Gated Classifiers: Boosting under High Intra-Class Variation

Oscar Danielsson, Babak Rasolzadeh and Stefan Carlsson
School of Computer Science and Communications
KTH, Stockholm, Sweden
{osda02,babak2,stefanc}@csc.kth.se

Abstract

In this paper we address the problem of using boosting (e.g. AdaBoost [7]) to classify a target class with significant *intra-class variation against a large background class. This situation occurs for example when we want to recognize a visual object class against all other image patches.*

The boosting algorithm produces a strong classifier, which is a linear combination of weak classifiers. We observe that we often have sets of weak classifiers that individually fire on many examples of the target class but never fire together on those examples (i.e. their outputs are anti-correlated on the target class). Motivated by this observation we suggest a family of derived weak classifiers, termed gated classifiers*, that suppress such combinations of weak classifiers. Gated classifiers can be used on top of any original weak learner.*

We run experiments on two popular datasets, showing that our method reduces the required number of weak classifiers by almost an order of magnitude, which in turn yields faster detectors. We experiment on synthetic data showing that gated classifiers enables more complex distributions to be represented. We hope that gated classifiers will extend the usefulness of boosted classifier cascades [29].

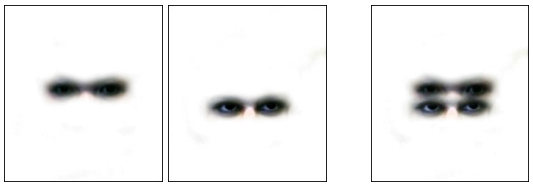
1. Introduction

Nonlinear classification of high dimensional data is a challenging problem. While designing such a classifier is difficult, boosting learning methods provide an effective stage-wise approach. Freund and Schapire showed that if weak classifiers can perform better than chance on every distribution over the training set, AdaBoost can provably achieve arbitrarily good generalization bound [7]. Perhaps the most demonstrating paper in applications of AdaBoost for detection discriminancy is still the famous paper by Viola and Jones, showing how AdaBoost can create the ideal building-blocks for a cascade of strong classifiers for the task of object detection [29].

However, despite the great success that AdaBoost (and its descendant algorithms) has seen in both theory and application, using AdaBoost to classify a target class with significant intra-class variation against a large background class remains a very challenging problem. For example, if viewpoint variation is introduced into the problem of face detection, the method of Viola and Jones can no longer be used out of the box. Figure 1 illustrates the problem. AdaBoost will learn a linear combination of weak classifiers that are common on the target class, resulting in a model that gets increasingly "blurry" with more intra-class variation. The problem is that eventually combinations of weak classifiers that never occur together on any example of the target class will generate false positives. For example, in figure 1(a), two different weak classifiers represent pairs of eyes in different locations and having two pairs of eyes gives higher face score than having only one pair of eyes.

In this paper we propose a way of directly addressing this problem within the AdaBoost framework. We term our solution *gated classifiers*. The main idea is that after a number of rounds of AdaBoost training, when we have such a "blurry" model of our target class, it would be instrumental to make available to the learner a derived family of weak classifiers that are able to suppress such combinations of already learnt weak classifiers that cause false positives. Gated classifiers are built by combining sets of already learnt weak classifiers using networks of logic gates and can be regarded as a singular (primitive) weak classifier within the boosting framework. Thus the conditions of the AdaBoost convergence proof [7] are not violated. On top of that, gated classifiers are easy to implement and will only require small changes to any existing AdaBoost code. We will argue in this paper that gated classifiers increase the ability of the strong classifier to handle intra-class variation while keeping it compact and efficient and with a minimal increase in the risk of over-fitting.

There are of course many other ways of dealing with intra-class variation. Some examples include: (1) training separate detectors for different subsets of the target class or training detector pyramids [15], (2) constructing image features that better separate the target class from the back-



(a) Two weak face classifiers and their linear combination

(b) Two weak cup classifiers and their linear combination

Figure 1. *Representing an object class as a linear combination of weak classifiers may give high scores to combinations of weak classifiers that never occur together. We use gated classifiers to suppress combinations of weak classifiers that are anti-correlated on the target class.*

ground [12, 2, 19] and (3) improving the weak classifiers (e.g. using decision trees instead of stumps or selecting better thresholds) [24]. Gated classifiers can be used together with any of these other strategies.

In the next section we will review more related work. The rest of this paper is structured as follows: in section 4 we motivate and define gated classifiers. We also describe how they are learnt. In section 5 we show experimentally that gated classifiers reduce the number of weak classifiers needed to solve a given classification task by almost an order of magnitude. In section 6 we discuss gated classifiers, their relation to other methods and future work. Finally, we sum up in section 7.

2. Related Work

Boosting, being the most successful ensemble learning method, has received substantial attention from the machine learning community. Freund and Schapire introduced the AdaBoost algorithm [7]. Schapire and Singer then proposed RealBoost, which is a generalization of AdaBoost where each weak hypothesis generates not only predicted classifications, but also self-rated confidence scores which estimate the reliability of its predictions [5]. Friedman et. al. explained boosting in terms of additive logistic regression and proposed for example GentleBoost and LogitBoost [8].

