA combination that we suggest, is that only one threshold θ_m for each target likelihood score $f_m(\boldsymbol{x}) = l_m$ is allowed.

In addition to **AND** and **OR** operators, the Boolean negation, (**NOT**), and as a consequence also the exlusive-OR (**XOR**) are utilized in the detector combinations in [13–15]. The 2^{2^M} possible Boolean combinations that can be formed by M fixed, i.e., non-tunable, detectors utilizing **AND**, **OR**, **XOR**, and **NOT** operators are studied in [13].

Boolean detector combinations where each of the available target likelihood scores $f_m(\boldsymbol{x}) = l_m, \; m = 1 \ldots M$ may be cast to Boolean values using multiple thresholds $\theta_m^1, \; \theta_m^2, \ldots$ are first made use of in [14]. However, the space of the Boolean combinations generated by their algorithm is left unspecified.

A question of how to select the best performing Boolean combination for a certain problem, while having M sensitivity tunable detectors, has been posed in many of the abovementioned works. To select between conjunctive (1) and disjunctive (2) combinations, in [10, 16], it is suggested to investigate the class-conditional cross-correlations of detector scores and to consider whether the specificity or the sensitivity is more important. The conjunctive fusion rule (1), which emphasizes specificity, should be used if there is negative correlation between detector outputs for samples of the "non-target" class. If on the other hand the correlation of detector output scores for samples from the "target" class is weak, disjunctive fusion rule (2) emphasizing sensitivity should be used. All in all, a Boolean combination is able to exploit negative or weak correlation of detector scores.

To select among the combinations of the form (3), rules of thumb have been drawn in [12] according to average cross-correlations between the scores from the used detectors. It is shown for three detectors with Gaussian score distributions and identical pairwise cross-correlations that either a conjunctive combination (1), a disjunctive combination (2), or a type (3) combination

$$\begin{aligned} B(\boldsymbol{x};\boldsymbol{\theta}) = & \left[\left(f_1(\boldsymbol{x}) \geq \theta_1^{\text{vote}}\right) \wedge \left(f_2(\boldsymbol{x}) \geq \theta_2^{\text{vote}}\right) \right] \quad \vee \\ & \left[\left(f_1(\boldsymbol{x}) \geq \theta_1^{\text{vote}}\right) \wedge \left(f_3(\boldsymbol{x}) \geq \theta_3^{\text{vote}}\right) \right] \quad \vee \\ & \left[\left(f_2(\boldsymbol{x}) \geq \theta_2^{\text{vote}}\right) \wedge \left(f_3(\boldsymbol{x}) \geq \theta_3^{\text{vote}}\right) \right], \end{aligned}$$

which stands for a majority vote rule, is the best and outperforms the component detectors. The one of those to be selected depends on class conditional cross-correlations between detectors.

The Iterative Boolean Combination (IBC) method in [14] is specifically designed to find the best possible Boolean combination, not restricted to monotone functions, for a certain sensitivity level of a combination. The search space of BFs is nevertheless restricted to avoid an unfeasibly large number of possibilities. The IBC method results in variety of Boolean detector compounds, but the study does not provide analysis of the form of the generated compounds nor characteristics of their resulting decision boundaries.

Theory of constructing BFs of unrestricted form, specifically in DNF as well as in CNF (conjunctive normal form), has been studied in depth, e.g., in [17]. BFs for classification have been studied vastly under terms logical analysis and inductive inference. Logical Analysis of Data (LAD) [18, 19] is a combinatorics- and optimization-based data analysis method first introduced in [20]. LAD methodology focuses on finding DNF-BF-type representations for classes.

The term inductive inference is used in many early texts concerning topics of machine learning, many of those discussing Boolean decision-making, e.g., [21, 22]. Using data binarization, e.g., as proposed in [6], all these results concerning BFs may be utilized in conjunction with continuous valued data.

Any BF may be converted into a binary decision tree, while the structure of the tree is generally not unique. In case of the proposed BOA DNF-BF, the corresponding deterministic read-once binary tree has depth $\geq \lceil \log_2(N_\theta + 1) \rceil$. In maximally deep node arrangement, the tree becomes a single branch tree with depth equal to the number N_θ of thresholds used in the BOA function. However, this kind of binary tree representation does not highlight the computational advantages of BOA cascade that we are interested in.

2.2 Algorithms for training a Boolean combination

The parameter $\boldsymbol{\theta}$ of a Boolean combination function $B(\boldsymbol{x};\boldsymbol{\theta})$ denotes all the thresholds $\theta_m^n \in \theta$ for $m = 1 \ldots M$, $n = 1 \ldots N_m$ used in the combination. For a Boolean combination $B(\boldsymbol{x};\boldsymbol{\theta})$ to perform well, suitable values for the set $\boldsymbol{\theta}$ of thresholds must be found. Most of the studies rely on training data-based exhaustive search for selecting the threshold values for $\boldsymbol{\theta}$, e.g., [10, 12, 13]. The computational load of this approach is $\mathcal{O}\left(T^{|\boldsymbol{\theta}|}\right)$, where $|\boldsymbol{\theta}| = \sum_{m=1}^{M} N_m$ is the total number of thresholds in $\boldsymbol{\theta}$ and T is the number of threshold values tested for each detector. The exhaustive search becomes computationally prohibitive if there are more than a couple of threshold values to find. Thus, more efficient algorithms are needed. In addition to algorithms readily proposed for tunable classification function training, we shortly review algorithms which have been developed for BF training for one operating point and their extensions to incremental learning.

A fast method for finding sets $\boldsymbol{\theta}$ of thresholds for different sensitivity levels of a Boolean combination $B(\boldsymbol{x};\boldsymbol{\theta})$ is presented in [10]. The method exploits the receiver operating characteristic (ROC) curve of each utilized detector $d_m(\boldsymbol{x},\theta) = \left(f_m(\boldsymbol{x}) \geq \theta\right)$. The ROC curve shows the true positive rate (*tpr*) against the false positive rate (*fpr*) at every operating point, defined by the threshold θ, of the detector. When $\theta = -\infty$, the classification by $d(\boldsymbol{x},\theta)$ results in $tpr = 100\%$ and $fpr = 100\%$. On the other hand, when $\theta = +\infty$, then $tpr = fpr = 0$. The method selects the thresholds for the Boolean combination iteratively by fusing two BF components—individual detectors or partial BFs—at a time according to their ROC curves. Formulas for ROC curves of a conjunctive and disjunctive combination of detectors d_A and d_B, $d_A \neq d_B$, are provided as

$$tpr_\wedge\left(fpr_\wedge\right) = \max_{fpr_A \cdot fpr_B = fpr_\wedge} \Big(tpr_A(fpr_A) \cdot tpr_B(fpr_B)\Big) \quad (4)$$

and

$$tpr_\vee\left(fpr_\vee\right) = \max_{fpr_A + fpr_B - fpr_A \cdot fpr_B = fpr_\vee} \Big(tpr_A(fpr_A) + tpr_B(fpr_B) - tpr_A(fpr_A) \cdot tpr_B(fpr_B)\Big), \quad (5)$$

where $tpr_*(fpr_*)$ denotes the true positive rate of a detector $d_*(\boldsymbol{x};\theta)$ at an operating point θ where its false positive rate is fpr_*.

The efficiency of the method is based on an assumption that the classifications made by different detectors are independent. Unfortunately, this often does not hold in practice. If the same measurement set or the same set of features are used for multiple detectors, or if multiple thresholds are to be found for a certain target likelihood l_m within a Boolean combination, dependencies between classifications are very likely. We compare our algorithm to this Boolean algebra of ROC curves in the Section 4 and use an implementation for BOA training shown in Appendix 1.

Another algorithm that does not assume independence of the used detectors was proposed in [11]. It suggests training the combination iteratively by finding thresholds for two detectors or partial combinations at a time, similarly to the Boolean algebra of ROC curves presented above. In this approach, the search of the best thresholds for a Boolean combination is done via exhaustive search over all the possible threshold settings for the two systems to be merged. In the ROC space, with all the possible threshold settings, a Boolean combination produces a constellation of performance points. The left top edge of this constellation, consisting of the operating points of superior performance, was introduced by [23] as the convex hull of the ROC constellation. In the algorithm of [11], before each new component fusion, the set of possible threshold values for the newly built partial combination is pruned to constitute of only the thresholds corresponding the performance points at the convex hull of this ROC constellation. The algorithm is originally designed for pure conjunctive (1) or disjunctive (2) Boolean combinations, but we have implemented it to deal with a BOA as described in the Appendix 1, and we use it for comparison to our algorithm.

In the literature concerning BFs, there are many algorithms, which are designed to find a BF which perfectly classifies the training data $\left\{X^0, X^1\right\}$ in $\{0,1\}^{N_{\text{attr}}}$. Finding the simplest possible BF to explain some data is an NP complete optimization problem with $2^{2^{N_{\text{attr}}}}$ possible solutions. Some of the algorithms are designed assuming monotonicity of data, the assumption which diminishes the number of possible solutions remarkably [24]. The number of possible BFs is further reduced in the case of continuous data which is binarized as in [6]. In this case, the data with $M \ll N_{\text{attr}}$ continuous attributes actually resides in the M-dimensional manifold of the N_{attr} dimensional space of binarized data. However, the number of possible BFs is still exponential. A few of the approaches target finding a BFn with imperfect classification performance, which usually is the desirable learning result with imperfect data.

Because of NP completeness of finding the best BF to explain some data, most of the algorithms in the literature operate in iterative manner using some greedy heuristics. An A^q algorithm [25] and LAD [20]-based methods construct a DNF-BF via iteratively searching for good conjunctions, each of which covers a part of positive training samples, to be combined disjunctively. On the contrary, OCAT-RA1 -algorithm [26], based on idea of one-clause-at-a-time (OCAT) [27], builds a CNF-BF via iterative selection of disjunctions. In case of continuous data binarized as in [6], algorithms developed for decision tree learning, e.g., ID3 [28], C4.5 [29], CART [30] are also suitable for DNF-BF building.

The A^q algorithm and LAD-based methods are to find two DNF-BFs which provide perfect classification of the training data. One function is to be used for detection of the positive class, and the other one for detecting the negative class. The covers, i.e., subspaces for which BF = *true*, of these DNF-BFs are disjoint, leaving part of the input space uncovered by either function. The algorithms use different heuristic criteria when searching for suitable conjunctions, i.e. *complexes* in terms of A^q.

For A^q algorithm the user may choose the criterion, one possible choice beings the number of positive samples covered by the *complex*, that is, conjunction. For LAD methodology, different criteria for optimality of conjunctions, called *patterns* in LAD, are discussed in [31]. Selectivity criterion favors minterms based on data, and evidential criterion favors patterns covering as many data samples as possible. Algorithms for constructing patterns according to these different criteria are given in [18, 32, 33].

Algorithms for BF inference allowing imperfect classification, which is generally associated with better generalization of data with outliers, are for example AQ15 algorithm [34], which is based on A^q, and OCAT-RA1 algorithm proposed in [26]. A procedure for pruning an overfit DNF-BF representation for better generalization is provided within AQ15 algorithm. It is based on counts of samples covered by each conjunction individually and together with other conjuntions. The conjunctions which are small in these numbers are the ones to be pruned. OCAT-RA1 constructs each disjunction of a CNF-BF by iteratively selecting attributes for it based on their rank of $N_{\mathbf{tp}}(a)/N_{\mathbf{fp}}(a)$, where $N_{\mathbf{tp}}(a)$ ($N_{\mathbf{fp}}(a)$) is the number of positive (negative) training samples, which have attribute $a = 1$. New attributes are selected until all the positive samples are covered by their disjunction.

The binary tree building algorithms, which iteratively build the tree by starting from the root node and performing a new split at every iteration, implicitly facilitate different level generalizations of data and generate a decision function of DNF-BF form. The splitting criterion for selecting attributes for new nodes in ID3, C4.5, and C5.0 is gain in information entropy. ID3 is applicable with binarized data, while C4.5 and C5.0 can handle continuous data by implicitly performing the binarization by usage of thresholds. The CART algorithm uses either Gini impurity or Twoing criterion to decide about the attributes used in nodes of the tree.

Incremental learning algorithms enable updating a classification function when new data becomes available. Some of the algorithms keep all the data available for future updates, while some algorithms discard the original data and perform the update based on new data only. Incremental algorithms, which utilize all the original training data aside of some new data, for updating a BF are for example GEM [35] and IOCAT [36].

Both of the algorithms assume a DNF-BF, and their update procedures consist of two phases. At the first phase, if some of the new negative samples are misclassified by the original DNF-BF, the faulty conjunctions are located and specialized to not to cover those new samples. Both of the algorithms perform this step by replacing each faulty conjunction by new conjunctions which are trained using data inside the cover of the original conjunction. GEM utilizes A^q algorithm and IOCAT utilizes OCAT-RA1 algorithm for this re-training. At the second phase of BF update, the DNF-BF is updated in terms of the uncovered new positive samples. GEM generalizes the existing conjunctions to cover the new positives using A^q.

In IOCAT, for each uncovered new positive sample, a conjunction, i.e., *clause* in terms of IOCAT, to be generalized is selected based on ratio $N_{\mathbf{tp}}(\text{clause})/N_{\text{attr}}(\text{clause})$ of the number of positive samples covered by the clause $N_{\mathbf{tp}}(\text{clause})$ and the number of attributes in the clause $N_{\text{attr}}(\text{clause})$. The selected conjunction is then retrained with non-incremental OCAT-RA1 algorithm using all the negative samples, the new positive sample and the positive samples within the space covered by the selected conjunction.

2.3 Cascade processing for reduced computational load of classification

The goal in cascaded processing for detection is in reducing the computational cost of classification. The idea is to evaluate the input in stages, such that at each cascade stage new information about the input is acquired and then either the classification is released or the next cascade stage is entered for new information. Decision cascades have been investigated mostly in the field of machine vision starting from [37, 38]. Face detection and pedestrian detection are the most common application areas where decision cascades have been used, e.g., in [39–42]. Decision cascades have been utilized in other fields, e.g., in [43] for cancer survival prediction and in [44] for web search.

In the task of object detection from images, the heavily imbalanced class distribution, as most of the search windows of different sizes and positions do not contain the target object, offers great possibilities to make “non-target” classification with minor examination. Object detection cascades are designed such that gradually more and more features are extracted for increased classification certainty. A class estimate is released as soon as the classification certainty is high enough. If this is the case before all the obtainable features or measurements have been extracted, computational savings appear.

