Boosting Algorithms for Simultaneous Feature Extraction and Selection

Mohammad J. Saberian Nuno Vasconcelos
Department of Electrical and Computer Engineering
University of California, San Diego
saberian@ucsd.edu, nuno@ece.ucsd.edu

Abstract

The problem of simultaneous feature extraction and selection, for classifier design, is considered. A new framework is proposed, based on boosting algorithms that can either 1) select existing features or 2) assemble a combination of these features. This framework is simple and mathematically sound, derived from the statistical view of boosting and Taylor series approximations in functional space. Unlike classical boosting, which is limited to linear feature combinations, the new algorithms support more sophisticated combinations of weak learners, such as "sums of products" or "products of sums". This is shown to enable the design of fairly complex predictor structures with few weak learners in a fully automated manner, leading to faster and more accurate classifiers, based on more informative features. Extensive experiments on synthetic data, UCI datasets, object detection and scene recognition show that these predictors consistently lead to more accurate classifiers than classical boosting algorithms.

1. Introduction

A central problem in computer vision is that of learning good image representations. One of the most challenging aspects of this problem is the search for the best features for the solution of problems such as object detection. Recent progress has drawn extensively in machine learning, where there are two main approaches to feature learning, feature selection (FS) and feature extraction (FE). FS starts from a large pool of features and uses learning techniques to select the best subset for the problem at hand. FE creates a set of features by designing a transformation from the image space to a new space where the problem of interest is best solved. In general, FE produces better solutions since it is not limited by the representation power of the original feature set. On the other hand, it usually involves a more complex learning problem, and is frequently too complex for practical applications.

While there are many approaches to unsupervised FE, e.g. dimensionality reduction methods such as principal component analysis, or manifold learning, sparse decompositions, deep belief networks, among others, substantially less progress has been reported in the area of supervised FE. This is of particular relevance for vision, where various authors have noted the importance of complex features in biological recognition [19], [17]. In fact, a major open problem for computer vision is how to replicate the ability of biology to learn layers of recognizers based on features that are simultaneously more selective for the objects of interest (discriminant) and invariant than their predecessors. Complex features also play an important role in the modeling of object parts, and are critical to the success of methods that represent objects as constellations of such parts [4]. The ability to address these questions requires the development of powerful supervised FE methods.

Substantially greater progress has been achieved in the area of supervised FS. In fact, modern classifiers, such as support vector machines (SVM), boosting, or logistic regression, implement simple hyperplane boundaries in learned feature spaces. The features that define these spaces are selected from either the set of training examples (e.g. the support vectors of the SVM) or from a large pre-defined feature pool (e.g the Haar wavelets popular in the boosting literature). These features tend to have little individual discriminant power, because they are either too example specific (SVM templates) or too simple (Haar wavelets). In result, a large number of features must be selected, and the classifier learned in a high dimensional space. While the robustness of the large margin principle enables great robustness to the dimensionality of this space, high dimensional spaces of weak features increase classifier complexity and the potential for over-fitting. It is not uncommon for a SVM to select 80% of its training set as support vectors, or for boosting to produce classifiers with thousands of features.

In fact, much of the research in fast object detection, e.g. the design of detector cascades [20], attempts to overcome this problem by judiciously controlling the number of features evaluated per classified example. Nevertheless, the weakness of the individual features bounds the representa-

Figure 1. Example of the Haar features used in face detection [20].

tion power of the resulting object detectors, and can limit object detection performance. This is illustrated in Figure 1 which depicts a well known example from face detection [20] where boosting is able to select effective linear combinations of very simple Haar wavelets. It is clear that the approximation of a face by these patterns will require a very large number of them. In the case of more complex objects, or objects that exhibit more variability, the approximation may not even be feasible with tractable amounts of detector complexity and speed. In the boosting literature, these scenarios are addressed by resorting to more complex weak learners, usually decision trees. This, however, is known to be somewhat unstable, since the robustness of boosting to over-fitting quickly erodes with tree depth. In practice, no tree depth guarantees uniformly good performance, and the default choice seems to be the decision stump, which is a threshold on a single feature.

