On Robustness of On-line Boosting - A Competitive Study

Christian Leistner Amir Saffari Peter M. Roth Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology
{leistner,saffari,pmroth,bischof}@icg.tugraz.at

Abstract

On-line boosting is one of the most successful on-line algorithms and thus applied in many computer vision applications. However, even though boosting, in general, is well known to be susceptible to class-label noise, on-line boosting is mostly applied to self-learning applications such as visual object tracking, where label-noise is an inherent problem. This paper studies the robustness of on-line boosting. Since mainly the applied loss function determines the behavior of boosting, we propose an on-line version of GradientBoost, which allows us to plug in arbitrary loss-functions into our on-line learner. Hence, we can easily study the importance and the behavior of different loss-functions. We evaluate various on-line boosting algorithms in form of a competative study on standard machine learning problems as well as on common computer vision applications such as tracking and autonomous training of object detectors. Our results show that using on-line GradientBoost with robust loss functions leads to superior results in all our experiments.

1. Introduction

Currently, boosting [8] is one of the best and thus one of the mostly applied classification methods in machine learning. It has been extensively discussed and analyzed, both theoretically and experimentally, and many different variants of boosting techniques have been proposed for different applications [25, 18, 26, 10, 12]. In particular, the seminal work of Viola and Jones [27] brought boosting to computer vision, which also paved its way for a wide range of applications. However, boosting is proven, again from both, the theoretical and the experimental point of view, to be sensitive to label noise. This issue was discovered relatively early and, hence, different more robust methods [16, 18, 5, 10, 7, 6, 15, 17] have been proposed.

Originally, boosting was developed for off-line learning, *i.e.*, all training samples must be given in advance. However, for many applications in computer vision such as tracking or background modeling the required data cannot be provided in advance. To overcome these problems Oza [19] proposed an on-line version of boosting[1]. Moreover, he showed that if the algorithm is running for infinite time, the on-line version converges to the off-line version of boosting. Based on these findings several methods for computer vision applications were developed. In particular, Javed *et al.*[13] were the first to use on-line boosting in order to train a visual object detector, while Grabner and Bischof [11] proposed an on-line version for feature selection and demonstrated excellent real-time tracking results. Later, Pham and Cham [21] presented an asymmetric on-line boosting version and Wu and Nevatia [28] showed encouraging self-learning results on the task of pedestrian detection.

However, most of the these vision applications (*e.g.*, tracking or detection) use variants of self-learning classifiers, where noise is inherent and hardly avoidable. For instance, even an already highly performing classifier with 95% accuracy still introduces 5% label noise into the next iteration of a self-learning process. If the algorithm is not able to cope with the noise, it will be accumulated and intensified over next iterations and might result in problems such as drifting and failing of the method. Some previous authors realized that problem and tried to circumvent it, *e.g.*, via using additional learners in order to get only very conservative updates [23] or by taking more robust weak learners in order to improve tracking systems [20]. Yet, few of them considered modifying the learning algorithm itself.

Although for on-line boosting robustness is highly important, in contrast to the off-line case, up to now this issue has not been studied. Thus, as the first contribution of this work, we study the robustness of on-line boosting for feature selection in terms of label noise. In particular, we follow the work of Long *et al.* [15], who showed that the loss function has not only high influence to the learning behavior but also on the robustness. Especially convex loss

[1] In this work, we distinguish "on-line" from "incremental" learning. An on-line method has to discard a sample after learning (no memory); in contrast, an incremental learning method is allowed to store it.

functions (typically used in boosting) are highly susceptible to random noise. Hence, to increase the robustness the goal is to find more suitable less noise sensitive loss functions. For that purpose, we first introduce a generic boosting algorithm, which we call *On-line GradientBoost*, where easily arbitrary loss functions can be plugged in. In fact, this method extends the GradientBoost algorithm of Friedman *et al.* [9] and is similar to the AnyBoost algorithm of Mason *et al.* [18]. Based on this algorithm, we develop different online boosting methods using the loss functions proposed for robust off-line boosting algorithms.