Numerous variants of the AdaBoost algorithm have appeared in the past decade. Some examples include FloatBoost [14], BrownBoost [6, 4], regularized AdaBoost [25], WeightBoost [11], EntBoost [13], KLBoost [16], Jensen-Shannon Boost [10] and WaldBoost [26]. Some versions of boosting address issues with applications of boosting in specific cases. For instance one limitation of AdaBoost arises in the context of skewed example distributions and cascaded classifiers: AdaBoost minimizes a quantity related to classification error rather than the number of false negatives, which typically has a hard upper bound when training a cascaded classifier. To address this problem Viola and Jones introduced AsymmBoost [28], which gives higher penalties for false negatives than does standard AdaBoost.

However, despite this immense body of literature on boosting, we have only found a few works aiming at a generic treatment of the problem of high intra-class variation (these are mentioned below). It seems like the common approach is instead to deal with intra-class variation individually for each specific classification task by including tailored features. For example Corso [2] adds a new class of hierarchical, adaptive features into boosting-based discriminative models. In the same spirit Laptev move from Haar features to HOG features [12] and Opelt et. al. use boundary fragments [19].

One generic way of handling intra-class variation is to exploit dependencies in the data (i.e. different feature values will be co-dependent). The most common example is probably to use decision trees instead of decision stumps as weak classifiers. In [20] the authors use relative spaces to deal with the problem of large intra-class variation, and boost a set of Fischer Linear Discriminant weak classifiers to a final strong classifier. This way they claim to be able to handle co-dependencies by partitioning the training data into sub-categories. Mita et al. [17] quite explicitly utilize Haar feature co-occurrences in order to extract structural similarities. A very similar approach is taken by Yamauchi et al. [30], with the increment that they have co-occurrence probability features instead of binary features, i.e. they utilize RealBoost instead of AdaBoost for the boosting part. In [18] authors create joint-features handling co-occurrences with two-stage boosting; multiple low-level HOG features are combined by using RealBoost.

These methods essentially capture high-order feature occurrence statistics (typically co-occurrences) and are generic in the sense that they can be used with any feature set. However, they all use sets of co-occurring features to build weak classifiers of increasing specificity. In the limit this paradigm approaches template matching, which is known to have problems with detection efficiency and generalization (over-fitting).

In contrast, our gated classifiers have a completely different way of handling intra-class variation. Instead of combining features to build increasingly specific weak classifiers, we combine already learnt weak classifiers that are anti-correlated to construct new weak classifiers that are optimal at correcting the errors (false positives) of the current strong classifier. We argue that we thus increase the ability of the classifier to handle intra-class variation while maintaining a compact and efficient classifier and a low risk of over-fitting.

3. AdaBoost

To define the basic notation we describe the AdaBoost algorithm, closely following Freund and Schapire [7], in listing 1. The input to the learner is a set of training examples $\{(\mathbf{x_i}, y_i)|i \in \mathcal{I}\}$, where $\mathbf{x_i} \in \mathcal{X}$ is a feature vector and $y_i \in \mathcal{Y} = \{0, 1\}$ is the corresponding label. The algorithm maintains an importance distribution over the training examples and $D_t(i)$ denotes the weight on the ith example at round t. The classification function of the strong classifier, $H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, is a linear combination of the classification functions of the weak classifiers. The classification function is thresholded to determine class membership.

Algorithm 1 AdaBoost

Require: $\{(\mathbf{x_i}, y_i)|i \in \mathcal{I}\}, T$
  Initialize $D_1(i) = 1/|\mathcal{I}|$
  for $t = 1$ to T **do**
    Train $h_t : \mathcal{X} \rightarrow \mathcal{Y}$ using distribution $\mathbf{D_t}$
    Choose $\alpha_t \in \mathbb{R}$
    $\mathbf{D_{t+1}} \leftarrow$ update weight distribution
  end for
  return $\{\alpha_1, \ldots, \alpha_T\}, \{h_1, \ldots, h_T\}$

The goal of the weak classifier h_t is to minimize the weighted classification error, ϵ_t:

$$\epsilon_t = P_{i \sim \mathbf{D_t}} \left(h_t(\mathbf{x_i}) \neq y_i \right) = \sum_{\{i \in \mathcal{I} | h_t(x_i) \neq y_i\}} D_t(i) \qquad (1)$$

In practice this typically boils down to: (1) defining a large pool of candidate classifiers, (2) evaluating ϵ_t for each candidate classifier in the pool and (3) selecting the one that yields the smallest error. Candidate classifiers are typically "simple", i.e. they are defined by very few parameters and have very limited representational flexibility. This decreases the risk of over-fitting. A common choice of classifier is the decision stump (a decision tree with unit depth).