An alternative solution is to maintain the features simple but add features combinations, such as feature products or logical operations, e.g. "and" or "or", to the feature pool. This, however is not trivial when the starting feature pool is large already, as is usually the case in vision (e.g. $50,000$ Haar wavelets in [20]). One possibility is to endow the learning algorithm with the ability to either use the available features or *create* new combinations of these features, as needed. Some steps towards this goal have recently been given in the boosting literature. One example is the method of [7] which learns products of decision trees. Alternatively, [3] suggests the addition, to the boosted predictor, of logical combinations ("and", "or") of previously selected weak learners. [13], on the other hand, relies on a combination of binary quantized feature responses and sequential forward selection to design new weak learners. Yet, all of these works rely on a single type of feature combination, or require extensive (or approximate) search of the space of feature combinations, to create new weak learners.

In this work, we aim to derive boosting algorithms for FE. This is done by formulating FE as the problem of learning more sophisticated combinations of weak learners than their classical addition. A new framework for the design of boosting algorithms that learns such combinations is then introduced. The framework is simple and mathematically sound, relying on the statistical view of Boosting [6] and Taylor series approximations in functional space [18]. The resulting boosting algorithms grow a predictor by selecting among a pair of predefined operations, which could be sums and products, or "ands" and "ors", among others. These operations can be added as needed, so as to produce groups of features of greater complexity than those in the initial pool. Two new boosting algorithms are derived from the proposed framework. In addition to the classical linear combination of weak learners, they can learn *sums of weak learner products* (SOP) or *products of weak learner sums* (POS). SOPBoost is shown to generalize classical boosting algorithms, converge to a globally optimal solution, and grow fairly complex predictor structures with few weak learners, in a fully automated way. Extensive experiments on synthetic data, UCI datasets, object detection and scene recognition show that it is consistently able to produce more accurate classifiers than classical boosting methods. This includes boosting algorithms with a variety of weak learners, ranging from simple regressors to trees of non-trivial depth.

2. Boosting

A classifier is a mapping $h(x)$ from examples $x \in \mathcal{X}$ into labels $y \in \{-1, 1\}$, usually implemented as $h(x) = sgn[f(x)]$ where $f : \mathcal{X} \to \mathbb{R}$ is a real-valued predictor. The optimal predictor $f^*(x)$ is the minimizer of the classification risk

$$R(f) = E_{X,Y}\{L[yf(x)]\} \approx \sum_i L[y_i f(x_i)] \qquad (1)$$

where $L(.)$ is a loss function that upper bounds the error rate. Boosting methods learn and approximate the optimal predictor as a *linear* combination of simpler predictors, $g_k : \mathcal{X} \to \mathbb{R}$, called weak learners i.e.

$$f(x) = \sum_k \alpha_k g_k(x) \quad g \in \mathcal{G} \qquad (2)$$

where $\mathcal{G} = \{g_1, .., g_m\}$ is the set of all weak learners. For mathematical consistency, we assume that $g(x) = 0$ and $g(x) = 1$ are in $\mathcal{G}$. Under the statistical view of [6, 12], boosting methods learn this linear combination by iterative descent methods in functional space, with respect to the optimization problem

$$\begin{cases} \min_{f(x)} & R(f) \\ s.t & f(x) \in \Omega_{\mathcal{G}}, \end{cases} \qquad (3)$$

where $\Omega_{\mathcal{G}}$ is the set of all linear combinations of weak learners in $\mathcal{G}$. Note that $\Omega_{\mathcal{G}}$ is a convex set and, if $R(.)$ is a convex function, the optimization problem of (3) is convex.