We also study the effect and the influence of different on-line weak learners for the overall behavior of on-line boosting. These theoretical considerations are confirmed by experimental results. In particular, we split our extensive experimental study into three parts. First, we study the different on-line boosting methods on a couple of different machine learning benchmark datasets for both, noise-free and noisy data. Second, we show the benefits of robust online boosting for the application of tracking, where problems such as occlusions typically lead to drifting. Finally, we investigate two co-training methods, where we show that due to the introduced robustness we can cope with wrongly generated labels, which allows to train efficiently competitive classifiers.

2. Robust Boosting

AdaBoost is highly susceptible to noise, where we talk about label noise, if a sample was assigned a wrong label during the labeling process of the data. This sensitivity comes from the fact that AdaBoost increases the weight of a mis-classified sample in each iteration. This re-weighting strategy allows boosting to concentrate on hard samples while easy samples are less emphasized. However, if the sample has a wrong label and the previous weak learners are assigning the true (but hidden) label to the sample, AdaBoost still will consider this as a mis-classification and dramatically (exponentially) increase its weight. This can finally corrupt the learning result. Therefore, the performance of the boosting algorithm will be highly dependent on the presence of such noisy samples.

In the case of off-line boosting, Maclin and Optiz [16] were one of the first to note the sensitivity of the AdaBoost algorithm [8] to label noise. Later, Dietterich [5] conducted more experimental studies analyzing different ensemble building methods and also noted the sensitivity of AdaBoost to label noise. Mason *et al.* [18] was one of the first to analyze boosting algorithms in the context of functional gradient descent. They also proposed a theoretically inspired loss-function and a boosting algorithm using that loss called *DoomII*, which showed increased robustness to label noise. Next, in their seminal work Friedman *et al.* [10] showed the connection of boosting algorithms to the stage-wise additive logistic regression methods from the applied statistics domain. Based on minimizing the negative log-likelihood, they proposed a loss-function and a corresponding boosting algorithm called *LogitBoost*, which also showed to be more resistant to label noise. Domingo and Watanabe [6] and Freund [7] also investigated this issue by proposing the *MadaBoost* and *BrownBoost* methods, respectively, which also try to decrease the label noise sensitivity of AdaBoost. Fast forwarding to recent works in this field, Long and Servedio [15] analyzed different convex loss-functions used in designing boosting algorithms and showed that from a theoretical point of view all of these methods will be sensitive to the label noise.

Recently, Masnadi-Shirazi and Vasconcelos [17] studied the problem of loss-function design from the perspective of *probability elicitation* in statistics and, based on this, derived a non-convex loss-function for boosting. This algorithm, denoted as *SavageBoost*, has shown to be highly resistant to the presence of label noise while also converging fast in case of no noise.

2.1. Loss-Functions

By reviewing robust off-line boosting algorithms in the previous section, we can realize that most of the efforts in order to remedy the noise sensitivity weakness of boosting has been mainly focused on designing better loss-functions. In Figure 1 we depict a few of the loss-functions commonly used in boosting and other machine learning methods. The corresponding loss-functions used in this figure are shown in Table 1. Here, $y \in \{-1, +1\}$ are the binary labels of a sample x and $F(x)$ is the real output of the classifier with the decision rule of $\hat{y} = \text{sign}(F(x))$. Traditionally, $yF(x)$ is called the *classification margin* of a sample. In fact, if the margin is negative, there is a mis-classification.

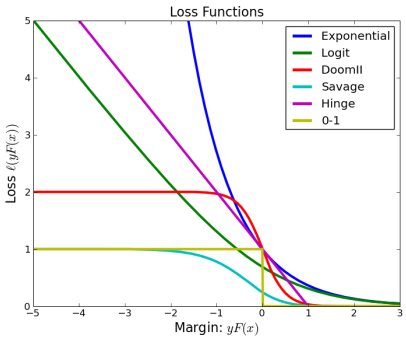


Figure 1. Different loss functions used in boosting and supervised machine learning methods.