The strong classifier captures first-order statistics of weak classifier responses. In cases where the target class exhibits significant intra-class variation and the background class "interferes" with the target class in the relevant feature space, first-order statistics may not be sufficient to separate the target class from the background. We will give an example of this in the next section and suggest a simple solution.

4. Gated Classifiers

In this section we will use a toy example to motivate why first-order weak classifier occurrence statistics may not be sufficient to capture target classes with significant intra-class variation. We then suggest a solution, which we term gated classifiers. Gated classifiers extend any existing pool of candidate classifiers and capture high-order occurrence statistics of previously learnt weak classifiers. In addition, they are easily implemented in hardware.

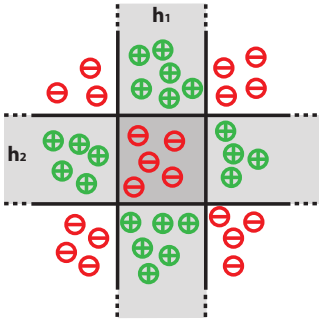


Figure 2. *A motivating example. See section 4.1 for description.*

4.1. A Motivating Example

In figure 2 we have a number of training examples represented in a 2D feature space. We also have a pool of two weak classifiers, h_1 and h_2, each having a region in feature space which it classifies as positive. We see that there is no way to linearly combine h_1 and h_2 so that the negative examples in the center and the positive examples get correctly classified. However, h_1 and h_2 are strongly correlated on the negative examples, while being anti-correlated on the positive examples. It would thus make sense to add a new weak classifier that classifies an example as negative if it activates both h_1 and h_2 and as positive otherwise. We define the new classifier as $h_3(\mathbf{x}) = \neg (h_1(\mathbf{x}) \wedge h_2(\mathbf{x}))$; h_3 captures second-order weak classifier occurrence statistics.

This type of situation arises whenever we have two weak classifiers that are likely to occur on an object of the target class but never occur together. Some examples include: (1) the arm of a pedestrian may be raised or lowered but not both at the same time, (2) the handle of a mug may be round or square but not both and (3) a face may be brighter or darker than the background but not both.

4.2. Definition

In this section we will define three different types of gated classifiers, which we have used in our experiments. We stress, however, that other types of gated classifiers can be defined similarly and that networks of gated classifiers can easily be constructed.

NAND-classifier In the previous section we constructed a weak classifier that suppressed examples where a pair of previously learnt weak classifiers co-occurred. This is an example of a NAND-classifier. However, there is no reason to restrict the definition to pairs. We could also suppress examples where some larger subset of previously learnt weak

classifiers co-occur. We thus define a NAND-classifier to classify as negative all examples where some subset of previously learnt weak classifiers co-occur:

$$h_{NAND}(\mathbf{x}) = \neg\left(h_{j_1}(\mathbf{x}) \wedge \ldots \wedge h_{j_n}(\mathbf{x})\right) \quad (2)$$

, where $j_1, \ldots, j_n \in \{1, \ldots, t-1\}$ and t is the current round of boosting.

XOR-classifier The XOR-classifier is strongly related to the NAND-classifier and simply adds a requirement that at least one of the classifiers $h_{j_1}, \ldots, h_{j_n}$ is activated.

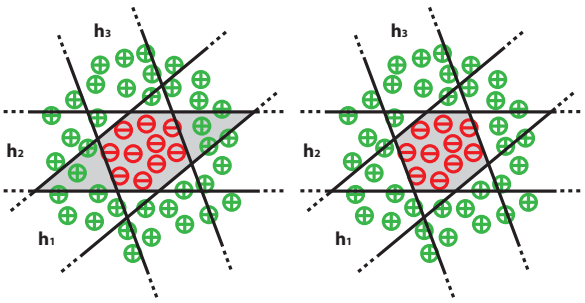
$$h_{XOR}(\mathbf{x}) = \begin{array}{l} \neg\left(h_{j_1}(\mathbf{x}) \wedge \ldots \wedge h_{j_n}(\mathbf{x})\right) \\ \wedge\left(h_{j_1}(\mathbf{x}) \vee \ldots \vee h_{j_n}(\mathbf{x})\right) \end{array} \quad (3)$$

Tautology and falsum We also define two trivial classifiers: the tautology classifier $h_\tau(\mathbf{x}) = 1$ and the falsum classifier $h_\phi(\mathbf{x}) = 0$. While these classifiers do not affect the contrast between the values of the classification function for positive and negative examples, they do play a role in the training phase. If the weight distribution gets too uneven toward either positive of negative examples, one of these classifiers will get selected and then examples will get reweighted so that unevenness is reduced. The classifier is not saved during training since it does not affect the classification performance.