Let the first and second derivative of the loss function used in (1) be $L' = \frac{\partial L(v)}{\partial v}$ and $L'' = \frac{\partial^2 L(v)}{\partial v^2}$ and assume that, after k iterations, the estimate of the optimal predictor is $f^k(x)$. Using a Taylor series expansion of $R(f^k + g)$ around f^k, the first and second order functional variations

along the direction of weak learner $g(x) \in \mathcal{G}$ are [18]

$$\delta R(f^k; g) = \left. \frac{\partial R(f^k + \xi g)}{\partial \xi} \right|_{\xi=0} \quad (4)$$

$$= \sum_i y_i g(x_i) L'[y_i f^k(x_i)] \quad (5)$$

$$\delta^2 R(f^k; g) = \left. \frac{\partial^2 R(f^k + \xi g)}{\partial \xi^2} \right|_{\xi=0} \quad (6)$$

$$= \sum_i g^2(x_i) L''[y_i f^k(x_i)]. \quad (7)$$

From these, [18] has shown that the best weak learner to add to the predictor at iteration $k+1$ is

$$g^* = \arg\min_{g \in \mathcal{G}} \delta R(f^k; g) \quad (8)$$

when gradient descent is used, and

$$g^* = \arg\max_{g \in \mathcal{G}} \frac{[\delta R(f^k; g)]^2}{\delta^2 R(f^k; g)} \quad (9)$$

when Newton method is used. Given the best weak learner, g^*, the optimal step size is

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}} R(f^k + \alpha g^*). \quad (10)$$

The predictor estimate is then updated as

$$f^{k+1}(x) = f^k(x) + \alpha^* g^*(x), \quad (11)$$

and the final predictor would be a *linear* combination of weak learners.

3. Boosting new feature combinations

Although boosting selects informative features for classification, the original set of weak learners may not be rich enough to capture all the feature space dimensions required for discrimination. For example, it may be necessary to use conjunctions of the features to capture some of these dimensions. In this case, the linear combination is insufficient, and boosting can fail to produce an accurate detector. To overcome this problem, we propose a boosting procedure that can learn more sophisticated combinations of weak learners, e.g. sums of products or products of sums.

3.1. Boosting Sum of Products, SOP-Boost

Given a set of weak learners $\mathcal{G}$, SOP-Boost aims to solve

$$\begin{cases} \min_{f(x)} & R(f) \\ s.t & f(x) \in \Omega^{sop}_{\mathcal{G}}. \end{cases} \quad (12)$$

where $\Omega^{sop}_{\mathcal{G}}$ is now the set of all possible *linear combinations of products* of weak learners, i.e.

$$\Omega^{sop}_{\mathcal{G}} = \left\{ h(x) | h(x) = \sum_j \prod_l g_{j,l}(x), g_{j,l} \in \mathcal{G} \right\}. \quad (13)$$

It can be shown that $\Omega^{sop}_{\mathcal{G}}$ is a convex set. Hence, for any convex function $R(.)$, the optimization problem of (12) is convex.

The proposed boosting algorithm is based on the TaylorBoost framework [18]. Assume that after k iterations the predictor has m terms,

$$f^k(x) = \sum_{j=1}^{m} p^k_j(x), \quad (14)$$

each of which is a single weak learner or a product of weak learners,

$$p^k_j(x) = \prod_{l=1}^{t_j} g_{j,l}(x) \quad g_{j,l}(x) \in \mathcal{G}. \quad (15)$$

At iteration $k+1$ it is possible to improve $f^k(x)$ with two types of updates: 1) additive and 2) multiplicative.