The first generation object detection cascades, used for example in [38], are able to make early classification to the “non-target” class only, as illustrated in Fig. 1 (left). To classify the input into the “target” class, the input must pass all tests $\left(f_s(\boldsymbol{x}) \geq \theta_s\right)$ of the cascade stages $s = 1 \ldots S$. This kind of one-sided cascade performs a conjunctive Boolean combination function

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigwedge_{s=1}^{S} \left(f_s\left(\boldsymbol{x}\right) \geq \theta_s\right).$$

The solution $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{true}$ denotes classification to the “target” class, and $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{false}$ denotes classification to the “non-target” class.

The second generation object detection cascades introduced in [45] and used also in [46] are able to make the early classification to both the classes, as illustrated in Fig. 1 (right). They utilize two thresholds on the target likelihood score $f_s(\boldsymbol{x}) = l_s$ at each cascade stage $s = 1 \ldots S-1$. One threshold, θ_s^{reject}, is used for early rejection, i.e., early classification to “non-target” class, if $\left(f_s(\boldsymbol{x}) < \theta_s^{\text{reject}}\right) = \textit{true}$. Another threshold, θ_s^{accept}, is used for early detection if $\left(f_s(\boldsymbol{x}) \geq \theta_s^{\text{accept}}\right) = \textit{true}$. This means that at each stage, either the classification is released, or the next stage is entered in case that $\theta_s^{\text{reject}} \leq l_s < \theta_s^{\text{accept}}$. At the last cascade stage, the classification is enforced by $\theta_S = \theta_S^{\text{reject}} = \theta_S^{\text{accept}}$. This kind of symmetrical cascade corresponds to a BF

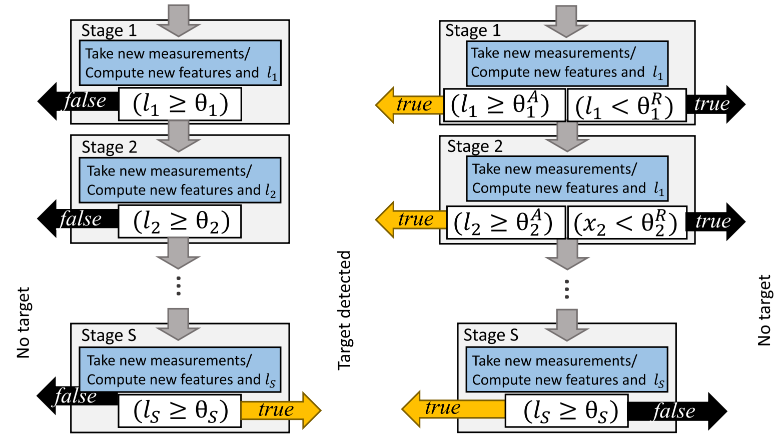

Stage 1
Take new measurements/ Compute new features and l_1
false
$(l_1 \geq \theta_1)$
Stage 2
Take new measurements/ Compute new features and l_2
false
$(l_2 \geq \theta_2)$
Stage S
Take new measurements/ Compute new features and l_S
false
$(l_S \geq \theta_S)$
true
No target
Target detected
Stage 1
Take new measurements/ Compute new features and l_1
true
$(l_1 \geq \theta_1^A)$
$(l_1 < \theta_1^R)$
true
Stage 2
Take new measurements/ Compute new features and l_1
true
$(l_2 \geq \theta_2^A)$
$(x_2 < \theta_2^R)$
true
Stage S
Take new measurements/ Compute new features and l_S
true
$(l_S \geq \theta_S)$
false
No target

Fig. 1 Two types of binary classification cascades. Typical object detection cascades utilized in computer vision. Left: an asymmetrical type cascade for classifying efficiently the "non-target" windows. Right: a symmetrical detection cascade which is capable of early classification to both classes

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{s=1}^{S} \bigwedge_{m=1}^{s-1} \left(f_m(\boldsymbol{x}) \geq \theta_m^{\text{reject}} \right) \wedge \left(f_s(\boldsymbol{x}) \geq \theta_s^{\text{accept}} \right), \tag{6}$$

whose output $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{true}$ denotes the classification to the "target" class and $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{false}$ denotes classification to the "non-target" class.

A cascade may be seen as a one branch decision tree, if the notion of tree is broadened from the traditional definition that a node makes a decision based on only one input attribute. In a "cascade-tree," a node function may utilize multiple input attributes, and the function may partition the corresponding input space freely to assign inputs to any of the leaves, i.e., classes, or down the branch to the next level node (stage of the cascade). In a cascade, the order of attribute acquisition is fixed in contrast to input-dependent order of attribute usage with a traditional decision tree.

For training, a detection cascade for computer vision applications, where the detectors to be utilized are designed having close to infinite pool of image features, e.g., Haar, HoG, an efficient cascade structure is guaranteed by concurrent design of detector functions f_s, $s = 1 \ldots S$, their thresholds θ_s^* and the cascade length S as proposed in [40, 47]. For a cascade with fixed length S, a method for concurrent learning of object detectors and their operating points is proposed in [39]. The methods proposed in the literature for finding operating points for pre-trained detectors within a detection cascade mostly assume strong correlation among detector scores. This is the case in [48], where an object detection cascade is designed using cumulative classifier scores, as well as in [45, 46], where the proposed algorithms are based on the assumption that the detector scores are highly positively correlated. If the detector scores are negatively or not correlated, those cascade training strategies turn unsuitable.

3 Methods

For combining multiple detector functions $f_m(\boldsymbol{x}) = l_m$, $m = 1 \ldots M$, which output likelihood scores $l_1, l_2, \ldots, l_M$ for the same target class, we propose to use a a BF. The proposed combination function utilizes Boolean **AND** ($\wedge$) and **OR** ($\vee$) operators and it is defined in disjunctive normal form. The proposed Boolean **OR** of **AND**s function (BOA) B yields a Boolean output $\mathrm{B} : \boldsymbol{x} \rightarrow \{\textit{false, true}\}$. The BOA output $\mathrm{B}(\boldsymbol{x}) = \textit{true}$ denotes input $\boldsymbol{x}$ classification to the "target" class and the BOA output $\mathrm{B}(\boldsymbol{x}) = \textit{false}$, i.e., $\neg\mathrm{B}(\boldsymbol{x}) = \textit{true}$, denotes classification to the "non-target" class.

Generally, a BF—possibly infinite—over a combination of thresholded detector scores is capable of producing any binary partition of the input space $\boldsymbol{x}$ or the space of target likelihood scores $(l_1, l_2, \ldots, l_M)$. Due to exclusion of the Boolean **NOT** rule, a BOA combination restricts the space of different partitions such that the spaces $\left\{ (l_1, l_2, \ldots, l_M) \mid \mathrm{B}(\boldsymbol{x}) = \textit{false} \right\}$ and $\{ (l_1, l_2, \ldots, l_M) \mid \mathrm{B}(\boldsymbol{x}) = \textit{true} \}$ are simply connected and the decision boundary is monotonic. This is illustrated in the example of Fig. 2, where the data points indicate laughter likelihoods from videos of MAHNOB laughter dataset [7], which is used in our evaluations.

We build a BOA combination of detector functions $f_m(\boldsymbol{x}) = l_m$, $m = 1 \ldots M$ using Boolean **OR** ($\vee$) and **AND** ($\wedge$) operators as

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{q=1}^{Q} \bigvee_{n=1}^{N_q} \left[\bigwedge_{i=1}^{M_q} \left(f_{z_q(i)}(\boldsymbol{x}) \geq \theta_{z_q(i)}^{q,n} \right) \right], \tag{7}$$

where in each vector $z_q \in \{1 \ldots M\}^{M_q}$ there are M_q detector identifiers $m \in \{1 \ldots M\}$ for BOA construction. Each

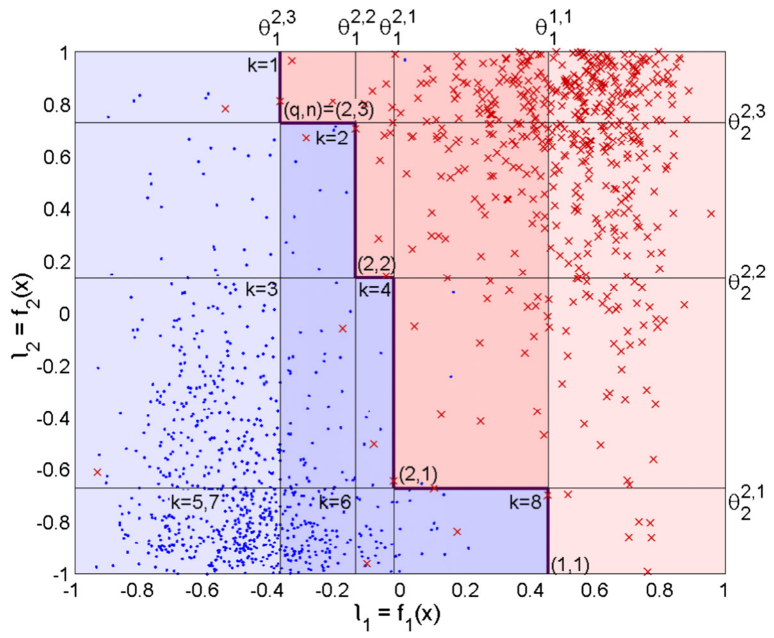


Fig. 2 Example of BOA decision boundary. Illustration of classification of MAHNOB Laughter dataset videos with BOA. Data $\boldsymbol{x} \in X = \{X^0, X^1\}$ from two classes, "laughter" and "speech," is represented in terms of two target likelihood scores l_1 and l_2. The data samples from the "laughter" class X^1 are shown with red crosses and the data samples of the "speech" class X^0 are shown with blue dots. The resulting decision boundary by the BOA combination (10) is shown with the bold angular line. Each threshold $\theta_m^{q,n}$, $m=1,2,\ q=1,2,\ n=1,2,3$ is illustrated with a thin line. The space of target likelihood scores where $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = true$ is colored with pink background, and the space where $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = true$ is colored with blue background. The palest background colors illustrate the subspaces, where the decision is done using the score l_1 only

term $\left[\bigwedge_{i=1}^{M_q}\left(l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n}\right)\right]$ in (7) is a *conjunction* over the Boolean threshold comparisons of the target likelihood scores $\{l_m \,|\, \exists i \;\; m = z_q(i)\}$. The multiplicity of a conjunction type z_q is denoted by N_q.

Every conjunction, enumerated by (q,n), operates with a distinct set of thresholds $\theta_{z_q(i)}^{q,n},\ i=1\ldots M_q$.

The negation of the BOA function (7) is used for the cascade implementation of its evaluation. In the BOA cascade, the classification to the "non-target" class is formulated via the negation of the BOA function—whenever the negated BOA function equals *true*. The Boolean negation of $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ in (7), in disjunctive normal form, is

$$\begin{aligned}\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) &= \bigvee_{k=1}^{K}\left[\bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q}\left(f_{z_q(\mathcal{I}(k,q,n))}(\boldsymbol{x}) < \theta_{z_q(\mathcal{I}(k,q,n))}^{q,n}\right)\right] \\ &= \underbrace{\bigvee_{i_{1,1}=1}^{M_1}\bigvee_{i_{1,2}=1}^{M_1}\bigvee_{i_{1,3}=1}^{M_1}\cdots\bigvee_{i_{1,N_1}=1}^{M_1}}_{N_1 \bigvee\text{-operators , i.e., } M_1^{N1}\text{ conjunctions}}\underbrace{\bigvee_{i_{2,1}=1}^{M_2}\bigvee_{i_{2,2}=1}^{M_2}\cdots\bigvee_{i_{2,N_2}=1}^{M_2}}_{N_2 \bigvee\text{-operators, i.e., } M_2^{N2}\text{ conjunctions}}\cdots \\ &\quad\cdots\underbrace{\bigvee_{i_{Q,1}=1}^{M_Q}\bigvee_{i_{Q,2}=1}^{M_Q}\cdots\bigvee_{i_{Q,N_Q}=1}^{M_Q}}_{N_Q \bigvee\text{-operators, i.e., } M_Q^{N_Q}\text{ conjunctions}}\left[\bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q}\left(f_{z_q(i_{q,n})}(\boldsymbol{x}) < \theta_{z_q(i_{q,n})}^{q,n}\right)\right].\end{aligned} \tag{8}$$

where the number of conjunctions is given by $K = \prod_{q=1}^{Q} M_q^{N_q}$, and the index $\mathcal{I}(k,q,n)$ of the detector function identifier m within vector z_q of the first representation is given by

$$\mathcal{I}(k,q,n) = \left\lfloor \frac{\left\lfloor \frac{k-1}{\prod_{i=q+1}^{Q} M_i^{N_i}} \right\rfloor}{M_q^{N_q-n}} \right\rfloor \bmod M_q \quad +1, \tag{9}$$

Figure 2 illustrates the decision boundary using a BOA combination with $z_1 = [1]$, $z_2 = [1,2]$, and $N_1 = 1, N_2 = 3$, which is

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \left(l_1 \geq \theta_1^{1,1}\right) \vee \bigvee_{n=1}^{3} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right) \right] \tag{10}$$

and its negation is

$$\begin{aligned}
\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) &= \bigvee_{k=1}^{8} \left[\bigwedge_{q=1}^{2} \bigwedge_{n=1}^{N_q} \left(f_{z_q(\mathcal{I}(k,q,n))}(\boldsymbol{x}) < \theta_{z_q(\mathcal{I}(k,q,n))}^{q,n} \right) \right] \\
&= \bigvee_{i_1=1}^{2} \bigvee_{i_2=1}^{2} \bigvee_{i_3=1}^{2} \left[\left(f_1(\boldsymbol{x}) < \theta_1^{1,1} \right) \wedge \bigwedge_{n=1}^{N_q} \left(f_{z_q(i_n)}(\boldsymbol{x}) < \theta_{z_q(i_n)}^{q,n} \right) \right] \\
&= \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right].
\end{aligned} \tag{11}$$

The corners of the resulting decision boundary are formed by the conjunctions $(q,n) = (1,1)$, $(2,1)$, $(2,2)$, and $(2,3)$ of (10), which are designated in Fig. 2 by the conjunction indexes (q,n) next to each corresponding outer corner of space $\{ \ (l_1,l_2) \mid \mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true} \ \}$. The outer corners of space $\{ \ (l_1,l_2) \mid \neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true} \ \}$, which are generated by the conjunctions $k = 1\ldots 8$ of (11), are similarly designated in Fig. 2.

There may be redundancy in the BOA equation or its negation, depending on values of the thresholds selected for $\boldsymbol{\theta}$. A conjunction within a BOA is redundant, if the BOA decision boundary does not change by removing that conjunction from the BOA equation.