4.3. Learning

In this section we turn our attention to learning a NAND- or XOR-classifier. These classifiers are constructed by combining a subset of the weak classifiers that were already selected (i.e. at round t of boosting we have $t-1$ weak classifiers to select from). The most direct approach to selecting the optimal classifier would be to evaluate all possible such classifiers and select the best one. However, there are $2^{t-1} - (t-1) - 1 = 2^{t-1} - t$ possibilities (all subsets of cardinality ≥ 2 of previously learnt weak classifiers). So exhaustive enumeration will in general be impossible. A reasonable strategy is then to enumerate all small gated classifiers, i.e. enumerate all subsets containing $\leq n$ weak classifiers (where n is chosen so that exhaustive enumeration is possible).

We can also incrementally extend a given NAND- or XOR-classifier. Take a NAND-classifier as an example (the XOR-classifier case is analogous). Refer to figure 3 and assume that we have classifier $h_a(\mathbf{x}) = \neg\left(h_{j_1} \wedge \ldots \wedge h_{j_n}\right)$ and let $\mathcal{I}_a^- = \{i \in \mathcal{I} | h_a(\mathbf{x_i}) = 0\}$ be the set of all training examples that are classified as negative by h_a. Then let $h_b(\mathbf{x}) = \neg\left(h_{j_1} \wedge \ldots \wedge h_{j_n} \wedge h_{j_{n+1}}\right)$ be the classifier we get by appending $h_{j_{n+1}}$ to h_a and let $\mathcal{I}_b^- = \{i \in \mathcal{I} | h_b(\mathbf{x_i}) = 0\}$ be the set of all training examples that



(a) $h_a(\mathbf{x}) = \neg\left(h_1 \wedge h_2\right)$ (b) $h_b(\mathbf{x}) = \neg\left(h_1 \wedge h_2 \wedge h_3\right)$

Figure 3. *Incremental extension of a NAND-classifier. The gray region is classified as negative.*

are classified as negative by h_b. Obviously $\mathcal{I}_b^- \subseteq \mathcal{I}_a^-$. We can now define the set of training examples that were classified as negative by h_a but are classified as positive by h_b: $\Delta\mathcal{I}^+ = \mathcal{I}_a^- \setminus \mathcal{I}_b^- = \left\{i \in \mathcal{I}_a^- | h_{j_{n+1}}(\mathbf{x_i}) = 0\right\}$. All positive examples in this set will now get correctly classified while the negative examples will get incorrectly classified. The classification error thus changes as follows:

$$\Delta\epsilon = \sum_{\{i|\mathbf{x_i} \in \Delta\mathcal{I}^+, y_i = 0\}} D_t(i) \quad - \sum_{\{i|\mathbf{x_i} \in \Delta\mathcal{I}^+, y_i = 1\}} D_t(i) \quad (4)$$

We can now define a simple greedy algorithm for learning large NAND- and XOR-classifiers. The first step of that algorithm is to exhaustively enumerate all small NAND- and XOR-classifiers, evaluate the classification error of each and select the best. We then loop through all weak classifiers not already used and append the one that yields the smallest $\Delta\epsilon$. We continue appending more weak classifiers greedily as long as the best $\Delta\epsilon$ is negative. At each step, we only need to evaluate the new classifier on the examples that were classified as negative by the previous classifier. The size of this set will be monotonically decreasing.

In listing 2 we give the whole AdaBoost algorithm including gated classifiers (compare to listing 1).

Our greedy weak learner will add a complexity term of $O(t^N)$ at round t of boosting (where N is the maximum size of the NAND or XOR classifier). Typically we set $N = 2$ or $N = 3$. At test time the input to the gated classifier will be the outputs of other weak classifiers that have already been evaluated. So the only computation required to evaluate the gated classifier is the NAND or XOR operation itself.

Gate Networks We note that the learning algorithm may generate gated classifiers consisting of more than one gate. This happens when a gated classifier is constructed using at least one other gated classifier. This is illustrated in figure 4.

Algorithm 2 AdaBoost with gated classifiers

Require: $\{(\mathbf{x_i}, y_i)|i \in \mathcal{I}\}, T$
  Initialize $D_1(i) = 1/|\mathcal{I}|$
  for $t = 1$ to T **do**
    Train $h_t : \mathcal{X} \rightarrow \mathcal{Y}$ using distribution $\mathbf{D_t}$
    Train $h_{NAND} = \neg \bigwedge_{j \in \mathcal{J}_1} h_j : \mathcal{X} \rightarrow \mathcal{Y}$
      where $\mathcal{J}_1 \subseteq \{1, \ldots, t-1\}$
    Train $h_{XOR} = \neg \bigwedge_{j \in \mathcal{J}_2} h_j \wedge \bigvee_{j \in \mathcal{J}_2} h_j : \mathcal{X} \rightarrow \mathcal{Y}$
      where $\mathcal{J}_2 \subseteq \{1, \ldots, t-1\}$
    $h_t^* \leftarrow$ *the best of* $h_t, h_\tau, h_\phi, h_{NAND}$ *and* h_{XOR}
    Choose $\alpha_t \in \mathbb{R}$
    $\mathbf{D_{t+1}} \leftarrow$ update weight distribution
  end for
  return $\{\alpha_1, \ldots, \alpha_T\}, \{h_1^*, \ldots, h_T^*\}$