From Figure 1 it is clear that the main difference between these loss functions is how they deal with mis-classified samples. There are two scenarios for mis-classification of a

Losses	Functions
0-1 Loss	$\ell_{0-1}(yF(x)) = \mathbb{I}(yF(x) < 0)$
Exponential [8]	$\ell_{exp}(yF(x)) = \exp(-yF(x))$
Logit [10]	$\ell_{log}(yF(x)) = \log(1 + \exp(-yF(x)))$
DoomII [18]	$\ell_{doom}(yF(x)) = 1 - \tanh(yF(x))$
Savage [17]	$\ell_{sav}(yF(x)) = 1/(1 + \exp(2yF(x)))^2$
Hinge	$\ell_{hin}(yF(x)) = \max(0, 1 - yF(x))$

Table 1. Loss functions used in Figure 1.

sample: (1) The sample is noise-free and it is the learning model which is not able to classify it correctly. (2) The sample has a label noise and the learning model is recovering its true (but hidden) label.

More specifically, let y^n be a noisy label, *i.e.*, $y^n = -y^t$, where y^t is the true label. If $y = y^t \neq \hat{y}$, we have to consider the first case; if $y = y^n \neq \hat{y}$ (and equally $\hat{y} = y^t$), the second one. For both cases, the higher the confidence of the classifier $F(x)$ the more the margin will be located towards the left part of Figure 1.

It clearly can be seen that different loss functions behave differently in such situations since they are covering different parts of the mis-classification spectrum. AdaBoost, which uses the exponential loss, has the most aggressive penalty for a mis-classification. This justifies why AdaBoost dramatically increases the weight of mis-classified samples. Going further, one can see that Logit and Hinge losses are less aggressive and their penalty increases linearly on the left side of the figure. In contrast, DoomII and Savage follow totally different strategies. First, their loss-functions are non-convex. Second, they almost give up on putting pressure over the classifier when there is a severe mis-classification (*i.e.*, $F(x)$ is large). Notably, DoomII gives up much earlier but maintains a higher overall penalty compared to the Savage loss.

2.2. On-line GradientBoost

To allow a comparison of different loss functions, we propose an on-line formulation of the GradientBoost [9] for feature selection, which we call *On-line GradientBoost*. GradientBoost performs an stage-wise functional gradient descent over a given loss function [18, 10, 9]. More specifically, for a loss $\ell(\cdot)$, we want to find a set of base functions or weak learners $\{f_1(x), \cdots, f_M(x)\}$ and their corresponding boosting model

$$F(x) = \sum_{m=1}^{M} f_m(x) \tag{1}$$

which minimizes this loss.

Given a loss function $\ell(\cdot)$ and a training dataset, $\mathcal{X} = \{(x_1, y_1), \cdots, (x_N, y_N)\}, x_n \in \mathbb{R}^D, y_n \in \{-1, +1\}$, the empirical loss is

$$\mathcal{L}(F(x)) = \sum_{n=1}^{N} \ell(y_n F(x_n)). \tag{2}$$

At stage t of GradientBoost, we are looking for a base function which has the maximum correlation with negative direction of the loss function. This can be written as

$$f_t(x) = \arg\max_{f(x)} \; -\nabla\mathcal{L}^T f(x), \tag{3}$$

where $\nabla\mathcal{L}$ is the gradient vector of the loss at $F_{t-1}(x) = \sum_{m=1}^{t-1} f_m(x)$. This can be simplified to

$$f_t(x) = \arg\max_{f(x)} \; -\sum_{n=1}^{N} y_n \ell'(y_n F_{t-1}(x_n)) f(x_n). \tag{4}$$

where $\ell'(\cdot)$ shows the derivatives of the loss with respect to F_{t-1}. Denote the sample weight as $w_n = -\ell'(y_n F_{t-1}(x_n))$. Thus, the optimization is equivalent to maximizing the weighted classification accuracy

$$f_t(x) = \arg\max_{f(x)} \sum_{n=1}^{N} w_n y_n f(x_n). \tag{5}$$

Based on these derivations, we propose an on-line version of this algorithm, which we show in Algorithm 1. As in [11] we keep a fixed set of weak learners and perform boosting on the selectors. The m^{th} selector maintains a set of K weak learners $\mathcal{S}_m = \{f_m^1(x), \cdots, f_m^K(x)\}$ and at each stage it selects its best performing weak learners. The optimization step Eq.(5) is then performed iteratively by propagating the samples through the selectors and updating the weight estimate λ_m according to the negative derivative of the loss function. Thus, the algorithm is independent of the used loss function.