Additive update: In this case we consider adding a weak learner to the predictor, i.e. $f^{k+1}(x) = f^k(x) + g(x)$. The optimal updates are as in standard boosting, i.e. $\delta R(f^k; g)$ and $\delta^2 R(f^k; g)$ are given by (5) and (7), respectively, the optimal weak learner, g^*_0, is obtained by (8) or (9), depending on the choice of gradient descent or Newton method, and the optimal step size, α^*_0, is given by (10). The updated predictor has risk

$$\hat{R}_0 = R(f^k + \alpha^*_0 g^*_0). \quad (16)$$

Multiplicative update: In this case, one of the existing terms is multiplied by a new weak learner, i.e. $p^{k+1}_r(x) = p^k_r(x) \times g(x)$. Using (14), this results in

$$f^{k+1}(x) = p^k_r(x) g(x) + \sum_{j \neq r} p^k_j(x) \quad (17)$$

$$= [f^k(x) - p^k_r(x)] + p^k_r(x) g(x) \quad (18)$$

$$= Q^k_r(x) + p^k_r(x) g(x) \quad (19)$$

where $Q^k_r(x) = f^k(x) - p^k_r(x)$. Using (19) and a Taylor series expansion of $R(f^{k+1})$ around functional $Q^k_r(x)$, the first and second order variations of the risk with respect to a multiplicative update of the r^{th} term in $f^k(x)$ are

$$\delta R(f^k; g, r) = \left. \frac{\partial R[Q^k_r + \xi p^k_r g]}{\partial \xi} \right|_{\xi=0} \quad (20)$$

$$= \sum_i y_i g(x_i) p^k_r(x_i) L'[y_i Q^k_r(x_i)] \quad (21)$$

$$\delta^2 R(f^k; g, r) = \left. \frac{\partial^2 R[Q^k_r + \xi p^k_r g]}{\partial \xi^2} \right|_{\xi=0} \quad (22)$$

$$= \sum_i [g(x_i) p^k_r(x_i)]^2 L''[y_i Q^k_r(x_i)]. \quad (23)$$

Algorithm 1 SOP-Boost

Input: Training set S_t, set of weak learners $\mathcal{G} = \{g_1, ...g_m\}$, Number of iteration N and a definition of loss function $L(.)$.
Initialization: Set $k = 0$, $m = 0$, $p_m^k(x) = 0$ and $f^k(x) = 0$.
while $k < N$ **do**
  Find the best additive update $\alpha_0^* g_0^*$ by using (5) and (7) in (8) or (9) and (10).
  Set $\hat{R}_0 = R(f^k + \alpha_0^* g_0^*)$.
  for $r = 1$ to m **do**
    Find the best update for r^{th} product term, $\alpha_r^* g_r^*$, using (21) and (23) in (8) or (9) and (24).
    Set $\hat{R}_r = R[(f^k - p_r^k) + p_r^k \alpha_r^* g_r^*]$.
  end for
  Set $r^* = \arg\min_r \hat{R}_r \quad r = 0, \ldots, m$.
  if $r^* = 0$ **then**
    $p_{m+1}^{k+1} = \alpha_0^* g_0^*$
    $m = m + 1$
  else
    $p_{r^*}^{k+1} = p_{r^*}^k \times \alpha_{r^*}^* g_{r^*}^*$
  end if
  $p_r^{k+1} = p_r^k \quad r \neq r^*$
  $f^{k+1}(x) = \sum_{r=1}^m p_r^{k+1}(x)$
  $k = k + 1$
end while
Output: decision rule: $sign[f^N(x)]$

The best weak learner, g_r^*, is given by the combination of (8) or (9) with (21) and (23), and the optimal step size is

$$\alpha_r^* = \arg\min_{\alpha \in \mathbb{R}} R(Q_r^k + \alpha g_r^*). \tag{24}$$

The updated predictor has risk

$$\hat{R}_r = R(Q_r^k + \alpha_r^* g_r^*). \tag{25}$$

SOP-Boost computes the optimal weak learners, and corresponding risks, under two strategies: 1) one additive update and 2) m multiplicative updates. It then selects the update that most reduces the classification risk. This method is presented in Algorithm 1.