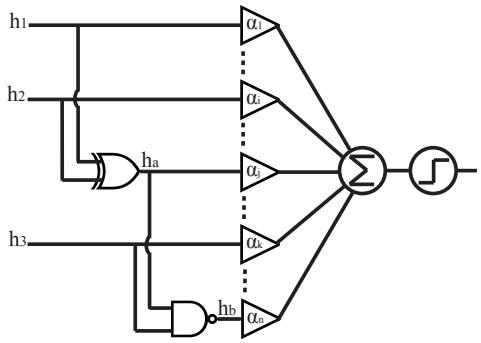


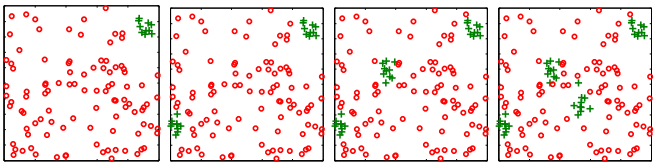
Figure 4. *When boosting with gated classifiers, increasingly complex gate networks may be constructed. In this example* h_1, h_2 *and* h_3 *are basic weak classifiers, while* h_a *and* h_b *are gated classifiers.* h_b *uses* h_a *as an input and is thus a compound gated classifier.* *(The final classification function is in this case* $H(\mathbf{x}) = \alpha_1 \cdot h_1(\mathbf{x}) + \alpha_i \cdot h_2(\mathbf{x}) + \alpha_j \cdot h_a(\mathbf{x}) + \alpha_k \cdot h_3(\mathbf{x}) + \alpha_n \cdot h_b(\mathbf{x}) + \ldots$, *where* $h_a(\mathbf{x}) = h_1(\mathbf{x}) \oplus h_2(\mathbf{x})$ *and* $h_b(\mathbf{x}) = \neg(h_a(\mathbf{x}) \wedge h_3(\mathbf{x}))$*).*

5. Experiments

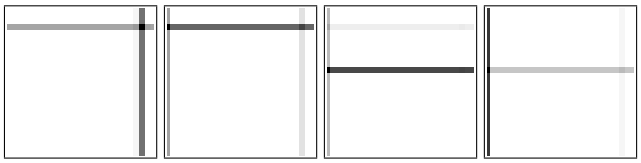
5.1. Experiment 1 - Synthetic data

The goal of the first experiment is to visualize the effects of using gated classifiers. We have generated a sequence of synthetic datasets, containing positive and negative examples represented by 2D vectors. The positive examples are drawn from a gaussian mixture distribution and the negative examples are drawn from a uniform background distribution. We generate a sequence of increasingly difficult datasets by adding more components to the gaussian mixture. Scatterplots of the first four datasets in the sequence are shown in figure 5(a).

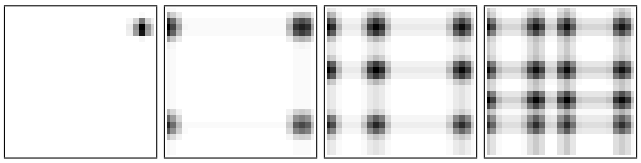
We defined a pool of weak classifiers by splitting the x- and y-axes into a set of intervals. Each interval X_c on the x-axis corresponds to a weak classifier $h_c(\mathbf{x}) = \mathbf{x}(1) \in X_c$ and similarly each interval Y_r on the y-axis corresponds to a weak classifier $h_r(\mathbf{x}) = \mathbf{x}(2) \in Y_r$.



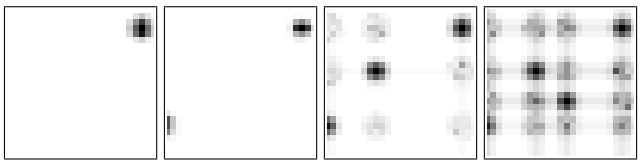
(a) Scatter plots of the synthetic datasets. Positive examples are green +-signs.



(b) Heat map of AdaBoost classification function



(c) Tautology and falsum classifiers added

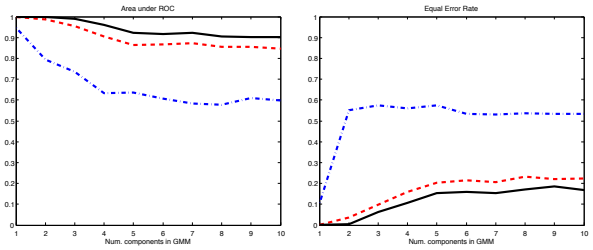


(d) NAND- and XOR-classifiers added

Figure 5. *In a) we show scatter plots of the first three datasets in the synthetic sequence. In b)-d) we show heat maps of the learnt classification functions using b) standard AdaBoost, c) adding tautology and falsum classifers and d) also adding NAND- and XOR-classifiers.*

For each dataset in the sequence we apply three different learning algorithms: (1) standard AdaBoost, (2) standard AdaBoost with tautology and falsum classifiers and (3) standard AdaBoost with tautology, falsum, NAND- and XOR-classifiers. In figures 5(b)-5(d) we show heat maps of the classification functions generated by each of the three learners. In figure 5(b), we see that standard AdaBoost stops after only a few rounds because no weak classifier with error better than chance can be found. Adding the tautology and falsum classifiers improves this situation and learning continues until a better representation of the true density is found (figure 5(c)). However we are still not able to suppress the clusters of false positives in the "intersections" of common (but mutually exclusive) weak classifiers. In figure 5(d) we see that adding NAND- and XOR-classifiers gives us this ability.