In Figure 2 we also plot the functions for the weight updates for different popular loss functions. As can be seen, the exponential loss has an unbounded weight update function, while all other loss functions are bounded between $[0, 1]$. Most importantly, Logit and Hinge weight update functions saturate at 1, as the margin decreases, while the weights of Doom and Savage fades out.

2.3. On-line Weak Learners

For the method described in Algorithm 1 we need on-line weak learners. Traditionally, decision trees or stumps have successfully been used for off-line boosting. For the on-line boosting, *e.g.*, Grabner and Bischof, [11], it is assumed that the feature responses are Gaussian distributed, where the means μ^+, μ^- and the standard deviations σ^+, σ^- are estimated with the help of a Kalman filter.

Some variants of on-line GradientBoost, *e.g.*, RealBoost, need confidence-rated weak predictions. Therefore, we use

Algorithm 1 On-line GradientBoost

Require: A training sample: (x_n, y_n).
Require: A differentiable loss function $\ell(\cdot)$.
Require: Number of selectors M.
Require: Number of weak learners per selector K.
Set $F_0(x_n) = 0$.
Set the initial weight $w_n = -\ell'(0)$.
for $m = 1$ to M **do**
  for $k = 1$ to K **do**
    Train k^{th} weak learner $f_m^k(x)$ with sample (x_n, y_n) and weight w_n.
    Compute the error: $e_m^k \leftarrow e_m^k + w_n \mathbb{I}(\text{sign}(f_m^k(x_n)) \neq y_n)$.
  end for
  Find the best weak learner with the least total weighted error: $j = \arg\min_k e_m^k$.
  Set $f_m(x_n) = f_m^j(x_n)$.
  Set $F_m(x_n) = F_{m-1}(x_n) + f_m(x_n)$.
  Set the weight $w_n = -\ell'(y_n F_m(x_n))$.
end for
Output the final model: $F(x)$

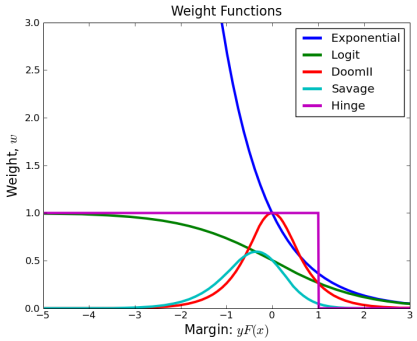


Figure 2. Weight update functions for different loss functions.

on-line histograms as weak learners since they can be easily calculated on-line. Following [10] we use the probabilistic output of the classifier in form of

$$f(x) = 0.5 \log \frac{p_+(x)}{1 - p_+(x)}, \tag{6}$$

where $p_+(x)$ is the probability of a sample to be positive. Histograms can also inherently describe multi-modal distributions, which as we show in the experiments is also helpful for algorithms which in principle do not demand confidence-rated predictions.

3. Experimental Results

The goals of the experiments are are trifold. First, we compare the performance of the novel On-line GradientBoost algorithm with the traditional on-line boosting method. Second, we investigate which loss function behaves best in case of label noise. Third, we analyze the effect of using on-line histograms instead of stumps [2]. Therefore, we organized an extensive set of experiments by using benchmark machine learning datasets, benchmark tracking videos, and co-training of object detectors [3].

3.1. Machine Learning Experiments

For the machine learning experiments we chose 8 benchmark datasets from UCI and LIBSVM repositories, which are shown in Table 2. We compare the performance of on-line *AdaBoost* and *GradientBoost* by using exponential, Logit, DoomII, and Savage losses. Note, when using exponential loss, we get the on-line formulation of RealBoost of Friedman *et al.* [10]. For these experiments, we randomly introduce label noise into the training set and train the on-line classifiers for 5 epochs. We repeat all these experiments for 3 times and report the average test errors. We also repeat these experiments for two different weak learners discussed in Section 2.3. This yields in total 192 experiments per classifier, which hopefully shows clearly which methods perform best in presence of noise. For multi-class datasets, we used an one-vs-all strategy and incorporated the ratio between positive and negative samples in the initial weight of samples.

Dataset	# Train	# Test	# Class	# Feat.
DNA	1400	1186	3	180
Letter	15000	5000	26	16
Magic	9510	9510	2	10
Pendigits	7494	3498	10	16
SatImage	3104	2000	6	36
Shuttle	30450	14500	7	9
Splice	1000	2175	2	60
USPS	7291	2007	10	256

Table 2. Datasets used in machine learning experiments.