3.2. Boosting Product of Sums, POS-Boost

Given a set of weak learners $\mathcal{G}$, POS-Boost aims to solve

$$\begin{cases} \min_{f(x)} & R(f) \\ s.t & f(x) \in \Omega_{\mathcal{G}}^{pos}. \end{cases} \tag{26}$$

where

$$\Omega_{\mathcal{G}}^{pos} = \left\{ h(x) | h(x) = \prod_j \sum_l g_{j,l}(x), g_{j,l} \in \mathcal{G} \right\}. \tag{27}$$

Note that $\Omega_{\mathcal{G}}^{sop}$ is not a convex set i.e. if $g_1, g_2 \in \Omega_{\mathcal{G}}^{pos}$, $g_1 + g_2$ does not necessarily belong to $\Omega_{\mathcal{G}}^{pos}$. Hence, the optimization problem of (26) is not convex. In this case, the minimization of the risk by descent methods produces a *local minimum*.

The proposed boosting algorithm is again based on TaylorBoost. Assume that after k iterations the predictor is a product of m sums

$$f^k(x) = \prod_{j=1}^m S_j^k(x) \tag{28}$$

where S_j^k is a sum of t_j weak learners,

$$S_j^k(x) = \sum_{l=1}^{t_j} g_{j,l}(x) \quad g_{j,l}(x) \in \mathcal{G}. \tag{29}$$

Similar to SOP-Boost, two updates are considered at iteration $k+1$.

Multiplicative Update: In this case we consider multiplying the predictor by a weak learner, i.e. $f^{k+1}(x) = f^k(x) \times g(x)$. Noting that this update is equivalent to $f^{k+1}(x) = 0 + f^k(x) \times g(x)$, and using a Taylor series expansion of $R(f^{k+1})$ around the zero functional, $z(x) = 0 \ \ \forall x$, leads to the following first and second order variations

$$\delta R(f^k; g) = \left. \frac{\partial R[0 + \xi f^k g]}{\partial \xi} \right|_{\xi=0} \tag{30}$$

$$= \sum_i y_i g(x_i) f^k(x_i) L'[0] \tag{31}$$

$$\delta^2 R(f^k; g) = \left. \frac{\partial^2 R[0 + \xi f^k g]}{\partial \xi^2} \right|_{\xi=0} \tag{32}$$

$$= \sum_i [g(x_i) f^k(x_i)]^2 L''[0]. \tag{33}$$

The best weak learner $g_0^*(x)$ is given by the combination of (8) or (9) with (31) and (33), and the optimal step size is

$$\alpha_0^* = \arg\min_{\alpha \in \mathbb{R}} R(f^k \times \alpha g_0^*). \tag{34}$$

The updated predictor has risk

$$\hat{R}_0 = R(f^k \times \alpha_0^* g_0^*). \tag{35}$$

Additive Update: In this case, a new weak learner is added to one of the existing summation terms, i.e. $S_r^{k+1}(x) = S_r^k(x) + g(x)$ for some r. This results in

$$f^{k+1}(x) = (S_r^k(x) + g(x)) \times \prod_{j \neq r} S_j^k(x) \tag{36}$$

$$= \prod_{j=1}^m S_j^k(x) + \prod_{j \neq r} S_j^k(x) g(x) \tag{37}$$

$$= f^k(x) + T_r^k(x) g(x) \tag{38}$$

where

$$T_r^k(x) = \prod_{j \neq r} S_j^k(x) = \frac{f^k(x)}{S_r^k(x)}. \quad (39)$$

Using a Taylor series expansion of $R(f^{k+1})$ around f^k, the first and second order variations of risk with respect to the addition of $g(x)$ to the r^{th} term of f^k are

$$\delta R(f^k; g, r) = \left. \frac{\partial R[f^k + \xi T_r^k g]}{\partial \xi} \right|_{\xi=0} \quad (40)$$

$$= \sum_i y_i g(x_i) T_r^k(x_i) L'[y_i f^k(x_i)] \quad (41)$$

$$\delta^2 R(f^k; g, r) = \left. \frac{\partial^2 R[f^k + \xi T_r^k g]}{\partial \xi^2} \right|_{\xi=0} \quad (42)$$