In figure 6 we plot the area under the ROC curve and the equal error rate for the three different learners on the sequence of datasets. Half of the dataset was used for training and the other half for validation.



(a) Area under ROC curve (AUC) (b) Equal error rate (EER)

Figure 6. *This figure shows plots of a) AUC and b) EER for the three different leaners: (1) standard AdaBoost (blue dash dotted curve), (2) std. AdaBoost with tautology and falsum classifiers (red dashed curve) and (3) std. AdaBoost with tautology, falsum, NAND- and XOR-calssifiers (black curve). The x-axis represents the number of components in the gaussian mixture used to generate positive examples.*

5.2. Experiment 2 - Pedestrians

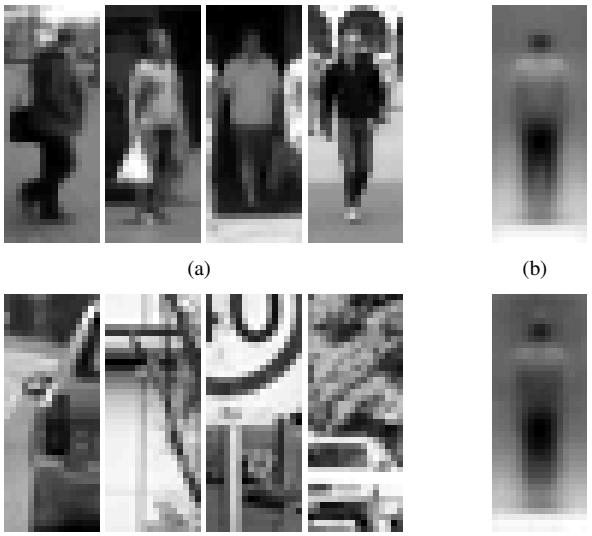
In this experiment we evaluate the use of gated classifiers in a realistic application: pedestrian detection. We used the NICTA pedestrian dataset [21], which is divided into several subsets. In our experiments we used only training set A, which was split in half for training and testing. Hard negative examples were found by (1) training a few stages of a cascaded classifier, (2) running that over a set of negative images and (3) collecting all false positive detections. In figure 7 we show a few training examples and average images for the positive and negative sets.

We then train an AdaBoost classifier with and without using gated classifiers and at each round of boosting we plot the false positive rate (fpr) at a detection rate (dr) of 0.9 on the test set. The result is shown in figure 8(a). We can se that the fpr drops much faster when gated classifiers are used. In this case a fpr of 0.5 is reached after 291 iterations, while standard AdaBoost requires 2667 iterations. In figure 8(b) we show the ROC curves of both classifiers after 291 iterations.

5.3. Experiment 3 - Faces

In this experiment we evaluate the use of gated classifiers for face detection, using the the Feret dataset [23, 22]. We used the same procedure as in the previous experiment, setting half of the training examples aside for testing and mining for hard negative examples by training a few stages of a cascaded classifier and collecting false positive detections from a set of background images. In figure 9 we show a few training examples and average images for the positive and negative sets.

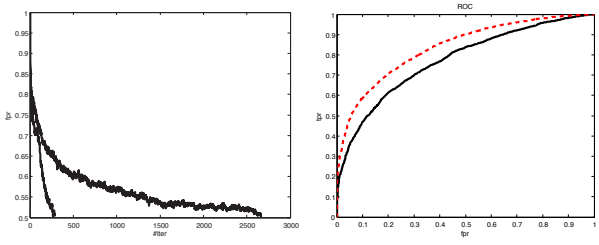
As in the previous experiment we then train an AdaBoost classifier with and without using gated classifiers and at each round of boosting we plot the false positive rate (fpr) at a detection rate (dr) of 0.9 on the test set. The results of this experiment are shown in figure 10(a). We can se that,



(a) (b)

(c) (d)

Figure 7. *Illustration of the NICTA pedestrian dataset, showing a) four positive examples, b) the average of all positive examples, c) four hard negative examples and d) the average of all hard negative examples. Hard negative examples were selected by training a few stages of a classifier cascade and collecting false positives.*