Considering a BOA with conjunction lists $z_1, z_2, \ldots, z_Q$ and conjunction multiplicities $N_1, N_2, \ldots, N_Q$, to find out whether a conjunction (q, n_q) is redundant or not, its thresholds $\left\{ \theta_{z_q(i)}^{q,n_q} \mid i = 1 \ldots M_q \right\}$ must be examined. Each threshold $\theta_{z_q(i)}^{q,n_q}$ must be compared to thresholds $\theta_{z_p(j)}^{p,n_p}$, $z_q(i) = z_p(j) = m$, on the same target likelihood score l_m, which are used within other conjunctions (p, n_p) of the BOA. The conjunctions (p, n_p) to be considered are those with z_p containing $m = z_q(i)$ and possibly other identifiers from z_q. The list z_p may not contain identifiers not listed in z_q. Formally, $\{ z_p \mid p \neq q \text{ and } \exists i,j \ni m = z_p(j) = z_q(i) \text{ and } \forall i \in \{1 \ldots M_p\}\ \exists j \ni z_p(i) = z_q(j) \}$. The range of n_p for (p, n_p) is naturally $n_p = 1 \ldots N_p$. For a conjunction (q, n_q) to be non-redundant, one of its thresholds $\theta_{z_q(i)}^{q,n_q}$, $i = 1 \ldots M_q$ must be smaller than any threshold $\theta_{z_p(j)}^{p,n_p}$, $z_q(i) = z_p(j) = m$, in its corresponding conjunctions (p, n_p). That is, in conjunction (q, n_q) there must exist at least one threshold θ_m^{q,n_q} for which $\theta_m^{q,n_q} < \theta_m^{p,n_p}$ of all the corresponding conjunctions (p, n_p).

3.1 BOA as a binary classification cascade

Algorithmically, a BF is evaluated in steps, i.e., sequentially. If any of the conjunctions of BOA function (7) or its negation (8) resolves as *true*, the entire functions (7) and (8) become determinate. In other words, as soon as any of the conjunctions (q,n), $q = 1 \ldots Q$, $n = 1 \ldots N_q$ of a BOA $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ outputs *true*, i.e., $\left[\bigwedge_{i=1}^{M_q} \left(l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n} \right) \right] = \mathit{true}$, it means that $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true}$. Without evaluating the rest of the BOA conjunctions the detection result "target event detected" may then be announced. Similarly, if any of the conjunctions $k = 1 \ldots K$ of the negation of the BOA $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ outputs "true," that is, if $\left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \left(l_{z_q(\mathcal{I}(k,n,q))} < \theta_{z_q(\mathcal{I}(k,n,q))}^{q,n} \right) \right] = \mathit{true}$, it means that $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true}$. The evaluation can then be stopped and the classification result "non-target" can be released.

Computationally, the heaviest part of BOA evaluation is the acquisition of target likelihood scores l_m for an input sample $\boldsymbol{x}$ by computing the functions $f_m(\boldsymbol{x}) = l_m$, $m = 1 \ldots M$. The cost of threshold comparisons within BOA may be considered negligible. From computational aspect of evaluating a BOA, once the likelihood score l_m is acquired, all the Boolean comparisons $(l_m \geq \theta_m^*)$ and $(l_m < \theta_m^*)$, which are based on the score l_m, become immediately available. In case the BOA function (7) or its negation (8) becomes determinate with the Boolean comparisons of already computed subset of scores l_m, $m = 1 \ldots M$, the classification may be released without running the rest of the detector functions at all.

We have implemented the BOA as a binary classification cascade, where a cascade stage $s \in \{1 \ldots S\}$ calculates a score $f_m(\boldsymbol{x}) = l_m = l_s$ using a predefined detector function f_m and offers a possibility for releasing the classification result, as shown in Fig. 3. Internal decisions at each stage $s = 1, 2, \ldots, S$ of the BOA cascade, whether to release a class estimate or to enter the next cascade stage, are made with BFs $B_s^{\text{class}}((l_1, l_2, \ldots, l_s)),\ \ s = 1 \ldots S$, i.e. $B_1^1(l_1)$, $B_1^0(l_1)$, $B_2^1(l_1, l_2)$, $B_2^0(l_1, l_2)$,..., $B_S^1(l_1, l_2, \ldots, l_S)$ and $B_S^1(l_1, l_2, \ldots, l_S)$. That is, the functions B_s^1 and B_s^0 of cascade stage s utilize the target likelihood scores $l_1, l_2, \ldots, l_s$. All these functions are partitions of the BOA function $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ of (7) and its negation $\neg\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ of (8) such that

$$\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = B_1^1 \vee B_2^1 \vee \ldots \vee B_S^1 \qquad (12)$$

and

$$\neg\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = B_1^0 \vee B_2^0 \vee \ldots \vee B_S^0. \qquad (13)$$

Formal expressions for the partition of the BOA function (7) into functions $B_1^1, B_2^1, \ldots, B_S^1$ and the BOA negation (8) into functions $B_1^0, B_2^0, \ldots, B_S^0$ are derived in Appendix 2.

As an example, operation process of BOA cascade $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = \left(l_1 \geq \theta_1^{1,1}\right) \vee \bigvee_{n=1}^{3} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right) \right]$ for MAHNOB laughter data classification is illustrated in the Fig. 2 with the background color of the l_1-vs. l_2-axis and is as follows. The classification takes place at the first cascade stage for all the samples $\boldsymbol{x}$ for whom $B_1^1(\boldsymbol{x}) = \left(l_1 \geq \theta_1^{1,1}\right) = true$ or $B_1^0(\boldsymbol{x}) = \left(l_1 < \min\left(\theta_1^{2,1}, \theta_1^{2,2}, \theta_1^{2,3}\right)\right) = \left(l_1 < \theta_1^{2,3}\right) = true$. In the first case, the classification is "Laughter detected," and in the second case "No Laughter." These subspaces of (l_1, l_2) on the left and right outskirts of Fig. 2 are indicated with a pale background color. In the second stage of the cascade processing, the likelihood $f_2(\boldsymbol{x}) = l_2$ is computed only for the samples with $\theta_1^{2,3} \leq l_1 < \theta_1^{1,1}$, although l_2 is shown for all the samples in the Fig. 2. With the dataset in the Fig. 2, it means that classification of approximately 65% of the samples are made using the detector function f_1 only.

The computational efficiency of the cascade naturally depends on the order of detector methods to be utilized at cascade stages. Generally, the faster methods should be evaluated first, and the slower ones later. If the methods $f_m,\ m = 1 \ldots M$ have very different computational loads $\mathcal{L}_m,\ m = 1 \ldots M$, it is very likely that a cascade ordered such that $\mathcal{L}_s \ll \mathcal{L}_{s+1}$, $s = 1 \ldots S-1$ is the most efficient one. Precisely, the most computationally efficient cascade structure may be defined via local inequalities among each two consecutive stages s and $s+1$ as follows. If we denote the probability of a sample arriving stage s to be classified at stage s after computing $l_s = f_s(x)$ with P_1, and the probability of a sample arriving stage s to be classified at stage s if the detector method f_{s+1} would be utilized instead of the method f_s with P_2, it must hold that $P_1 \geq (\mathcal{L}_s/\mathcal{L}_{s+1})P_2$.

In our work, the computational loads of the detector methods are very different from each other, i.e. $\mathcal{L}_s/\mathcal{L}_{s+1} \ll 1$. Thus within the BOA cascade the detector methods f_m $m = 1 \ldots M$ are ordered according to their computational loads. For notational simplicity we assume that the detector methods f_m used in a BOA cascade are enumerated such that for their computational loads $\mathcal{L}_m$ it holds that $\mathcal{L}_m \ll \mathcal{L}_{m+1}$, and now in a BOA cascade $f_s = f_m$. The Table 3 demonstrates the computational efficiency achieved in our experiments.

For a sample in a dataset X, the computational load of classification with a BOA cascade is on average

$$\sum_{m=1}^{S} \left(1 - \frac{\left| \left\{ \boldsymbol{x} \in X,\ \bigvee_{s=1}^{m-1} B_s^1(\boldsymbol{x}) \vee B_s^0(\boldsymbol{x}) = true \right\} \right|}{|X|} \right) \cdot \mathcal{L}_m.$$

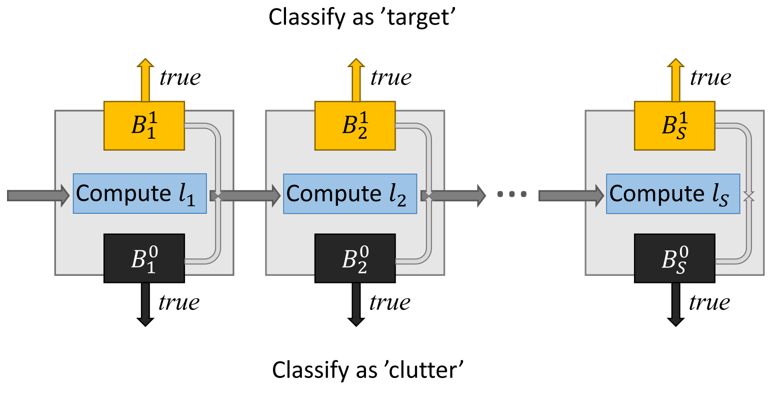


Fig. 3 BOA cascade. Classification process with BOA classification cascade. At each stage s of the cascade new target likelihood score $f_s(\boldsymbol{x}) = l_s$ is computed and either classification is made, or next stage is entered. The internal Boolean decision makers $B_1^1, B_2^1, \ldots, B_S^1$ are partitions of BOA function (7), and $B_1^0, B_2^0, \ldots, B_S^0$ are partitions of (8)

To design a specific type of BOA cascade, e.g., one-sided or symmetrical, the lists z_q, $q = 1 \ldots Q$, which determine the detector functions to be utilized within the conjunctions of the BOA, must be selected appropriately. For data with clearly unbalanced class distribution, one-sided cascade is computationally efficient if the early classification option is available for the prevalent class. This is the case if the decision-makers $B_s^{\text{prevalent}}$, $s = q \ldots S$ for the prevalent class are functioning while the decision-makers B_s^{rare}, $s = 1 \ldots S-1$ for the rare class are null/nonexistent as $B_s^{\text{rare}}(\boldsymbol{x}) = \textit{false} \ \forall \ \boldsymbol{x}$. Thus, for the usual case, where the "target" class is rare and the "non-target" class is the prevalent one, to ensure a computationally efficient one-sided BOA cascade, the BOA must be conjunctive, designed with only one conjunction list $z_1 = [1, 2, \ldots, S]$. In case the target likelihood scores are negated, i.e., $-f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}), \ldots, -f_S(\boldsymbol{x})$ are used, conjunction list of every subvector of $[1, 2, \ldots, S]$, should be used to build a one-sided cascade capable of early classification to "non-target" class. For example in case $S = 3$, the conjunction lists would thus be $z_1 = [1]$, $z_2 = [2]$, $z_3 = [3]$, $z_4 = [1, 2]$, $z_5 = [1, 3]$, $z_6 = [2, 3]$ and $z_7 = [1, 2, 3]$.

A symmetrical cascade, which enables early classification to both the classes at all the cascade stages, is suitable for classification tasks with both even and unbalanced class distributions. The time to decision efficiency of the cascade depends on capability of all the internal decision makers B_s^1 and B_s^0 for $s = 1 \ldots S$ of the cascade to make early classifications. Functioning decision makers for all the stages and both the classes to build a symmetrical BOA cascade are ensured by constructing the BOA from cumulative conjunction lists $z_1 = [1]$, $z_2 = [1, 2], \ldots, z_S = [1, 2, \ldots, S]$, that is $z_s = [1, 2, \ldots, s]$.

3.2 BOA tunability property

Classification performance of the BOA depends on all the values of thresholds $\theta_m^{q,n}$, $m = 1 \ldots M$, $q = 1 \ldots Q$, $n = 1 \ldots N_q$ in $\boldsymbol{\theta}$. Classifying data $X = \{X^0, X^1\}$ from two classes with a BOA $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ results in certain true positive rate $tpr_{\boldsymbol{\theta}}$ and false positive rate $fpr_{\boldsymbol{\theta}}$, which produces one point into a space of precision (P) vs. recall (R). Classifying the data X with the BOA $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ with all the possible sets of different threshold values in $\boldsymbol{\theta}$ results in a constellation of performance points in (P, R) space. Best performing threshold values for the BOA are those corresponding to the classification performance on the upper frontier of this (P, R) constellation.

We want to make the BOA sensitivity tunable with a single parameter in similar way to individual detectors. For that, we introduce a parameter $\alpha \in [0 \ldots 1]$, which denotes the sensitivity setting of a BOA. A value of the parameter α corresponds to a fixed set $\boldsymbol{\theta}_\alpha$ of the BOA threshold values such that $\mathrm{B}(\boldsymbol{x}; \alpha) = \mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}_\alpha)$. In the next section, we introduce an algorithm to select threshold values for $\boldsymbol{\theta}_\alpha$ for a range of values of the sensitivity parameter α. These operating points result in the BOA performance to be close to the upper frontier of the (P, R) constellation of BOA performance with all the possible settings of $\boldsymbol{\theta}$.

The user may then select for a BOA $\mathrm{B}(\boldsymbol{x}; \alpha)$ the operating point α with the most desirable behavior with the factual costs of a false positive $\mathcal{C}_{fp}$ and a false negative $\mathcal{C}_{fn}$ of the problem. The operating point α^* of minimal expected misclassification cost can be found at

$$\alpha^* = \min_{\alpha} \left(P\left(\boldsymbol{x} \in X^1\right) \cdot \left(1 - tpr_\alpha\right) \cdot \mathcal{C}_{fn} + P\left(\boldsymbol{x} \in X^0\right) \cdot fpr_\alpha \cdot \mathcal{C}_{fp} \right).$$

where $P\left(\boldsymbol{x} \in X^1\right)$ and $P\left(\boldsymbol{x} \in X^0\right)$ are the prior probabilities of the classes.

3.3 The proposed algorithm to set parameters of a BOA

We train the BOA $\mathrm{B}(\boldsymbol{x}; \alpha)$ by finding suitable values for thresholds $\boldsymbol{\theta}_\alpha$ for a range of values of $\alpha \in [0 \ldots 1]$ in terms of training data X. The possible threshold values ϑ_m considered for a target likelihood score l_m are given by the scores of target class samples $\boldsymbol{x} \in X^1$ as $\vartheta_m = f_m(\boldsymbol{x}) = l_m$.