$$= \sum_i [g(x_i) T_r^k(x_i)]^2 L''[y_i f^k(x_i)]. \quad (43)$$

The best weak learner, g_r^*, is given by the combination of (8) or (9) with (41) and (43), and the optimal step size is

$$\alpha_r^* = \arg\min_\alpha R(f^k + \alpha T_r^k g_r^*). \quad (44)$$

The updated predictor has risk

$$\hat{R}_r = R(f^k + \alpha_r^* T_r^k g_r^*). \quad (45)$$

Similarly to SOP-Boost, POS-Boost computes the optimal weak learners, and corresponding risks, for each update scenario. It then selects the update that most reduces the classification risk. This method is presented in Algorithm 2.

3.3. Discussion

We have so far derived descent methods in functional space to learn combinations of weak learners that are more sophisticated than those of regular boosting. In fact, using the $+$ and $\times$ operators it is possible to learn four types of combinations, 1) sum of products (SOP-Boost), 2) product of sums (POS-Boost), 3) pure linear (regular boosting), which is a special case of SOP-Boost, and 4) pure product (similar to [7]) which is a special case of POS-Boost. The derivations of SOP-Boost and POS-Boost are the most general for these two operators. They can also be easily generalized to any other pair of operators. This just requires application of a Taylor series expansion, finding the best update under each of the possible predictor update strategies, and selecting the best. For example, it would be possible to use the logical operators "and", "or" (as in [3]) to create the logical equivalent of SOP and POS.

Another interesting property of SOP and POS-Boost is that they combine weak learners automatically. There is no need to pre-specify parameters such as the degree of each term, or the number of terms. On the contrary, the proposed framework adaptively finds the most informative combination of weak learners for a specific classification problem.

Algorithm 2 POS-Boost

Input: Training set S_t, set of weak learners $\mathcal{G} = \{g_1, ...g_m\}$, Number of iteration N and a definition of loss function $L(.)$.
Initialization: Set $k = 0$, $m = 0$, $p_m^k(x) = 1$ and $f^k(x) = 1$
while $k < N$ **do**
  Find the best multiplicative update $\alpha_0^* g_0^*$ by using (31) and (33) in (8) or (9) and (34).
  Set $\hat{R}_0 = R(f^k \times \alpha_0^* g_0^*)$.
  for $r = 1 : m$ **do**
    Find the best update for r^{th} summation term, $\alpha_r^* g_r^*$, using (41) and (43) in (8) or (9) and (44).
    Set $\hat{R}_r = R[f^k + \frac{f^k}{S_r^k} \alpha_r^* g_r^*]$.
  end for
  Set $r^* = \arg\min \hat{R}_r \quad r = 0, \ldots, m$
  if $r^* = 0$ **then**
    $S_{m+1}^{k+1} = \alpha_0^* g_0^*$
    $m = m + 1$
  else
    $S_{r^*}^{k+1} = S_{r^*}^k + \alpha_{r^*}^* g_{r^*}^*$
  end if
  $S_r^{k+1} = S_r^k \quad r \neq r^*$
  $f^{k+1}(x) = \prod_{r=1}^m S_r^{k+1}(x)$
  $k = k + 1$
end while
Output: decision rule: $sign[f^N(x)]$

Hence, if there is any underlying structure, such as important feature correlations, SOP and POS-Boost are likely to exploit it. This is discussed in more detail in Section 4.3, where we tested our method on face detection using combinations of Haar wavelets.

4. Evaluation

This section compares the performance of SOP and POS-Boost with regular boosting (denoted Lin-Boost) on synthetic and real data. In addition, we apply SOP-Boost to the computer vision problems of object detection and scene classification.