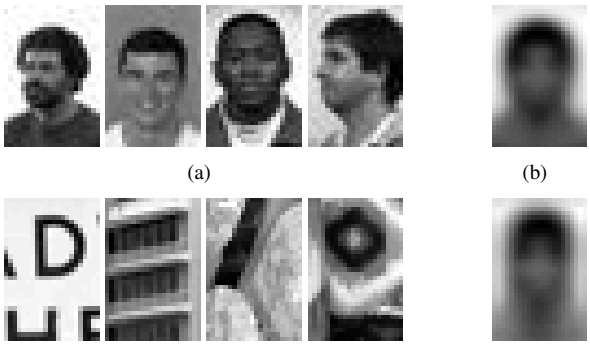
(a) Fpr vs. number of iterations (b) ROC curves @ 291 iterations

Figure 8. *Results of using gated classifiers for pedestrian detection. a) Fpr on validation set vs. number of AdaBoost iterations. The lower curve uses gated classifiers. b) ROC curve of both classifiers at 291 iterations. Upper (red, dashed) curve uses gated classifiers.*

again, the fpr drops much faster when gated classifiers are used. In this case a fpr of 0.5 is reached after 44 iterations, while standard AdaBoost requires 358 iterations. In figure 10(b) we show the ROC curves of both classifiers after 44 iterations.

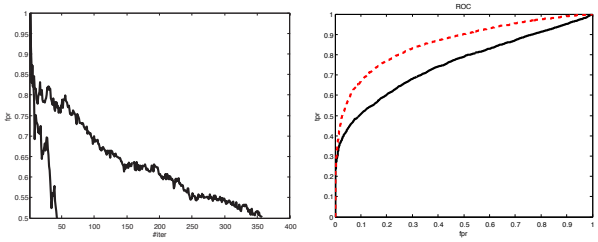
6. Discussion and Future Work

In learning situations when the target class has high intra-class variation and positive and negative examples are "mixed" in the feature space, the decision boundary required to solve the problem is complex. In the previous section we demonstrated that such decision boundaries can be impossible (experiment 1) or at least very difficult (experiments 2 and 3) to generate by boosting a "simple" weak



(a) (b) (c) (d)

Figure 9. *Illustration of the Feret face dataset, showing a) four positive examples, b) the average of all positive examples, c) four hard negative examples and d) the average of all hard negative examples. Hard negative examples were selected by training a few stages of a classifier cascade and collecting false positives.*



(a) Fpr vs. number of iterations (b) ROC curves @ 44 iterations

Figure 10. *Results of using gated classifiers for face detection. a) Fpr on validation set vs. number of AdaBoost iterations. The lower curve uses gated classifiers. b) ROC curve of both classifiers at 44 iterations. Upper (red, dashed) curve uses gated classifiers.*

learner. When we added gated classifiers, the number of weak classifiers required to reach a specific false positive rate and detection rate on the validation set decreased by almost and order of magnitude for both pedestrians and faces. This clearly demonstrates the usefulness of gated classifiers. With gated classifiers the strong classifier remains compact and efficient even in the presence of high intra-class variation.

The VC-dimension when using gated classifiers By adding gated classifiers to our weak learner we make it more flexible and thus we expose ourselves to the risk of over-fitting. Luckily, our experiments indicate that over-fitting does not occur. In addition we can also compute a theoretical upper bound for the VC-dimension of a gated classifier. This can be used to bound the generalization error when using gated classifiers [7, 27].

To compute an upper bound for the VC-dimension when using gated classifiers we apply Theorem 1 of Baum and Haussler [1]. We can view the gated classifier as a two-layer feed-forward network where the computation units of the first layer are the two basic weak classifiers and the computation unit of the second layer is the logical gate. Let $\mathcal{H}$ be the class of binary functions that can be generated by our basic weak learner (not using gated classifiers) and let $d_{\mathcal{H}} = VCdim(\mathcal{H})$. Then let $\mathcal{F} = \{NAND, XOR\}$. By inspection of the truth-tables of NAND and XOR we see that $d_{\mathcal{F}} = VCdim(\mathcal{F}) = 1$. Thus the sum over all computation units of the VC-dimensions of the classes of functions associated with each unit is $d = 2 \cdot d_{\mathcal{H}} + d_{\mathcal{F}} = 2 \cdot d_{\mathcal{H}} + 1$. Finally, we let $\Theta(\mathcal{H}) = \{f(h_1, h_2) | f \in \mathcal{F} \wedge h_1, h_2 \in \mathcal{H}\}$. Baum and Haussler's Theorem 1 [1] implies that the number of different functions that can be realized by $h_g \in \Theta(\mathcal{H})$ when the domain is restricted to a set of size m is at most $(3 \cdot e \cdot m/d)^d$. If $m \geq 5.5 \cdot d$, then $(3 \cdot e \cdot m/d)^d < 2^m$, which implies that the VC-dimension is smaller than m. Thus an upper bound for the VC-dimension when using gated classifiers is $\lfloor 5.5 \cdot d \rfloor = \lfloor 11 \cdot d_{\mathcal{H}} + 5.5 \rfloor$.