The proposed algorithm, BOATHRESHOLDSEARCH, for training a BOA is presented in Algorithm 1. As input, the algorithm needs training data $X = \{X^0, X^1\}$ from two classes, the conjunction lists $z_1, z_2, \ldots, z_Q$, maximal conjunction set multiplicities $N_1, N_2, \ldots N_Q$ of the BOA and the maximal number $N_S^{\max}$ of candidates for $\boldsymbol{\theta}_\alpha$ saved by the algorithm for each α. The algorithm produces sets $\boldsymbol{\theta}_{\alpha_t}$ of fixed threshold values for BOA operating points $\alpha_t = \frac{t}{T}$, $t = 0 \ldots T$, where T equals the number of samples $\boldsymbol{x} \in X^1$. These operating points correspond to true positive rates $0, \frac{1}{T}, \frac{2}{T}, \ldots, \frac{T-1}{T}, 1$ on training data X. The algorithm searches for suitable threshold values step by step starting by selecting values for $\boldsymbol{\theta}_0$ for $\alpha_0 = 0$ and terminating after selecting values for $\boldsymbol{\theta}_1$ for $\alpha_T = 1$. The method is greedy in a sense that when searching for values for α_t at iteration t, the search starts from a potential set of threshold values for α_{t-1} provided by iteration $t-1$, and the threshold values are allowed to change only gradually for minimizing the number of false positives locally.

The algorithm starts by fixing the BOA thresholds for sensitivity level $\alpha_0 = 0$ to be $\boldsymbol{\theta}_{\alpha_0} = \{\infty\}$. The BOA with parameter setting $\alpha_0 = 0$ does not accept any sample to the "target" class, i.e., $\mathrm{B}(\boldsymbol{x}; \alpha_0) = \textit{false} \ \ \forall \ \ \boldsymbol{x} \in X$. Thus, the algorithm starts with $tpr_{\alpha_0} = fpr_{\alpha_0} = 0$. The threshold setting $\boldsymbol{\theta}_{\alpha_0}$ and the corresponding number 0 of false positives are placed into a set $\mathcal{S}_0$ as an entry $(\boldsymbol{\theta} = \{\infty\}, fp = 0)$ for the next step to start with.

At each step $t = 1 \ldots T$, every threshold setting $\boldsymbol{\theta}$, given by entries $(\boldsymbol{\theta}, fp)$ in $\mathcal{S}_{t-1}$, provided by the step $t-1$, is adjusted. One adjusted set $\boldsymbol{\theta}^{\text{new}}$ is obtained by mitigating one or multiple thresholds $\theta_m^{q,n} \in \boldsymbol{\theta}$ of one

Algorithm 1 An algorithm to find thresholds for a BOA combination

```
1: procedure BOATHRESHOLDSEARCH
2:     input: $\mathcal{Z}, \mathrm{N}, \mathcal{F}, X, N_{\mathcal{S}}^{\max}$
           # $\mathcal{Z}$ contains the $Q$ conjunction lists $z_1, \ldots, z_Q$.
           # N contains the maximal conjunction multiplicities $N_1, \ldots, N_Q$.
           # $\mathcal{F}$ contains the detector functions $f_m$, $1 = 1 \ldots M$.
           # $X = X^0 \cup X^1$ is the training data from two classes.
           # $N_{\mathcal{S}}^{\max}$ is the maximal size of set $\mathcal{S}_t$ used within the algorithm.
3:     for m= 1...M do
4:         Set $\vartheta_m = \{f_m(\boldsymbol{x}) \mid \boldsymbol{x} \in X^1\}$
5:     end for
6:     Set $\alpha_0 = 0$, $\boldsymbol{\theta}_0 = \{\infty\}$, $tp_0 = 0$, $fp_0 = 0$
7:     Set $\mathcal{S}_0 = \{(\boldsymbol{\theta} = \{\infty\}, fp = 0)\}$.
8:     for $t = 1 \ldots |X^1|$ do
9:         Set $\mathcal{S}_t = \emptyset$
10:        for each $(\boldsymbol{\theta}, fp) \in \mathcal{S}_{t-1}$ do
11:            for each $(q, n)$ for $q = 1 \ldots Q$, $n = 1 \ldots N_q$ do
12:                for each subvector $\zeta$ of $z_q$ do
13:                    Set $\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}$
14:                    Within $\boldsymbol{\theta}^{\text{new}}$, set $\theta_m^{q,n} = \theta_m^{q,n} - \delta_m$, for all $m = \zeta(i)$, $i = 1 \ldots M_\zeta$
                       using such $\delta_m$ that $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$ accepts exactly one more
                       sample $\boldsymbol{x} \in X^1$ than $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$, and $\theta_m^{q,n} \in \vartheta_m$.
15:                    Count the number of false positives $fp^{\text{new}}$ with $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$.
16:                    Set $\theta_{\text{all}}^{\gamma,\nu} = \infty$ for every conjunction $(\gamma, \nu)$ of $\boldsymbol{\theta}^{\text{new}}$ which is redundant.
17:                    Set $\mathcal{S}_t = \mathcal{S}_t \cup (\boldsymbol{\theta}^{\text{new}}, fp^{\text{new}})$.
18:                end for
19:            end for
20:        end for
21:        Set $\alpha_t = \frac{t}{T}$, $tp_t = tp_{t-1} + 1$, $fp_t = \min_{(\boldsymbol{\theta}, fp) \in \mathcal{S}_t} fp$
22:        Set $\boldsymbol{\theta}_t = \boldsymbol{\theta}^*$ such that $(\boldsymbol{\theta}^*, fp_t) \in \mathcal{S}_t$ and within $\boldsymbol{\theta}^*$ the number of
               conjunctions $(\gamma, \nu)$ for which $\theta_{\text{all}}^{\gamma,\nu} < \infty$ is the smallest.
23:        Prune $\mathcal{S}_t$ by keeping $N_{\mathcal{S}}^{\max}$ entries with the smallest $fp$.
24:    end for
25:    return: $\alpha_t$, $\boldsymbol{\theta}_t$, $tp_t$, $fp_t$ $\forall$ $t = 1 \ldots |\{p\}|$.
26: end procedure
```

conjunction (q, n) of the BOA. Within each BOA conjunction (q, n), there are $2^{M_q} - 1$ subsets of thresholds $\left\{\theta_{z_q(i)}^{q,n} \mid i \subseteq \{1 \ldots M_q\}\right\}$ to search for the best change from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^{\text{new}}$. Thus in the complete BOA function there are $P = \sum_{q=1}^{Q} N_q \cdot \left(2^{M_q} - 1\right)$ possible subsets of thresholds to change, and thus one $\boldsymbol{\theta}$ generates up to P changed threshold settings $\boldsymbol{\theta}^{\text{new}}$.

When mitigating the values of thresholds $\left\{\theta_{z_q(i)}^{q,n} \mid i \subseteq \{1 \ldots M_q\}\right\}$ of a conjunction (q, n) from their values in $\boldsymbol{\theta}$ for $\boldsymbol{\theta}^{\text{new}}$, the amount of changes are such that $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$ accepts exactly one more sample $\boldsymbol{x} \in X^1$ than $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$. That is, $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = \mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$ $\forall$ $\boldsymbol{x} \in X^1 \backslash \boldsymbol{x}_*$, $\mathrm{B}(\boldsymbol{x}_*; \boldsymbol{\theta}) = \textit{false}$ and $\mathrm{B}(\boldsymbol{x}_*; \boldsymbol{\theta}^{\text{new}}) = \textit{true}$. If redundancy of BOA function appears with the new threshold set $\boldsymbol{\theta}^{\text{new}}$, all the thresholds $\theta_{z_q(i)}^{q,n}$, $i = 1 \ldots M_q$ of the redundant conjunctions (q, n) are reset to be $\theta_*^{q,n} = \infty$. All the acquired new settings $\boldsymbol{\theta}^{\text{new}}$ are saved with their resulting false positive counts into a set $\mathcal{S}_t$ as entries $\{(\boldsymbol{\theta}, fp)^{\text{new}}\}$ to be potential settings for α_t.

After processing every entry $(\boldsymbol{\theta}, fp) \in \mathcal{S}_{t-1}$ and saving all the generated new entries into $\mathcal{S}_t$, the best set $\boldsymbol{\theta}^*$ of BOA thresholds among the entries of $\mathcal{S}_t$ is selected for $\boldsymbol{\theta}_{\alpha_t} = \boldsymbol{\theta}^*$ to correspond to α_t. The best set $\boldsymbol{\theta}^*$ is a selected to be the one corresponding to the smallest number of false positives among the entries in $\mathcal{S}_t$ and using as few BOA conjunctions as possible with non-infinite thresholds. The set $\mathcal{S}_t$ is then pruned to keep the maximal allowed number $N_{\mathcal{S}}^{\max}$ of the best entries for the next step to start with. In the experiments, we used $N_{\mathcal{S}}^{\max} = 10$, as larger number did not improve the recognition accuracy notably while making the algorithm run remarkably slower.

Figure 4 illustrates the thresholds $\boldsymbol{\theta}_\alpha$ found by the algorithm with $N_{\mathcal{S}}^{\max} = 1$ for a BOA

$$\mathrm{B}(\boldsymbol{x}; \alpha) = \mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}_\alpha) = \left(l_1 \geq \theta_1^1\right) \vee \left[\left(l_1 \geq \theta_1^2\right) \wedge \left(l_2 \geq \theta_2^2\right)\right]. \tag{14}$$

for $\alpha = 0, \frac{1}{T}, \frac{2}{T}, \ldots, \frac{T-1}{T}, 1$.

The memory requirement of the algorithm, besides the training data and the output variables, during the algorithm run is the storage needed for the set $\mathcal{S}_t$ of the potential operating points to be stored at each iteration. As maximally $N_{\mathcal{S}}^{\max}$ operation points are passed from one iteration to the next one, the number of operation points to be held in memory during an iteration of the algorithm run is maximally $N_{\mathcal{S}}^{\max} \times \sum_{q=1}^{Q} N_q \left(2^{M_q} - 1\right)$.

Computational complexity of the BOATS algorithm is $\mathcal{O}\left(|X^1| \, N_{\mathcal{S}}^{\max} \, N_{\text{conj}}^{\max} \, 2^M\right)$. In practice, multiple positive samples are often selected concurrently, diminishing the multiplier $|X^1|$. The limit $N_{\mathcal{S}}^{\max}$ is an input parameter which allows the user to decide about the accuracy vs time and memory complexity trade-off of the algorithm. $N_{\text{conj}}^{\max}$ is the maximum number of conjunctions in the DNF-BF BOA-function, which takes place at the operating point of recall= 1. At operating points with lower recall values, the true value is generally lower, and using $N_{\text{conj}}^{\max}$ sets upper

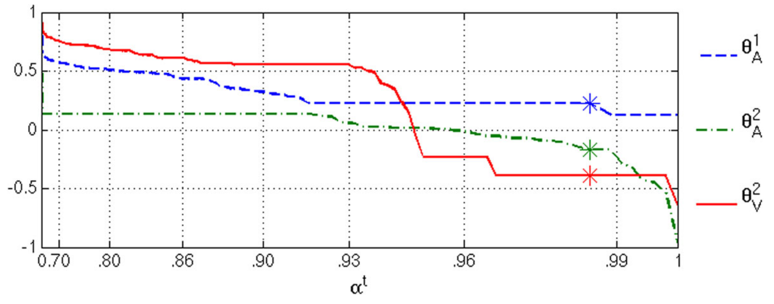


Fig. 4 BOA training. The sequence of thresholds $\boldsymbol{\theta}_{\alpha_t} = \left[\theta_A^1, \theta_A^2, \theta_V^2\right]_t$, $t=0 \ldots T$ found by the proposed BOATS algorithm for a BOA (14) with $N_{\boldsymbol{S}}^{\max} = 1$. Thresholds of the operating point $\boldsymbol{\alpha}_t$ with highest accuracy on train data is marked with asterisks

limit for the time complexity. The number 2^M is upper limit of options tested when processing each conjunction, the true number for each conjunction (q, n) is $2^{M_q} - 1$.

4 Results and discussion

In this section, we report our experiments to evaluate the performance of the proposed BOA cascade of multiple sensitivity tunable detectors both in terms of detection accuracy and computational load of classification. We also analyze the proposed BOA training algorithm to showcase how good operating points it can find for a BOA combination. To substantiate the eligibility of our work, we compare the acquired results with others found in the literature.

We first introduce the datasets used for the two explored tasks, namely laughter detection and context change detection, and discuss the used performance measures. Then, we contrast our results with the proposed BOA classifier and a C5.0 -tree classifier in laughter detection task to results by other solutions found in the literature. We also compare the proposed BOA training algorithm to other training algorithms adopted from literature and explore the detection performance with different BOA combinations.

4.1 Data and performance measures

4.1.1 MAHNOB Laughter dataset

For laughter detection, i.e., laughter vs speech classification, we use data from the MAHNOB Laughter dataset of [7]. The data consists of 1399 video clips of lengths from 0.15 s to 28 s of 22 different persons. 845 of the video clips represent speech and 554 of them represent laughter. The data is recorded in two modilities; frontal closeup video with frame rate 25 fps, and audio from a lapel microphone with sampling frequency 44.1 kHz.

A frame from one of the videos is shown in Fig. 5 to demonstrate the data.

We run the tests using 22-fold cross-validation where at each fold the videos of one person are left out for testing, and all the rest of the videos are used for training.

We build the BOA combinations using similar classifiers as are used for the baseline method in [7]. Those are an audio stream based detector, which provides laughter likelihood $f_A(\boldsymbol{x}_{\text{audio}}) = l_A$, and a video frame based detector, which provides laughter likelihood $f_V(\boldsymbol{x}_{\text{visual}}) = l_V$ for each video clip. The computational load of the audio stream based detector is very small compared to the computational load of the visual stream based detector.

The audio stream-based laughter detector utilizes the 6 first MFCC features from audio frames of length 20 ms. A single output feedforward neural network (NN) is trained to produce audio frame-wise target class likelihoods l_a using mean squared error (MSE) error function. The NN has one hidden layer with 20 neurons and all the neurons of the network use tangential sigmoid transfer function. The target class likelihood l_A for a video clip is an average over the frame-wise values as $l_A = \frac{1}{N_a}\sum_{\tau=1}^{N_a} l_a(\tau)$, where N_a is the number of audio frames in the clip.