Figures 3(a) and 3(b) show the test error with respect to the amount of label noise in the training set for each classifier and each dataset by using stumps and histograms, respectively. The main outcome of these experiments is that on average On-line GradientBoost using Logit or Savage losses performs best. Even though using the DoomII loos function provides excellent results outperforming the others for some datasets, the results obtained by using Logit and Savage loss-functions are consistently among the best.

Additionally, as it can be seen, the Real loss-function

[2] Note that due to performance issues we do not use the features proposed in [20].

[3] Source code is available under www.ymer.org/amir/software/online-gradient-boosting

does not perform very well when using stumps, but is competitive when the weak learner is switched to histograms. The reason for this behavior could be attributed to the fact that stumps are not able to return probabilistic outputs, which are required by on-line GradientBoost losses. By referring to Figure 2, we can see that for the Real loos-function does not provide a bounded weight update function, and therefore, without a probabilistic estimates, the sample weights grow unbounded. Additionally, by comparing these two figures it becomes clear that, in general, histograms are better than stumps; especially, for AdaBoost and RealBoost significant improvements can be achieved.

Figure 4 shows the number of wins for each classifier over different noise levels over all datasets and weak learners. As can be seen, SavageBoost wins more than all other methods, but LogitBoost gains more and more wins if the noise level is increased, while SavageBoost has no win when the noise level is at its maximum.

3.2. Visual Tracking

In this experiment, we evaluated on-line GradientBoost on various publicly available tracking scenarios and compared it to a state-of-the-art tracker based on on-line AdaBoost [11]. For GradientBoost, we chose the logistic loss, since the theoretical considerations as well as experimental results on the machine learning data show that this method can be considered a suitable trade-off between accuracy and robustness. Note that we skipped the results for Savage-loss since it performs similar to the logit-loss. We also use histograms with 32 bins as weak learners. For both, AdaBoost and GradientBoost, the same implementation framework with the same features was used. Since the main purpose of this experiment is the comparison of the two on-line algorithms for the tracking task, we only use simple Haar-features, did not implement any rotation and scale search, and abstain from any other engineering, although these things would definitely help to improve the tracking results. For both on-line boosting methods we used 50 selectors each of them having 150 features.

Datasets Our datasets consist of three publicly available sequences presenting various types of lighting, pose, scale and appearance changes. The first "Occluded Face" was taken from [1] [4]. Then, we took the famous "David Indoor" sequence from Ross *et al.* [22] and "Rotating Girl" from [2]. All sequences are grey-scale and resized to 320 x 240 pixels.

The detailed results of our experiments are depicted in Figure 5, where the x-axes indicate the Euclidian distance to the ground-truth. Some representative results are given in Figure 6. As can be seen, the performance of the two on-line boosting methods for the task of visual tracking can vary a lot between the different sequences. An important insight, however, is the more noise robust GradientBoost method never performs worse than AdaBoost and, *e.g.*, in the "David Indoor"-scene performs superior to the base-line algorithm.

[4] Please note that we do not compare to [1], because it uses much better features and our focus lies on the comparison of the different boosting algorithms

3.3. Visual Co-training of Person Detectors

In this experiment, we analyze the robustness of on-line boosting on training a pedestrian detector on both labeled and unlabeled data using co-training [3], which is another typical application of on-line boosting. As for tracking, we compare on-line GradientBoost with a logit-loss to on-line AdaBoost for feature selection.