Note that in the discussion above we let the learner of gated classifiers select any pair of basic weak classifiers from $\mathcal{H}$, but we actually only search over pairs of already learnt weak classifiers so the true VC-dimension should be significantly smaller than the upper bound.

Alternatives An alternative method for generically handling high intra-class variation would be to build more specific weak classifiers [17, 30, 18]. In the limit we could have one weak classifier for each positive training example (template matching - i.e. each weak classifier has zero false positive rate and ϵ detection rate). One problem with this sort of approach is that the number of weak classifiers required to represent the whole class is extremely large, making efficient detection a tough challenge [9]. More importantly, there is also a big risk that even a very large training set does not represent the whole target class (over-fitting).

Hardware implementation Another advantage of gated classifiers is that they lend themselves readily for hardware implementation, for example using FPGAs. This could be useful for implementing the algorithm onboard cameras and other ubiquitous equipment.

Future work In future work we will make a more thorough evaluation of generic gated classifiers and also experiment with different feature pools, like HOG-features [3]. It will also be interesting to see if gated classifiers will increase the applicability of AdaBoost and to further explore the theoretical properties of gated classifiers.

7. Conclusion

In this paper we proposed gated classifiers to solve the problem of using boosting to learn target classes with large intra-class variation. We showed experimentally that gated

classifiers reduces the number of weak classifiers required to represent a complex decision boundary by almost an order of magnitude. Gated classifiers are easy to implement and can be used to empower any basic weak learner. In addition, gated classifiers can easily be implemented in hardware, for example using FPGAs. We therefore believe that gated classifiers may find many uses in the computer vision industry.

Acknowledgements This work has been funded by the Swedish Foundation for Strategic Research (SSF); within the project VINST and the Centre for Autonomous Systems. B. Rasolzadeh also acknowledge support from the computer vision company OculusAI (www.oculusai.com). Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office.

References

[1] E. B. Baum and D. Haussler. What size net gives valid generalization? *Advances in Neural Information Processing Systems*, 1989.

[2] J. J. Corso. Discriminative modeling by boosting on multilevel aggregates. *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *IEEE Conference Computer Vision and Pattern Recognition*, 2005.

[4] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[5] R. E.Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[6] Y. Freund. An adaptive version of the boost by majority algorithm. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1999.

[7] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119 – 139, 1997.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. In *Dept. of Statistics, Stanford University Technical Report.*, 1998.

[9] D. M. Gavrila. A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[10] X. Huang, S. Z. Li, and Y. Wang. Jensen-shannon boosting learning for object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[11] R. Jin, Y. Liu, L. Si, J. Carbonell, and A. Hauptmann. A new boosting algorithm using input-dependent regularizer. *International Conference on Machine Learning*, 2003.

[12] I. Laptev. Improving object detection with boosted histograms. *Image and Vision Computing*, 27:535–544, 2009.

[13] D. Le and S. Satoh. Ent-boost: Boosting using entropy measure for robust object detection. *International Conference on Pattern Recognition*, 2006.

[14] S. Li, Z. Zhang, H. Shum, and H. Zhang. Floatboost learning for classification. *Advances in Neural Information Processing Systems*, 2002.

[15] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1112 – 1123, 2004.

[16] C. Liu and H.-Y. Shum. Kullback-leibler boosting. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[17] T. Mita, T. Kaneko, B. Stenger, and O. Hori. Discriminative feature co-occurrence selection for object detection. *IEEE Trans. Pattern Analysis Machine Intelligence*, 30(7):1257–1269, 2008.

[18] T. Mitsui and H. Fujiyoshi. Object detection by joint features based on two-stage boosting. *International Workshop on Visual Surveillance*, pages 1169 – 1176, 2009.

[19] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. *European Conference on Computer Vision*, 2006.

[20] Y. Ouyang, M. Tang, J. Wang, H. Lu, and S. Ma. Boosting relative spaces for categorizing objects with large intra-class variation. *ACM International Conference on Multimedia*, pages 663–666, 2008.

[21] G. Overett, L. Petersson, N. Brewer, L. Andersson, and N. Pettersson. A new pedestrian dataset for supervised learning. *IEEE Intelligent Vehicles Symposium*, 2008.

[22] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss. The feret evaluation methodology for face recognition algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:1090–1104, 2000.

[23] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss. The feret database and evaluation procedure for face recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

[24] B. Rasolzadeh, L. Petersson, and N. Pettersson. Response binning: Improved weak classifiers for boosting. *IEEE Intelligent Vehicles Symposium*, 2006.

[25] G. Ratsch, T. Onoda, and K. R. Muller. Soft margins for adaboost. *Machine Learning*, 2000.

[26] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[27] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

[28] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing Systems*, 2002.

[29] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[30] Y. Yamauchi, M. Takaki, T. Yamashita, and H. Fujiyoshi. Feature co-occurrence representation based on boosting for object detection. *International Workshop on Socially Intelligent Surveillance and Monitoring*, pages 31 – 38, 2010.