The video frame-based laughter detector starts with extracting the 20 face points, shown in Fig. 6, from each video frame using an algorithm from [49]. The utilized face points correspond to points used in [7]. Then, the dimensionality of each face point feature vector is

Fig. 5 A video frame from MAHNOB Laughter data set. A video frame from MAHNOB Laughter data set. This frame is from a video which contains laughter

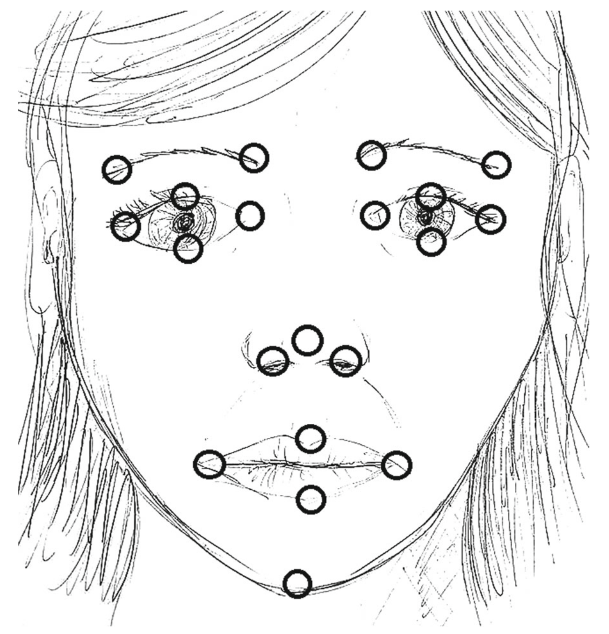


Fig. 6 Face points. The 20 face points used as features for laughter-speech classification of MAHNOB Laughter dataset videos

reduced from 40 to 20 by principal component analysis (PCA). For frame-wise laughter likelihood estimates l_v an NN is trained. It is built of 1 hidden layer of 10 neurons. All the neurons use tangential sigmoid transfer function, and mean squared error (MSE) loss function is applied for training. Video clipwise laughter likelihood is given as an average over the frame-wise values as $l_V = \frac{1}{N_V}\sum_{\tau=1}^{N_V} l_v(\tau)$, where N_V is the number video frames in the clip.

4.1.2 CASA dataset

For video context change detection problem we use CASA database[1] from [8]. Over 7 h of lifelog video material is filmed with a small pen camera, which operates at frame rate 15 frames/second and frame size 176×144 pixels. The stereo sound track is recorded by a pair of in-ear microphones with 44.1 kHz sampling rate and stored without compression. The database contains video material from 23 different types of environments.

For a context change detection task we created 30 video files of length 5–20 min. Each file is concatenated on average of 105 clips of length 1–30 s from the video material of CASA database. The context—one of the 23 different environments included in the database—is kept the same for 1–5 successive clips, otherwise each clip is taken from a randomly selected video file. There are on average 42 context changes within each created video file. We run our tests using 6-fold cross-validation, where at each fold 5 files are reserved for testing and the remaining 25 files are used for system training.

We use three different detectors to spot context changes in the created videos. Brief descriptions of the used detectors are given here, while the details of them can be found in [50]. The fastest one of the used detectors operates on the audio stream of the video. The audio is analyzed in frames of length 80 ms with 40 ms overlap of successive frames. From each audio frame, MFCC features are computed, and within a sliding window of 125 audio frames, mean and variance of 20 MFCC coefficients are computed. Transitions in these statistics are converted to a context change likelihood l_1 for each audio frame. The computation time of scores l_1 on a single CPU desktop computer is 0.8 ms per audio frame, that is 10 ms per one second of audio.

Two other utilized context change detectors operate on the image modality of the video. The faster one of the detectors on visual modality collects RGB histograms of video frames and produces the context change likelihood value l_2 for each video frame according to the city block distance between adjacent RGB histograms. The computation time of l_2 is about 29 ms per video frame, that is approximately 435 ms per one second of video.

The more accurate one of the used detectors on visual modality, proposed in [51], counts incidences of SIFT descriptor codebook elements within each video frame, and collects a SIFT histogram, i.e., so-called bag-of-words feature vector, for each video frame. The context change likelihood value l_3 for each video frame is computed as the city block distance between SIFT-histograms of successive video frames. The computation time of l_3 is about 12.3 s per video frame, that makes about 184 s per one second of video.

4.1.3 Performance measures

In the literature the performance of detectors is often presented by a receiver operation characteristic (ROC) curve. However, in our evaluations, we prefer the curve of *precision* vs *recall* (P-R curve) because in case of imbalanced class distributions P-R curve is more faithful to the absolute number of erroneous classifications than the ROC -curve of *tpr* in respect to *fpr*. To demonstrate the performance of a certain operating point of a detector, we use measures like accuracy, F_1-score, and computational load.

Average values of these performance numbers over cross-validation folds are presented as results. With MAHNOB laughter dataset, 22-fold cross-validation is used. In each fold, video files of one speaker are used for testing, and the rest of the files are used for training the component detectors and the BOA -cascade. With CASA dataset, 6-fold cross validation is used similarly. In each fold, 25 video files are used for training the individual classifiers and the BOA -combination, and 5 files are used for testing the system.

4.2 Comparing BOA cascade to existing work in laughter vs speech classification

We compare the performance of the proposed BOA cascade to results we obtained with C5.0 -tree building algorithm [52] as well as results obtained by other authors in laughter vs speech classification, i.e., laughter detection, with the MAHNOB laughter dataset. For the task, we use a BOA detector

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \left(l_A \geq \theta_A^1\right) \vee \bigvee_{n=1}^{N_2} \left[\left(l_A \geq \theta_A^{2,n}\right) \wedge \left(l_V \geq \theta_V^{2,n}\right) \right], \tag{15}$$

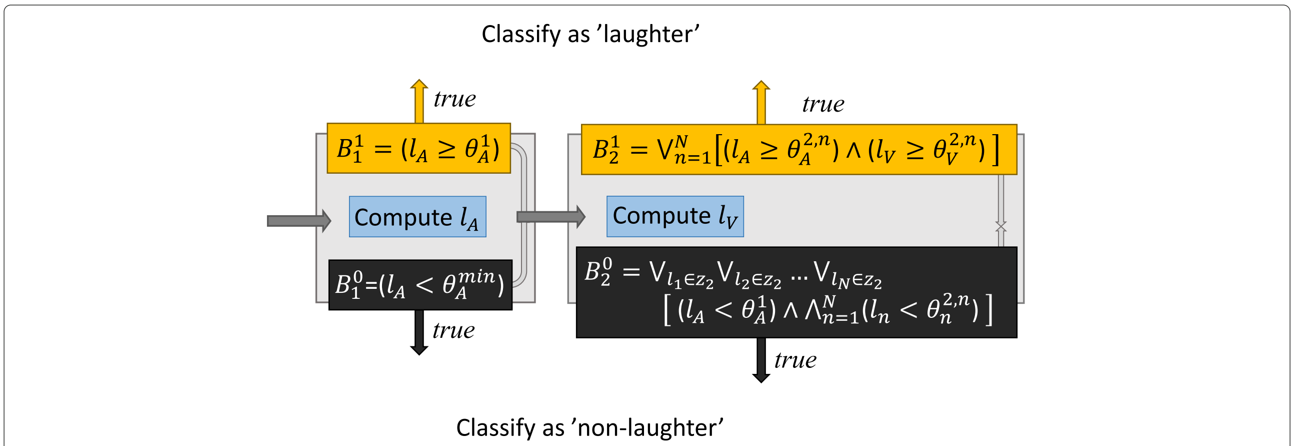
whose threshold parameters θ_A^1, $\theta_A^{2,1}$, $\theta_V^{2,1}$, $\theta_A^{2,2}$, $\theta_V^{2,2}$, ...,$\theta_A^{2,N}$, $\theta_V^{2,N}$ are learned by the proposed training algorithm. The Fig. 7 illustrates the BOA cascade of (15). The computational load of acquiring l_A from audio stream is only a fraction of the load of computing l_V from video frames. Thus, the ratio of samples that need the computation of l_V reflects well the average computational load of classifying a sample with BOA cascade of (15).

Table 1 presents results with C5.0 tree building algorithm as well as those found in the literature in contrast to our solution. We report performance numbers with a BOA cascade of (15) with $N = 1$ and also with N selected adaptively by the proposed training algorithm. The decision trees obtained with C5.0 algorithm [52] are converted to DNF-BF -form (15) and evaluated in cascaded manner similarly to BOA evaluation. The number N in the DNF-BF (15) of a tree varies according to the structure of the tree, which is given by the algorithm. The minimal leaf size of a tree was defined by 10-fold cross validation using the training data. The boosted C5.0 forest contains 10 trees trained with different weightings by the training algorithm on training samples. The classification of the forest is obtained via voting by the trees.

The C5.0 forest outperforms all the other solutions in terms of classification accuracy, whereas the performance of single C5.0 tree is comparable to performance obtained with BOA classifiers. When a C5.0 tree is evaluated in cascaded manner, very similar computational savings as with a BOA cascade are obtained. Both the BOA detectors outperform the solutions of [53, 54], albeit the classifier in [54] is trained with another database, which likely explains its lower detection accuracy. The results obtained by [55] reach similar accuracy and F_1-scores than our BOA cascades, but their result is not fully comparable as they use only a subset of 15 speakers out of 22 used by all the other authors. However, the computational load of our solution is significantly lower, compared to all these other multimodal solutions. With our BOA cascade of (15) with $N = 1$, only 11% of samples needed the computation of l_V, thus it is about nine times faster than the other solutions. The BOA cascade of (15) with N selected by the proposed training algorithm reaches slightly higher accuracy than the reference solutions while being still three times faster than them.

4.3 Comparing training algorithms for BOA combination

We use the CASA lifelog data and the context change detection task for illustrating the capability of the proposed training algorithm to find successful operating points for a BOA combination. For context change detection we use BOA combinations built of three detectors,



Classify as 'laughter'
true
$B_1^1 = (l_A \geq \theta_A^1)$
Compute l_A
$B_1^0 = (l_A < \theta_A^{min})$
true
true
$B_2^1 = \vee_{n=1}^N[(l_A \geq \theta_A^{2,n}) \wedge (l_V \geq \theta_V^{2,n})]$
Compute l_V
$B_2^0 = \vee_{l_1 \in z_2} \vee_{l_2 \in z_2} \dots \vee_{l_N \in z_2} [(l_A < \theta_A^1) \wedge \wedge_{n=1}^N (l_n < \theta_n^{2,n})]$
true
Classify as 'non-laughter'

Fig. 7 Symmetrical BOA cascade for laughter detection. Symmetrical BOA cascade, which is capable of making early classification to both classes, realizing of BOA (15) for laughter detection. The conjunction set $z_2 = \{l_A, l_V\}$ and the threshold θ_A^{min} is $\theta_A^{min} = \min\left(\theta_A^1, \theta_A^{2,1}, \theta_A^{2,2}, \dots, \theta_A^{2,N}\right)$. B_2^0 contains $K = 2^N$ conjunctions, where the first threshold comparison is always $\left(l_A \geq \theta_A^1\right)$. The comparisons indexed by $n = 1 \dots N$ operate either on l_A or l_V according to binary (M_2-ary) N-digit representation of the conjunction index $k = 1 \dots K$, bin(k). If the n:th digit of bin(k) is 0, l_A is used, and l_V is used if the n:th digit of bin(k) is 1

Table 1 Results in laughter detection task

	acc.	F_1^{sp}	F_1^{lg}	v.f. %
$(l_A \geq \theta_A)$	95.1	.944	.927	0%
$(l_V \geq \theta_V)$	84.4	.818	.795	100%
BOA cascade of (15), $N = 1$	96.0	.966	.958	11%
BOA cascade of (15), N by BOATS	96.9	.972	.955	33%
C5.0 tree	96.7	.965	.948	23%
Boosted C5.0 forest	97.3	.978	.958	≈ 100%
[53]	92.7	.943	.905	100%
[54][a]	91.7	.932	.893	100%
[55][b]	96.9	.973	.963	100%

Comparison of laughter detectors on MAHNOB laughter data. The used measures of performance are the overall accuracy, F_1 -scores for both speech (F_1^{sp}) and laughter (F_1^{lg}), and percentage of video clips, the classification of which utilized also visual features (**v.f.**). The BOA detectors are used at the operating point $\boldsymbol{\alpha}$ of the highest accuracy on training set

[a]Comparison with [54] is not directly comparable, as the classifier in [54] is trained with another dataset

[b]Results of [55] are with 15 speakers while the other authors use 22 speakers in their tests

which are introduced in "Data and performance measures." We train the thresholds of a BOA with the proposed training algorithm (BOATS) and two reference algorithms adapted form literature, and then compare the resulting F_1-scores of classification. The reference algorithms that we use for this evaluation are iterative exhaustive search (IES) based on work in [11] and Boolean algebra of ROC curves (BAROC) introduced in [10]. The implementations of IES and BAROC, adapted for BOA training, are presented in Algorithms 2, 3 and 4 in Appendix 1. The iterative framework used of both the algorithms is presented in Algorithm 2. The Algorithm 3 shows the core operations of IES, and the Algorithm 4 presents the operations for BAROC.

Figure 8 shows the F_1-scores with operating points obtained with three algorithms, BOATS, IES and BAROC, for a BOA

$$\mathrm{B_{AND}} = \bigvee_{n=1}^{N} \left[\left(l_1 \geq \theta_1^n\right) \wedge \left(l_2 \geq \theta_2^n \wedge \left(l_3 \geq \theta_3^n\right) \right] \quad (16)$$

with different conjunction multiplicities N. The IES algorithm can be seen to find the best operating point when $N = 1$ with its exhaustive search. However, when N is increased, IES is unable to improve the BOA performance due to that the suboptimal operating points of each individual conjunction, which nevertheless might produce better performance when used within a disjunctive combination, are pruned by the algorithm.

The BAROC algorithm performs worse than the other algorithms due to its assumption of detector independence, which does not hold with the two visual stream based detectors. Moreover, by the definition of the Boolean algebra of ROC curves in (4) and (5), BAROC is unable to find the opportunities provided by utilizing multiple conjunctions over the same conjunction set.

The proposed BOATS algorithm finds suboptimal operating points for the BOA, but is able to utilize the opportunities offered by using multiple conjunctions over the same conjunction set, and thus outperforms the IES algorithm with $N > 1$. The performance ceases to improve when the conjunction multiplicity grows larger than 7. This is due to both the data characteristics and algorithm behavior favoring small number of conjunctions, i.e., small N.