3.3.1 Single-view co-training

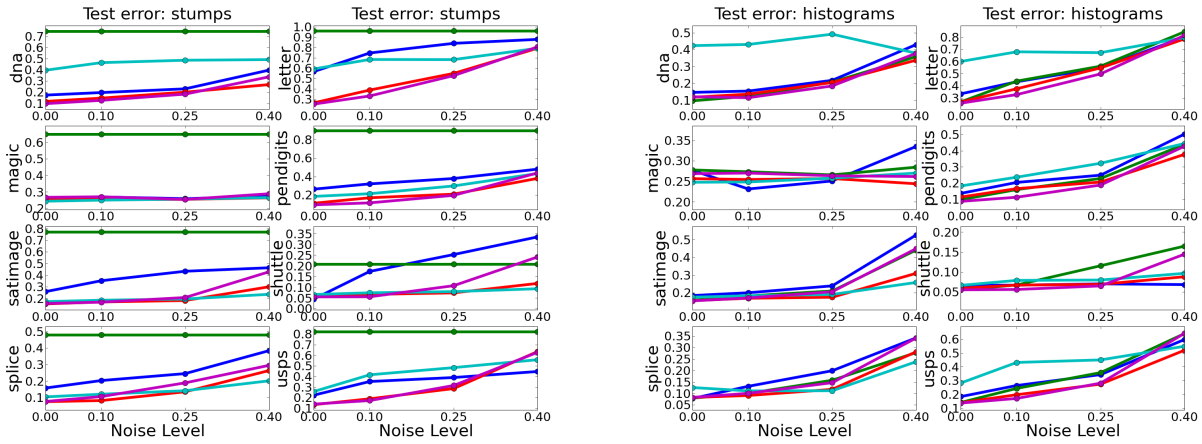
For this experiment, we adapted the co-training approach of Levin *et al.* [14], who co-trained two boosted off-line classifiers (one for gray-value images and the other from background subtracted images) to learn a car detector. In particular, we trained the initial classifiers using only 25 labeled positives samples, which are then updated by on-line co-training.

To demonstrate the learning progress, after a pre-defined number of processed training frames t we saved a classifier (*i.e.*, $t = 0$, $t = 250$, $t = 500$, and $t = 750$), which was then evaluated on an independent test sequence (*i.e.*, the current classifier was evaluated but no updates were performed!). To analyze the detection results, we compare $precision$-$recall$-curves (RPC) obtained from a 50% overlap criterion. The corresponding curves for AdaBoost and GradientBoost are illustrated in Figure 7a and Figure 7b, respectively. It clearly can be seen that using the non-robust learner the system fails completely. In fact, the upcoming wrong updates cannot be handled and the classifier is degraded. In contrast, even starting from a similar initial level, using GradientBoost, finally, a competitive classifier can be obtained.

3.3.2 Multi-view co-training

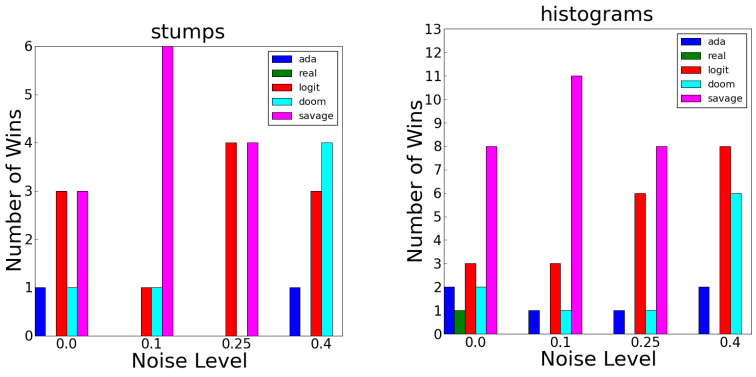
In this experiment, we followed the approach presented in [24], which showed that incorporating an additional camera (*i.e.*, an additional geometric cue) into the co-training process can significantly improve the training results. In particular, for the multi-camera co-training strategy the multiple views on the data are realized by multiple camera views, mapping the local image coordinate systems onto each other by using a homography.

Again, as in Section 3.3.1, we run the same experiment for the standard on-line AdaBoost as well as for the on-line