In a Table 2, we show the best F_1-scores of the operating points found by the three algorithms for BOA combinations

$$\begin{aligned}
\mathrm{B_{OR}} &= (l_1 \geq \theta_1) \vee (l_2 \geq \theta_2) \vee (l_3 \geq \theta_3) \\
\neg\mathrm{B_{\neg OR}} &= \neg\Big[(-l_1 \geq \theta_1) \vee (-l_2 \geq \theta_2) \vee (-l_3 \geq \theta_3) \Big] \\
\mathrm{B_{AND}} &= \bigvee_{n=1}^{N} \Big[\left(l_1 \geq \theta_1^n\right) \wedge \left(l_2 \geq \theta_2^n\right) \wedge \left(l_3 \geq \theta_3^n\right) \Big]
\end{aligned} \quad (17)$$

$$\begin{aligned}
\mathrm{B}_{\mathcal{P}} &= \bigvee_{q=1}^{7} \bigvee_{n=1}^{N_q} \bigwedge_{i=1}^{M_q} \left(l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n} \right) \\
\neg\mathrm{B}_{\neg\mathcal{P}} &= \neg\left[\bigvee_{q=1}^{7} \bigvee_{n=1}^{N_q} \bigwedge_{i=1}^{M_q} \left(-l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n} \right) \right].
\end{aligned}$$

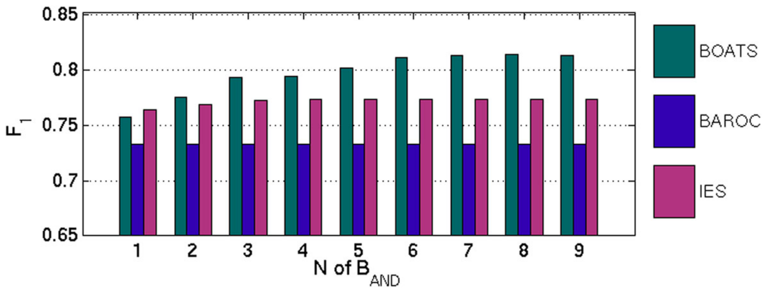


Fig. 8 Experiments 1. Comparing context change detection performance of BOA $\mathrm{B_{AND}}$ (16) with different conjunction multiplicities N when the thresholds are selected by different algorithms

Table 2 Comparison of algorithms for BOA training

BOA	BOATS	IES	BAROC
B_{OR}	68.3	68.6	68.2
$B_{AND}, N = 1$	75.7	76.5	73.5
$\neg B_{\neg OR}$	76.3	73.2	73.2
$B_{AND}, N = 6$	81.1	77.3	73.3
$\neg B_{\neg \mathcal{P}}, N_q = 1$	81.3	73.4	73.7
$B_{\mathcal{P}}, N_q = 1$	80.0	81.8	80.5
$B_{\mathcal{P}}, N_q$ by BOATS	81.7		

Average test F_1-score over sixfold cross-validation sets in context change detection task with BOA combinations (17) trained with different algorithms. The used operating point of the BOA is the one with highest F_1-score on train data separately for each CV-fold

where the conjunction lists of $B_{\mathcal{P}}$ and $B_{\neg\mathcal{P}}$ are $z_1 = [1], z_2 = [2], z_3 = [3], z_4 = [1,2], z_5 = [1,3], z_6 = [2,3], z_7 = [1,2,3]$.

For the disjunctive BOA B_{OR}, the operating points found by the three algorithms are very similar. The IES algorithm finds the best operating point for this BOA with its exhaustive search. The proposed BOATS algorithm does not leave far behind, nor does the Boolean algebra for ROC curves. The assumption of the BAROC algorithm about the detector independence, which does not hold with these detectors, does not impair its performance in training the BOA B_{OR}, where only disjunctive **OR** -operator is used.

The conjunctive BOA B_{AND} with $N = 1$ and the disjunctive $\neg B_{\neg OR}$ have equally expressive decision boundaries. Ideally they would result in identical classifiers, but due to characteristics of the training algorithms they result in having different thresholds. Similarly the ideal decision boundaries of B_{AND} with $N = 6$ and $\neg B_{\neg\mathcal{P}}$ coincide. Results with those pairs of BOAs trained with the BOATS and BAROC algorithms are similar, which was expected because of the similarity of decision boundaries. The iterative exhaustive search does not find as good operating points for the BOA combinations when negative scores $-l_m$ are used. This is due to the selection of the threshold values to test, which in this case of using negative detector scores is done based on "non-target" class samples, as explained in Section 3.3.

For the BOA $B_{\mathcal{P}}$ with $N_q = 1, q = 1 \ldots Q$, the IES algorithm is able to find the best performing operating point. The iterative exhaustive search is thus effective in finding good thresholds for BOAs with different conjunctions. IES was not run with $N_q > 1$, because of its extremely long computation time for such a long BOA. BAROC algorithm finds a comparable operating point for the BOA $B_{\mathcal{P}}$ with $N_q = 1, q = 1 \ldots Q$. This is probably due to the abundance of different conjunctions in the BOA to be combined disjunctively, where the inaccurate independence assumption of BAROC does not matter so much. Also for the BAROC -algorithm, the result with the BOA $B_{\mathcal{P}}$ with $N_q > 1$ is not reported, because it is the same as with $N_q = 1$ by definition. The proposed BOATS algorithm leaves slightly behind IES and BAROC for the BOA $B_{\mathcal{P}}$ with $N_q = 1\,\forall q$. However, when the conjunction multiplicities N_q are unlimited, BOATS finds an operating point with similar performance with the best one with $N_q = 1\,\forall q$ by IES.

4.4 Computational efficiency of BOA

In this section, we report performance of different BOA cascades in terms of both F_1-score and the average computational load of classification in respect to real time processing. The BOA cascades are trained with the proposed BOATS algorithm for the video context change detection task which has highly unbalanced class distribution, the "no change" class being the prevalent one.

The BOA cascades B_{AND}, $\neg B_{\neg OR}$, and $\neg B_{\neg\mathcal{P}}$ of (17) correspond to one-sided cascades with early classification opportunity to the "no change" class. They are assumed to be computationally efficient with this data, where samples of "no change" form the large majority of data. The BOA cascades of B_{OR} and $B_{\mathcal{P}}$ are one-sided, having the early classification opportunity solely to the "target" class. They are likely to be slow with this data.

For this comparison we use, in addition to the BOA cascades of (17), symmetrical cascades realizing of

$$\begin{aligned} B_{C2} &= \left(l_1 \ge \theta_1^1\right) \vee \bigvee_{n=1..N} \left[\left(l_1 \ge \theta_1^{2,n}\right) \wedge \left(l_2 \ge \theta_2^{2,n}\right) \right] \\ B_{C3} &= \left(l_1 \ge \theta_1^1\right) \vee \bigvee_{n=1..N_2} \left[\left(l_1 \ge \theta_1^{2,n}\right) \wedge \left(l_2 \ge \theta_2^{2,n}\right) \right] \\ &\quad \vee \bigvee_{n=1..N_3} \left[\left(l_1 \ge \theta_1^{2,n}\right) \wedge \left(l_2 \ge \theta_2^{2,n}\right) \wedge \left(l_3 \ge \theta_3^{2,n}\right) \right]. \end{aligned} \tag{18}$$

The BOA cascade of B_{C2} is similar to the laughter detection cascade of Fig. 7 with $l_A = l_1$ and $l_V = l_2$. The cascade of B_{C3} with $N_2 = N_3 = 1$ is illustrated in Fig. 9.

In Table 3, we show for the different BOA cascades their best F_1-scores as well as their computation times (CT) of classification using a desktop PC in respect to real time processing. The individual detectors $d_m = (l_m \ge \theta_m)$, $m = 1, 2, 3$ have very different computational loads. Compared to real time processing, the detectors d_1 and d_2 are very fast, d_3 being extremely slow.

The BOA cascade of B_{C2} with $N = 1$ has a computational cost of only a fraction of real time, while achieving an outstanding improvement of classification performance over individual detectors $d_m = (l_m \ge \theta_m)$, $m = 1, 2, 3$. It requires a tiny fraction of the computational load of d_3 and less than 5% of the computational load of d_2 while only doubling the time of the fastest detector d_1.

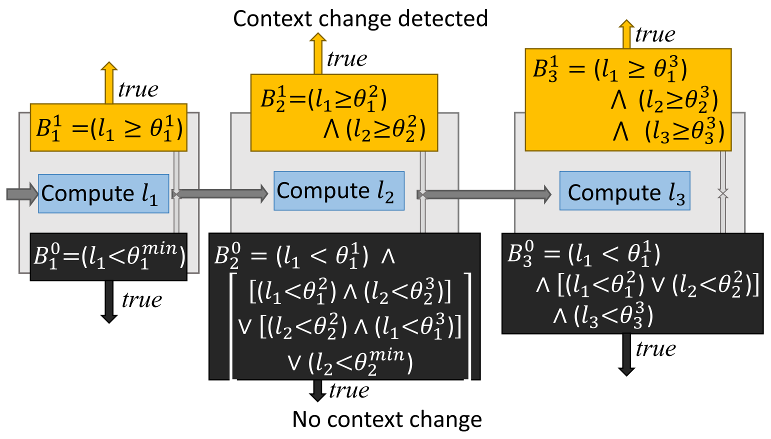

Context change detected
true
$B_1^1 = (l_1 \geq \theta_1^1)$
true
$B_2^1 = (l_1 \geq \theta_1^2) \wedge (l_2 \geq \theta_2^2)$
true
$B_3^1 = (l_1 \geq \theta_1^3) \wedge (l_2 \geq \theta_2^3) \wedge (l_3 \geq \theta_3^3)$
Compute l_1
Compute l_2
Compute l_3
$B_1^0 = (l_1 < \theta_1^{min})$
true
$B_2^0 = (l_1 < \theta_1^1) \wedge [[(l_1 < \theta_1^2) \wedge (l_2 < \theta_2^3)] \vee [(l_2 < \theta_2^2) \wedge (l_1 < \theta_1^3)] \vee (l_2 < \theta_2^{min})]$
true
$B_3^0 = (l_1 < \theta_1^1) \wedge [(l_1 < \theta_1^2) \vee (l_2 < \theta_2^2)] \wedge (l_3 < \theta_3^3)$
true
No context change

Fig. 9 BOA cascade example. The BOA cascade of B_{C3} with $N_1 = N_2 = N_3 = 1$. Threshold θ_m^{min} means $\theta_m^{min} = \min\left(\theta_m^2, \theta_m^3\right)$

At the same time it reaches $F_1 = .764$, which is about 9 percent units higher than .674 of d_1, 14 percent units higher than .525 of d_2 and 11 percent units higher than .553 of d_3. With N of B_{C2} not restricted, the F_1-score further improves to .778, but the computational benefit over always computing both l_1 and l_2 is lost.

The BOA cascade of B_{C3} utilizes all the three available detectors. Thus, the F_1-scores obtained with it are all the more improved from those obtained with B_{C2}. Real time processing is compromised by incorporating the extremely slow computation of l_3. However, with the cascade processing, the total computational load of B_{C3} with $N_2 = N_3 = 1$ is reduced to less than 2% of that of always computing all the scores l_1, l_2, and l_3. At the same time the F_1-score is improved to 0.774. With N_2 and N_3 unrestricted and selected by the proposed BOA training algorithm, F_1-score improves further to 0.813, the average computation time being still less than 4% of the time of always computing all the scores l_1, l_2 and l_3.

When observing the computational loads of different BOA cascades in Table 3, we may notice that remarkable computational savings appear whenever the BOA utilizes the computationally heaviest detector function $f_3(\boldsymbol{x}) = l_3$ only by combining it conjunctively with the faster detector functions $f_1(\boldsymbol{x}) = l_1$ and $f_2(\boldsymbol{x}) = l_2$. This is the case in BOA combinations B_{C2}, B_{C3}, B_{AND}, $\neg\mathrm{B}_{\neg OR}$ and $\neg\mathrm{B}_{\neg\mathcal{P}}$. The BOA cascades of B_{OR} and $\mathrm{B}_{\mathcal{P}}$ utilize a conjunction list $z_3 = [3]$, which means using the threshold comparison $\left(l_3 \geq \theta_3^3\right)$ as an individual conjunction within the BOA function. Because of this these BOA cascades can not avoid computing l_3 unless the input is accepted to the rare "context change detected" class by conjunctions using only scores l_1 and l_2. The BOA cascade of B_{OR} is computationally the most inefficient, as it is able to avoid computing l_3 only if the input is classified as "context change detected" by threshold comparison $(l_1 \geq \theta_1)$ or $(l_2 \geq \theta_2)$. The BOA cascade of $\mathrm{B}_{\mathcal{P}}$ is slightly more efficient due to its conjunctions $\bigvee_{n=1}^{N_q}\left(l_1 \geq \theta_1^{q,n}\right) \wedge \left(l_2 \geq \theta_2^{q,n}\right)$, based on conjunction list $z_q = [1, 2]$, capable of classifying the input as "context change detected" with only l_1 and l_2.

The best F_1-score, $F_1 = .814$, among the BOA cascades not utilizing a conjunction list $z_q = [3]$ is achieved with B_{AND}. Only slightly higher score, $F_1 = .817$, was obtained with BOA cascade $\mathrm{B}_{\mathcal{P}}$, but the computational efficiency obtainable with a cascade structure is obstructed by its computationally inefficient BOA design.

Precision vs recall curves of the detectors $d_1 = (l_1 \geq \theta)$, $d_2 = (l_2 \geq \theta)$, and $d_3 = (l_3 \geq \theta)$, and some BOA

Table 3 Performance comparison of different BOA cascades

BOA	F_1	CT
$d_1 = (l_1 \geq \theta_1)$	.674	0.01
$d_2 = (l_2 \geq \theta_2)$	.525	0.43
$d_3 = (l_3 \geq \theta_3)$	.553	184
$\mathrm{B}_{C2}, N_q = 1\ \forall q$	.764	0.02
B_{C2}, N_q by BOATS	.778	0.44
$\mathrm{B}_{C3}, N_q = 1\ \forall q$	.774	2.5
B_{C3}, N_q by BOATS	.813	6.9
$\neg\mathrm{B}_{\neg OR}$	.763	4.1
$\neg\mathrm{B}_{\neg\mathcal{P}}, N_q = 1\ \forall q$	.813	7.0
B_{AND}, N by BOATS	.814	8.5
B_{OR}	.683	182.0
$\mathrm{B}_{\mathcal{P}}, N_q = 1\ \forall q$	.798	153.3
$\mathrm{B}_{\mathcal{P}}, N_q$ by BOATS	.817	124.3

Results in terms of F_1-score and computation time (CT) in respect to video time in scene detection task with detectors d_1, d_2, d_3 and BOA combinations of (17) and (18). The BOA thresholds are selected with the proposed BOATS algorithm. The results are test averages over sixfold cross validation sets. The used operating point of each BOA is the one with highest F_1-score on train data separately for each CV-fold combinations of them trained with the BOATS algorithm are shown in Fig. 10. We can see that all the BOA combinations improve the precision-recall curve over the curves of the individual detectors remarkably.