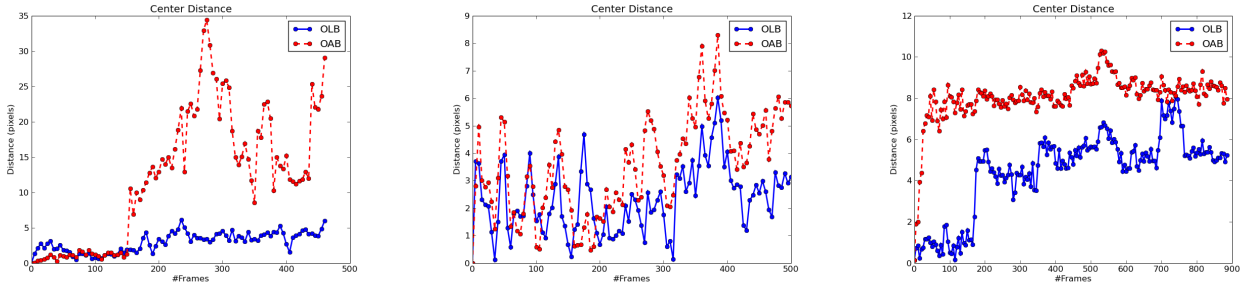
(a) Stumps (b) Histograms

Figure 3. Results for machine learning experiment when (a) stumps and (b) histograms are used as weak learners: test error is shown with respect to the label noise level. Classifiers: AdaBoost (blue), Real (green), Logit (red), DoomII (cyan), Savage (magenta).



(a) Stumps (b) Histograms

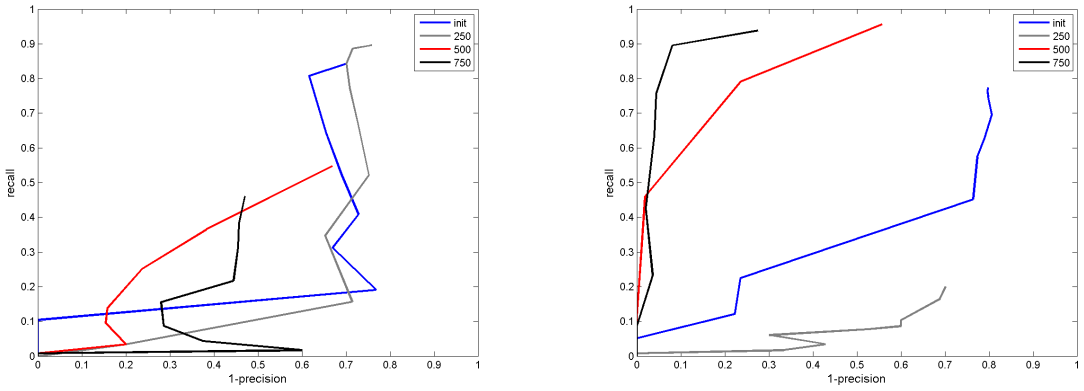
Figure 4. Different number of wins for machine learning experiment when (a) stumps and (b) histograms are used as weak learners: test error is shown with respect to the label noise level. Classifiers: AdaBoost (blue), Real (green), Logit (red), DoomII (cyan), Savage (magenta).



(a) David (b) Girl (c) Faceocc

Figure 5. Comparison of ORFs with OAB on four state-of-the-art tracking sequences. As can be seen, our method significantly outperforms boosting on all sequences.

Figure 6. Comparison of OLB and OAB on three public sequences.



(a) AdaBoost (b) GradientBoost

Figure 7. Single-view *Co-training* using (a) the non-robust on-line AdaBoost algorithm and (b) the robust On-line GradientBoost algorithm.

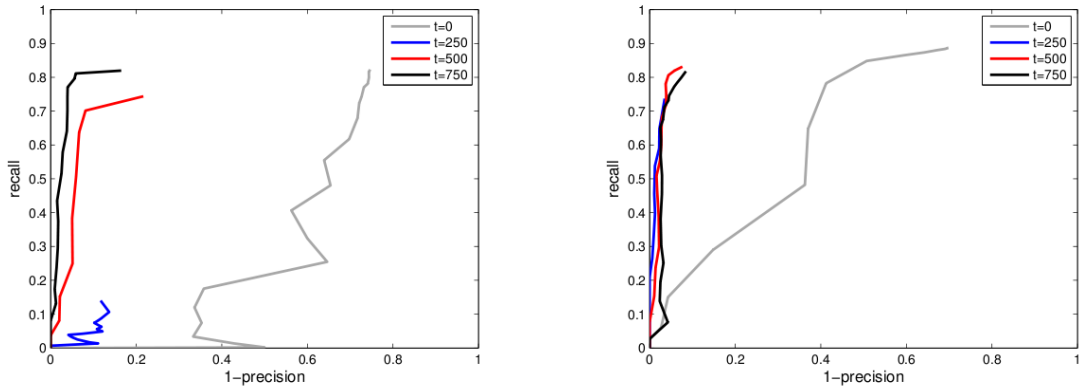
GradientBoost. The thus obtained results for one of the two views are shown in Figure 8a and Figure 8b, respectively. In contrast to the results shown in Section 3.3.1 it can be seen that for both methods finally competitive classifiers are obtained. In fact, the geometric constraints provide a very strong cue and only very valuable samples are used for updating; hence, the on-line learner has to handle a reduced amount of noise. Hence, although GradientBoost converges a bit faster, we can conclude that for such a more robust scenario AdaBoost is sufficient to deliver sufficient results.