5 Conclusions

We proposed to use a monotone Boolean function for combining multiple binary classifiers and showed how to implement it as a computationally efficient binary classification cascade. The proposed Boolean OR of ANDs (BOA) cascade is defined by a BF over multiple detector scores, and it is implemented as a classification cascade for computational efficiency. We also presented an algorithm, BOA threshold search (BOATS), for learning the thresholds of a BOA cascade.

We showed experimentally that the BOA cascade achieves the state-of-the-art performance in laughter detection task with MAHNOB laughter dataset while requiring much less computational power than the other solutions found in the literature. We also showed that the proposed algorithm suits best for learning thresholds of a BOA combination, compared to other learning strategies for Boolean combinations found in the literature. Finally, we explored the detection performance of different BOA cascades in terms of their F_1-scores and computational loads of detection. We showed that a BOA cascade improves the classification accuracy remarkably over the individual detectors while mostly requiring only a fraction of their combined computation time.

Endnote

[1] Demonstration available at http://arg.cs.tut.fi/demo/CASAbrowser/

Appendix 1

Reference algorithms for BOA training

The Algorithm 2 contains the functionality for training a Boolean BOA combination iteratively, by fusing two elements at a time. The symbol $\mathbf{\Theta}$ denotes a matrix of thresholds. Each row of $\mathbf{\Theta}$ contains one threshold setting for the corresponding Boolean classifier. The boldface symbols **tp** and **fp** are used to denote vectors of true positives and false positives resulting with different threshold settings in $\mathbf{\Theta}$ of a corresponding Boolean classifier, respectively.

One conjunction (q, n) of a BOA is built on lines 8–22 within the loop starting from line 7. On lines 23–28, the newly trained conjunction is combined with the conjunctions trained already.

The algorithm returns thresholds for found operating points α of the BOA in matrix $\mathbf{\Theta}_B$. The corresponding true positive rates and false positive rates on training data are returned in vectors $\mathbf{tp}_B$ $\mathbf{fp}_B$ We use this framework for training a BOA with either Boolean algebra of ROC curves (BAROC) by [10] or iterative exhaustive search (IES) by [11].

The training algorithm to be used is selected by a variable ALG. If ALG = IES, the combining is performed with Algorithm 3, and if ALG = BAROC, the combination of two sets of thresholds is done by Algorithm 4.

Appendix 2

Boolean decision makers at BOA cascade stages

At each stage $s = 1 \ldots S$ of a BOA cascade, one target likelihood score $f_m(\boldsymbol{x}) = l_m = l_s, m \in \{1 \ldots M\}$ is computed. All the scores $l_i, \;\; i = 1..s$ are thus available at cascade stage s to make the classification or the decision to enter the next cascade stage. BFs $B_1^1(l_1)$, $B_1^0(l_1)$, $B_2^1(l_1, l_2)$, $B_2^0(l_1, l_2)$,..., $B_S^1(l_1, l_2, \ldots, l_S)$ and $B_S^1(l_1, l_2, \ldots, l_S)$ are set to make these internal decisions of the cascade. As illustrated in Fig. 3, at each stage s, after computing the predefined target likelihood score l_s, a classification to the "target" class is made if $B_s^1(l_1, l_2, \ldots, l_s) = true$ and a classification to the "non-target" class is made if $B_s(l_1, l_2, \ldots, l_s)^0 = true$. If both the functions, B_s^0 and B_s^1, output *false*, the next cascade stage is entered. The functions B_S^0 and B_S^1 at the last cascade stage S are negations of each other ensuring the classification to be made.

The decision makers $B_s^1, \; s = 1 \ldots S$ are partitions of the BOA function (7), and the functions $B_s^0, \;\; s = 1 \ldots S$ are partitions of the negation (8) of the BOA function. This ensures that the decision makers $B_s^1, B_s^0, \;\; s = 1 \ldots S$ are

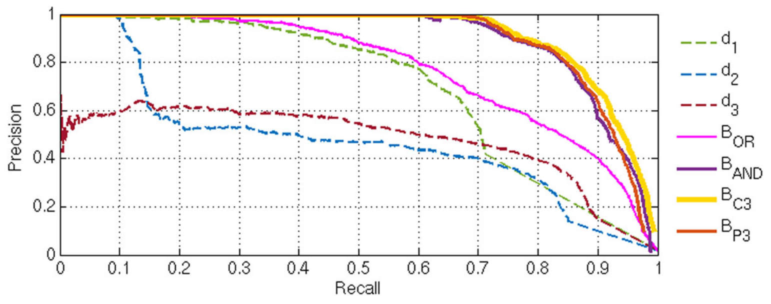


Fig. 10 P-R curves of BOA cascades. Precision vs. recall curves of detectors d_1, d_2, and d_3 and some BOA combinations of them

Algorithm 2 An algorithm to find thresholds for a BOA combination with IES or BAROC procedure

```
1:  procedure ITERATIVEBC
2:      input: $\mathcal{Z}$, N, , $\mathcal{F}$, $X$, ALG
            # $\mathcal{Z}$ contains $z_1, z_2, \ldots, z_Q$.
            # N contains $N_1, N_1, \ldots, N_Q$.
            # $\mathcal{F}$ contains the detector functions $f_m$, $m=1\ldots M$.
            # $X = X^0 \cup X^1$ contains the training data.
            # ALG is either IES or BAROC

3:      for $m = 1 \ldots M$ do
4:          Set $\vartheta_m = \{f_m(\boldsymbol{x}) \mid \boldsymbol{x} \in X^1\}$
5:      end for
6:      Set $B(\boldsymbol{x};\boldsymbol{\theta}) = false$, $\boldsymbol{\Theta}_B = \emptyset$, $\mathbf{tp}_B = 0$, $\mathbf{fp}_B = 0$
7:      for $q = 1 \ldots Q$ and $n = 1 \ldots N_q$ do
8:          Set $m = z_q(1)$
9:          Set $B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta}) = (f_m(\boldsymbol{x}) \geq \theta)$
10:         Set $\boldsymbol{\Theta}_{\text{conj}} = \vartheta_m$
11:         $\mathbf{tp}_{\text{conj}} = |\{ \boldsymbol{x} \mid \boldsymbol{x} \in X^1, B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\Theta}_{\text{conj}}) = true \}|$
12:         $\mathbf{fp}_{\text{conj}} = |\{ \boldsymbol{x} \mid \boldsymbol{x} \in X^1, B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\Theta}_{\text{conj}}) = false \}|$
13:         for $i = 2, 3, \ldots, M_q$ do
14:             Set $n = z_q(i)$
15:             Set $\boldsymbol{\Theta}_n = \vartheta_n$
16:             if ALG=IES then
17:                 Set new $\boldsymbol{\Theta}_{\text{conj}}$, $\mathbf{tp}_{\text{conj}}$, $\mathbf{fp}_{\text{conj}}$ according to
                      ES2($B_{\text{conj}}$, $\boldsymbol{\Theta}_{\text{conj}}$, $\wedge$, $(l_n \geq \theta)$, $\boldsymbol{\Theta}_n$, DATA)
18:             else if ALG = BAROC then
19:                 Set new $\boldsymbol{\Theta}_{\text{conj}}$, $\mathbf{tp}_{\text{conj}}$, $\mathbf{fp}_{\text{conj}}$ according to
                      BA2(($|X^1|\mathbf{tp}_{\text{conj}}$, $|X^0|\mathbf{fp}_{\text{conj}}$), $\boldsymbol{\Theta}_{\text{conj}}$, $\wedge$, $(|X^1|\mathbf{tp}_n, |X^0|\mathbf{fp}_n)$, $\boldsymbol{\Theta}_n$)
20:             end if
21:             Update $B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta}) = B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta}) \wedge (f_n(\boldsymbol{x}) \geq \theta)$
22:         end for
23:         if ALG=IES then
24:             Set new $\boldsymbol{\Theta}_B$, $\mathbf{tp}_B$, $\mathbf{fp}_B$ according to
                  ES2($B$, $\boldsymbol{\Theta}_B$, $\vee$, $B_{\text{conj}}$, $\boldsymbol{\Theta}_{\text{conj}}$, DATA)
25:         else if ALG=BAROC then
26:             Set new $\boldsymbol{\Theta}_B$, $\mathbf{tp}_B$, $\mathbf{fp}_B$ according to
                  BA2($(|X^1|\mathbf{tp}_B, |X^0|\mathbf{fp}_B)$, $\boldsymbol{\Theta}_B$, $\vee$, $(|X^1|\mathbf{tp}_{\text{conj}}, |X^0|\mathbf{fp}_{\text{conj}})$, $\boldsymbol{\Theta}_{\text{conj}}$)
27:         end if
28:         Update $B(\boldsymbol{x};\boldsymbol{\theta}) = B(\boldsymbol{x};\boldsymbol{\theta}) \vee B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta})$
29:     end for

30:     return: $\boldsymbol{\Theta}_B$, $\mathbf{tp}_B$, $\mathbf{fp}_B$.
31: end procedure
```

Algorithm 3 Exhaustive search of ROCCH operating points for a Boolean combination of two detectors

```
1:  procedure ES2
2:      input: $B_1$, $\boldsymbol{\Theta}_1$, $\star$, $B_2$, $\boldsymbol{\Theta}_2$, DATA
            # $\star$ is the Boolean operator to be used, either AND ($\wedge$) or OR ($\vee$)

3:      for $p = 1 \ldots$ number of sets $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}_1$ do
4:          for $q = 1 \ldots$ number of sets $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}_2$ do
5:              Set $\mathbf{tp}(p,q) = \left|\left\{ \boldsymbol{x} \mid \begin{smallmatrix} \boldsymbol{x} \in X^1 \\ B_1(\boldsymbol{x};\boldsymbol{\Theta}_1(p)) \star B_2(\boldsymbol{x};\boldsymbol{\Theta}_2(q)) = true \end{smallmatrix} \right\}\right|$
6:              Set $\mathbf{fp}(p,q) = \left|\left\{ \boldsymbol{x} \mid \begin{smallmatrix} \boldsymbol{x} \in X^0 \\ B_1(\boldsymbol{x};\boldsymbol{\Theta}_1(p)) \star B_2(\boldsymbol{x};\boldsymbol{\Theta}_2(q)) = true \end{smallmatrix} \right\}\right|$
7:          end for
8:      end for
9:      for $fp = 0 \ldots |X^0|$ do
10:         Find $(p^*, q^*) = \arg\max_{\mathbf{fp}(p,q)=fp} \mathbf{tp}(p,q)$
11:         Set $\mathbf{fp}_{\text{ROCCH}}(fp) = fp$
12:         Set $\mathbf{tp}_{\text{ROCCH}}(fp) = \mathbf{tp}(p^*, q^*)$
13:         Set $\boldsymbol{\Theta}_{\text{ROCCH}}(fp) = [\boldsymbol{\Theta}_1(p^*), \boldsymbol{\Theta}_2(q^*)]$
14:     end for

15:     return: $\boldsymbol{\Theta}_{ROCCH}$, $\mathbf{tp}_{\text{ROCCH}}$, $\mathbf{fp}_{\text{ROCCH}}$.
16: end procedure
```

consistent. This means that both B_s^1 and B_s^0 never output *true* concurrently, i.e. if $B_s^1(\boldsymbol{x}) = true$ then $B_s^0(\boldsymbol{x}) = false$ and similarly if $B_s^0(\boldsymbol{x}) = true$ then $B_s^1(\boldsymbol{x}) = false$. It also means that if classification is made by B_s^1 or B_s^0 at a cascade stage s, the decision makers B_r^1 and B_r^0 of the other stages $r = 1 \ldots S$, $r \neq s$ would not make contradicting classifications. Formally, if $\exists s\ B_s^c = true$ then $B_r^{\neg c} = false \quad \forall r \in 1 \ldots S$.

The internal decision makers at BOA cascade stages $s = 1 \ldots S$ for the "target" class are

$$B_s^1(\boldsymbol{x};\alpha) = \bigvee_{z_q \,\middle|\, \begin{smallmatrix} \exists j \quad s = z_q(j), \\ \nexists jm = z_q(j), m > s \end{smallmatrix}} \bigvee_{n=1}^{N_q} \bigwedge_{i=1}^{M_q} \left(f_{z_q(i)}(\boldsymbol{x}) \geq \theta_{z_q(i)}^{q,n} \right). \tag{19}$$

That is, B_s^1 contains the conjunctions (q,n) of the BOA (7) that utilize the newly computed likelihood score l_s and possibly those computed at earlier stages, but naturally none of the scores l_m, $m > s$. Examples can be seen in Figs. 7 and 9.