4. Conclusion

We studied the robustness of on-line boosting for feature selection against class-label noise. Since it was shown that the applied loss-function has high impact on the behavior of boosting, we proposed a novel boosting algorithm *On-line GradientBoost*, that provides a flexible way to plug in and test different kinds of loss-functions. Based on this algorithm we were able to analyze different loss-functions and their influence on the robustness of on-line boosting. Additionally, we studied the behavior depending on different weak learners. We run experiments for machine learning data as well as for real-world applications (*i.e.*, tracking and object detection) and showed that a more noise robust on-line learner combined with simple histogram-based weak learners can significantly improve the results in the presence of noise. In this work, we mainly considered the influence of different loss functions. However, for future work, we also plan to investigate the influence of other strategies that might help improving the robustness of on-line boosting, for instance, methods based on solving linear programs similar to νLP-Boost [4] or *BrownBoost* [7].

Acknowledgment

This work was supported by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04 and the Austrian Science Fund (FWF P18600), by the FFG projects AUTOVISTA (813395) and EVis (813399) under the FIT-IT programme.



(a) AdaBoost (b) GradientBoost

Figure 8. Multi-view *Co-training* using (a) the non-robust on-line AdaBoost algorithm and (b) the robust On-line GradientBoost algorithm.

References

[1] A. Adam, E. Rivlin, and I. Shimschoni. Robust fragments-based tracking using the integral histogram. In *Proc. CVPR*, 2006.

[2] S. Birchfeld. Elliptical head tracking using intensity gradients and color histograms. In *Proc. CVPR*, 1998.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. COLT*, 1998.

[4] A. Demiriz, K. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[5] T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. 40:139–157, 1998.

[6] C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. In *Proc. COLT*, 2000.

[7] Y. Freund. An adaptive version of the boost by majority algorithm. In *Proc. COLT*, 2000.

[8] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. ICML*, 1996.

[9] J. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

[11] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. CVPR*, 2006.

[12] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[13] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proc. CVPR*, 2005.

[14] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proc. ICCV*, 2003.

[15] P. M. Long and R. A. Servedio. Random classification noise defeats all convex potential boosters. In *Proc. ICML*, 2008.

[16] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proc. National Conf. on Artificial Intelligence*, 1997.

[17] H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances NIPS*, 2008.

[18] L. Mason, J. Baxter, P. Bartlett, and M. Frean. chapter Functional gradient techniques for combining hypotheses, pages 221–247. MIT Press, Cambridge, MA., 1999.

[19] N. Oza and S. Russell. Online bagging and boosting. In *Proc. Artificial Intelligence and Statistics*, 2001.

[20] T. Parag, F. Porikli, and A. Elgammal. Boosting adaptive linear weak classifiers for online learning and tracking. In *Proc. CVPR*, 2008.

[21] M.-T. Pham and T.-J. Cham. Online asymetric boosted clasifiers for object detection. In *Proc. CVPR*, 2007.

[22] D. Ross, J. Lim, and M. Yang. Adaptive proballistic visual tracking with incremental subspace update. In *Proc. ECCV*, 2004.

[23] P. M. Roth and H. Bischof. *Machine Learning Techniques for Multimedia*, chapter Conservative Learning for Object Detectors, pages 139–158. Springer, 2008.

[24] P. M. Roth, C. Leistner, H. Grabner, and H. Bischof. *Multi-Camera Networks, Principles and Applications*, chapter Online Learning of Person Detectors by Co-Training from Multiple Cameras, pages 313–334. Academic Press, 2009.

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

[26] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[27] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001.

[28] B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. In *Proc. CVPR*, 2007.