Similarly, the internal decision makers B_s^0, $s = 1 \ldots S$ of the BOA cascade for the "non-target" class are partitioned from the negated BOA; B_s^0 contains the conjunctions k of the BOA (8) that utilize the newly computed likelihood score l_s and possibly those computed at earlier stages, but none of the scores l_m, $m > s$. The partition of the

Algorithm 4 Combining ROC curves of two detectors using the Boolean algebra of ROC curves by [10]

```
1:  procedure BA2
2:      input: $(\mathbf{tpr}_1, \mathbf{fpr}_1)$, $\boldsymbol{\Theta}_1$, $\star$, $(\mathbf{tpr}_2, \mathbf{fpr}_2)$, $\boldsymbol{\Theta}_2$
        # $\star$ is the Boolean operator to be used, either AND ($\wedge$) or OR ($\vee$)

3:      for $fpr = 0 \ldots 1$ do
4:          if $\star = \wedge$ then
5:              Find $(p^*, q^*) = \arg \max_{\mathbf{fpr}_1(p)\cdot\mathbf{fpr}_2(q)=fpr} \mathbf{tpr}_1(p) \cdot \mathbf{tpr}_2(q)$
6:          else if $\star = \vee$ then
7:              Find $(p^*, q^*) = \arg \max_{\mathbf{fpr}_1(p)+\mathbf{fpr}_2(q)-\mathbf{fpr}_1(p)\cdot\mathbf{fpr}_2(q)=fpr} \mathbf{tpr}_1(p) + \mathbf{tpr}_2(q) - \mathbf{tpr}_1(p)\cdot\mathbf{tpr}_2(q)$
8:          end if
9:          Set $\mathbf{fpr}_{12}(fpr) = fpr$
10:         Set $\mathbf{tpr}_{12}(fpr) = \mathbf{tpr}_1(p^*) \cdot \mathbf{tpr}_2(q^*)$
11:         Set $\boldsymbol{\Theta}_{12}(fpr) = \{\boldsymbol{\Theta}_1(p^*), \boldsymbol{\Theta}_2(q^*)\}$
12:     end for

13:     return: $\boldsymbol{\Theta}_{12}$, $(\mathbf{tpr}_{12}, \mathbf{fpr}_{12})$.
14: end procedure
```

K conjunctions of $\neg$B of (8) is given by a Boolean variable $\mathbf{c}_s(k)$, which denotes whether the k:th conjunction of the negated BOA (8) is used for decision maker B_s^0. It is recursively defined as

$$\mathbf{c}_0(k) = false \quad \forall k = 1 \ldots K$$
$$\mathbf{c}_s(k) = \bigwedge_{r=1}^{s-1} \neg \mathbf{c}_r(k) \wedge \bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \bigwedge_{m=s+1}^{S} \neg \left[z_q(\mathcal{I}(k,q,n)) = m \right], \tag{20}$$

where $\mathcal{I}(k,q,n)$ is given by (9). The first part of the Eq. (20) makes sure that the conjunction k has not been used for B_r^0, $r < s$, while the rest of the equation checks whether detector functions beyond f_s, i.e., any of $f_{s+1}, f_{s+2}, \ldots, f_S$, are used in the conjunction k of (8) and sets $\mathbf{c}_s(k) = false$ if so.

Now, the decision-makers B_s^0 for the "non-target" class are

$$B_s^0 = \bigvee_{k \left| \begin{array}{l} k \in \{1 \ldots K\} \\ \mathbf{c}_s(k) = true \end{array} \right.} \left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \left(f_{z_q(\mathcal{I}(k,q,n))}(\boldsymbol{x}) < \theta^{q,n}_{z_q(\mathcal{I}(k,q,n))} \right) \right], \tag{21}$$

where the detector function indicator index $\mathcal{I}(k,q,n)$ is given by (9), $K = \prod_{q=1}^{Q} M_q^{N_q}$ and BOA variables z_q, N_q, for $q = 1 \ldots Q$ are adopted from (8). Using the alternative notation of the $\neg$B (8), the decision makers B_s^0, $s = 1 \ldots S$ for the "non-target" class may be written as

$$B_s^0(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{\substack{i_{1,1}=1\ldots M_1 \\ z_1(i_{1,1})\le s}} \bigvee_{\substack{i_{1,2}=1\ldots M_1 \\ z_1(i_{1,2})\le s}} \cdots \bigvee_{\substack{i_{1,N_1}=1\ldots M_1 \\ z_1(i_{1,N_1})\le s}} \bigvee_{\substack{i_{2,1}=1\ldots M_2 \\ z_2(i_{2,1})\le s}} \cdots \bigvee_{\substack{i_{2,N_2}=1\ldots M_2 \\ z_2(i_{2,N_2})\le s}} \bigvee_{\substack{i_{Q,1}=1\ldots M_Q \\ z_Q(i_{Q,1})\le s}} \cdots \bigvee_{\substack{i_{Q,N_Q}=1\ldots M_Q \\ z_Q(i_{Q,N_Q})\le s}} \left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \left(f_{z_q(i_{q,n})}(\boldsymbol{x}) < \theta^{q,n}_{z_q(i_{q,n})} \right) \right]. \tag{22}$$

This notation, while possibly being more comprehensible, includes all the decision makers B_r^0, $r < s$ in B_s^0, however this redundancy does not affect the functionality.

Abbreviations

BAROC: Boolean algebra of ROC curves; BF: Boolean function; BOA: **OR** of **AND**s function; BOATS: An algorithm to search thresholds for a BOA detector; CNF: Conjunctive normal form; CNF-BF: Boolean function in conjunctive normal form; CPU: Central processing unit; DNF: Disjunctive normal form; DNF-BF: Boolean function in disjunctive normal form; IBC: Iterative Boolean combination; IES: Iterative exhaustive search; LAD: Logical analysis of data; MFCC: Mel-frequency cepstral coefficient; OCAT: One clause at a time; RGB: Red-green-blue color format; ROC: Receiver operating characteristic curve; ROCCH: ROC convex hull; SIFT: Scale invariant feature transform

Acknowledgements

Discussions with Prof. Jiří Matas leveraged the research from implementing ad hoc ideas to doing excogitated research. The authors would also like to thank prof. Bhaskar Rao on useful discussions related to cascades and BFs. Furthermore, the authors want to thank the anonymous reviewers for their well-informed comments for improving the manuscript.

Funding

Funding for this research was provided by the Tampere University of Technology.

Availability of data and materials

Mahnob Laughter dataset is located at https://mahnob-db.eu/laughter/. CASA dataset is located at http://arg.cs.tut.fi/demo/CASAbrowser/.

Authors' contributions
KM has written the manuscript and implemented and executed the experiments. TV has been involved in designing the experiments as a supervisor and helped KM in writing the manuscript in a solid scientific way. JK gave the initial idea of utilizing Boolean functions for combining classifiers. He also provided his help in software related problems. All authors read and approved the final manuscript.

Ethics approval and consent to participate
People appearing in the videos of Mahnob Laughter dataset have given their consent for data usage for research purposes.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 2 May 2017 Accepted: 6 July 2018
Published online: 24 July 2018

References

1. S Yang, P Luo, C-C Loy, X Tang, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Wider face: A face detection benchmark, (2016)
2. S Zhang, R Benenson, M Omran, J Hosang, B Schiele, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. How far are we from solving pedestrian detection? (2016)
3. T Virtanen, A Mesaros, T Heittola, MD Plumbley, P Foster, E Benetos, M Lagrange, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. (Tampere University of Technology. Department of Signal Processing, 2016). ISBN (Electronic): 978-952-15-3807-0
4. J Ashbourn, *Biometrics: Advanced Identity Verification: the Complete Guide*. (Springer, 2014)
5. A Courbet, D Endy, E Renard, F Molina, J Bonnet, Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates. Sci. Transl. Med. **7**(289) (2015)
6. E Boros, PL Hammer, T Ibaraki, A Kogan, Logical analysis of numerical data. Math. Program. **79**(1), 163–190 (1997)
7. S Petridis, B Martinez, M Pantic, The MAHNOB laughter database. Image Vis. Comput. **31**(2), 186–202 (2013)
8. A Mesaros, T Heittola, A Eronen, T Virtanen, in *Signal Processing Conference, 2010 18th European*. Acoustic event detection in real life recordings (IEEE, 2010), pp. 1267–1271
9. J Daugman, *Biometric Decision Landscapes, vol. 482*. (University of Cambridge, Computer Laboratory, 2000)
10. ME Oxley, SN Thorsen, CM Schubert, in *Information Fusion, 2007 10th International Conference On*. A boolean algebra of receiver operating characteristic curves (IEEE, 2007), pp. 1–8
11. Q Tao, R Veldhuis, Threshold-optimized decision-level fusion and its application to biometrics. Pattern Recog. **42**(5), 823–836 (2009)
12. K Venkataramani, BV Kumar, in *Multimedia Content Representation, Classification and Security*. Role of statistical dependence between classifier scores in determining the best decision fusion rule for improved biometric verification (Springer, 2006), pp. 489–496
13. M Barreno, A Cardenas, JD Tygar, in *Advances in Neural Information Processing Systems 20*. Optimal roc curve for a combination of classifiers, (2008), pp. 57–64
14. W Khreich, E Granger, A Miri, R Sabourin, Iterative boolean combination of classifiers in the roc space: an application to anomaly detection with hmms. Pattern Recognit. **43**(8), 2732–2752 (2010)
15. E Granger, W Khreich, R Sabourin, Fusion of biometric systems using boolean combination: an application to iris-based authentication. Int. J. Biometrics. **4**(3), 291–315 (2012)
16. C Shen, On the principles of believe the positive and believe the negative for diagnosis using two continuous tests. J. Data Sci. **6**, 189–205 (2008)
17. Y Crama, PL Hammer, *Boolean Functions: Theory, Algorithms, and Applications. Encyclopedia of Mathematics and its Applications*. (Cambridge University Press, 2011)
18. G Alexe, S Alexe, TO Bonates, A Kogan, Logical analysis of data – the vision of peter l. hammer. Ann. Math. Artif. Intell. **49**(1), 265–312 (2007)
19. I Chikalov, V Lozin, I Lozina, M Moshkov, HS Nguyen, A Skowron, B Zielosko, *Logical Analysis of Data: Theory, Methodology and Applications*. (Springer, Berlin, 2013), pp. 147–192
20. PL Hammer, Partially defined boolean functions and cause-effect relationships. Lecture in International Conference on Multi-attribute Decision Making Via OR-based Expert Systems (1986)
21. RS Michalski. (RS Michalski, JG Carbonell, TM Mitchell, eds.) (Springer, Berlin, Heidelberg, 1983), pp. 83–134
22. AP Kamath, NK Karmarkar, KG Ramakrishnan, MGC Resende, A continuous approach to inductive inference. Math. Program. **57**(1), 215–238 (1992)
23. T Fawcett, An introduction to roc analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
24. P Hess, Dedekind's problem: monotone boolean functions on the lattice of divisors of an integer. Pacific J. Math. **81**(2), 411–415 (1979)
25. RS Michalski, in *V international Symposium on Information Processing (FCIP 69), Vol A3 (Switching Circuits)*. On the quasi-minimal solution of the general covering problem, (1969)
26. AS Deshpande, E Triantaphyllou, A greedy randomized adaptive search procedure (grasp) for inferring logical clauses from examples in polynomial time and some extensions. Math. Comput. Model. **27**(1), 75–99 (1998)
27. F Pawley, A Syder, The one-clause-at-a-time hypothesis. Perspect. Fluen, 163–199 (2000)
28. JR Quinlan, Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
29. JR Quinlan, *C4.5: Programs for Machine Learning*. (Morgan Kaufmann Publishers Inc., San Francisco, 1993)
30. L Breiman, JH Friedman, RA Olshen, CJ Stone, *Classification and Regression Trees*. (Chapman & Hall, New York, 1984)
31. PL Hammer, A Kogan, B Simeone, S Szedmák, Pareto-optimal patterns in logical analysis of data. Discrete Appl. Math. **144**(1-2), 79–102 (2004)
32. S Alexe, PL Hammer, Accelerated algorithm for pattern detection in logical analysis of data. Discret. Appl. Math. **154**(7), 1050–1063 (2006). Discrete Mathematics and Data Mining II (DM and DM II)
33. TO Bonates, PL Hammer, A Kogan, Maximum patterns in datasets. Discret. Appl. Math. **156**(6), 846–861 (2008). Discrete Mathematics and Data Mining II
34. RS Michalski, I Mozetic, J Hong, N Lavrac, in *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence. AAAI'86*. The multi-purpose incremental learning system aq15 and its testing application to three medical domains (AAAI Press, 1986), pp. 1041–1045
35. RE Reinke, in *Machine Intelligence 11*, ed. by JE Hayes, D Michie, and J Richards. Incremental Learning of Concept. Descriptions: A Method and. Experimental Results (Clarendon Press Oxford, 1988)
36. SN Sanchez, E Triantaphyllou, J Chen, TW Liao, An incremental learning algorithm for constructing boolean functions from positive and negative examples. Comput. Oper. Res. **29**(12), 1677–1700 (2002)
37. R Feraund, OJ Bernier, J-E Viallet, M Collobert, A fast and accurate face detector based on neural networks. IEEE Trans. Pattern Anal. Mach. Intell. **23**(1), 42–53 (2001)
38. P Viola, MJ Jones, Robust real-time face detection. Int. J. Comput. Vis. **57**(2) (2001)
39. L Lefakis, F Fleuret, in *NIPS*. Joint cascade optimization using a product of boosted classifiers, (2010)
40. MJ Saberian, N Vasconcelos, Learning optimal embedded cascades. IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 2005–2018 (2012)
41. C Shen, P Wang, S Paisitkriangkrai, A van den Hengel, Training effective node classifiers for cascade classification. Int. J. Comput. Vis. **103**, 326–347 (2013)
42. H Li, Z Lin, X Shen, J Brandt, G Hua, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. A convolutional neural network cascade for face detection, (2015), pp. 5325–5334
43. VC Raykar, B Krishnapuram, S Yu, in *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*. Designing efficient cascaded classifiers: tradeoff between accuracy and cost, (2010)
44. M Chen, Z Xu, KQ Weinberger, O Chapelle, D Kedem, in *AISTATS*. Classifier cascade for minimizing feature evaluation cost, (2012)

45. J Sochman, J Matas, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Waldboost - learning for time constrained sequential detection, (2005)
46. T Wu, S-C Zhu, in *ICCV*. Learning near-optimal cost-sensitive decision policy for object detection, (2013)
47. MM Dundar, J Bi, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Joint optimization of cascaded classifiers for computer aided detection, (2007)
48. C Zhang, P Viola, in *NIPS*. Multiple-instance pruning for learning efficient cascade detectors, (2007)
49. X Zhu, D Ramanan, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Face detection, pose estimation, and landmark localization in the wild, (2012), pp. 2879–2886
50. K Mahkonen, J-K Kämäräinen, T Virtanen, in *Computer Vision-ACCV 2014 Workshops*. Lifelog scene change detection using cascades of audio and video detectors (Springer, 2014), pp. 434–444
51. J Lankinen, J-K Kämäräinen, in *VISAPP (1)*. Video shot boundary detection using visual bag-of-words, (2013), pp. 788–791
52. R Research, C5.0. http://rulequest.com/download.html. Accessed 2018
53. O Rudovic, S Petridis, M Pantic, in *Proceedings of the 21st ACM International Conference on Multimedia*. Bimodal log-linear regression for fusion of audio and visual features (ACM, 2013), pp. 789–792
54. S Petridis, V Rajgarhia, M Pantic, Comparison of Single-model and Multiple-model Prediction-based Audiovisual Fusion, ISCA Speech Organisation (2015)
55. H Rao, Z Ye, Y Li, MA Clements, A Rozga, JM Rehg, in *Joint Conference on Facial Analysis, Animation and Audio-Visual Speech Processing (FAAVSP)*. Combining acoustic and visual features to detect laughter in adults' speech, (2015), pp. 153–156