4.1. Synthetic data

We start with an experiment relative to the XOR problem, which provides insight. Training and test sets contain $4,000$ examples drawn from four 2D Gaussian random variables of means $[2, 2], [-2, -2], [2, -2], [-2, 2]$ and covariances $[1, .5; .5, 2]$, $[.4, .1; .1, .8]$, $[.4, .1; .1, .8]$, $[1, .3; .3, 1]$, respectively. Samples from the first two Gaussians are considered positive examples and the remaining negatives. This data is shown in figure 2. The weak learners are regressors on the example coordinates. For these learners, the bound-

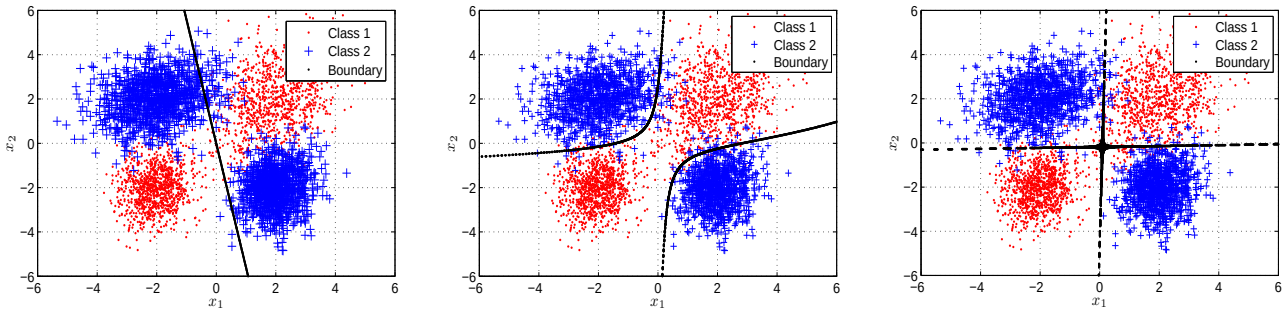

Figure 2. Decision boundaries learned by (Left) regular boosting, (Middle) SOP-Boost and (Right) POS-Boost, on XOR data.

ary produced by regular boosting is a 2D line[1]. On the other hand, because the predictors of SOP-Boost or POS-Boost are multi-nomials, the corresponding boundary can be much more complex. SOP-Boost, POS-Boost, and Lin-Boost, were used with second order TaylorBoost and the logistic loss. The predictors obtained after 20 boosting iterations were of the form

$$\begin{aligned} f_{sop} &= (h_1 \times h_2) + (h_3 \times h_4 \times h_7 \times h_8) \\ &\quad + (h_5 \times h_6 \times h_{14}) + (h_9 \times h_{10} \times h_{11} \times h_{19}) \\ &\quad + (h_{12} \times h_{13} \times h_{17} \times h_{18}) + (h_{15} \times h_{16} + h_{20}). \end{aligned} \tag{46}$$

$$\begin{aligned} f_{pos} &= (\overline{h}_1 + \overline{h}_5 + \overline{h}_7 + \overline{h}_9 + \overline{h}_{13} + \overline{h}_{17}) \\ &\quad \times (\overline{h}_2 + \overline{h}_4 + \overline{h}_6 + \overline{h}_{12} + \overline{h}_{16} + \overline{h}_{20}) \\ &\quad \times (\overline{h}_3 + \overline{h}_8 + \overline{h}_{10} + \overline{h}_{11} + \overline{h}_{14} + \overline{h}_{15} + \overline{h}_{18} + \overline{h}_{19}) \end{aligned} \tag{47}$$

$$f_{Lin} = \sum_{i=1}^{20} \hat{h}_i \tag{48}$$

where $h_k, \overline{h}_k, \hat{h}_k$ are the weak learners selected in the k^{th} iteration of SOP-Boost, POS-Boost and Lin-Boost, respectively. Note that the index indicates the selection order, and is not an identifier of the weak learner. The decision boundaries associated with these predictors are shown in Figure 2. The final test-set error-rate was 47.90% for Lin-Boost, 3.87% for POS-Boost and 2.88% for SOP-Boost. This is an example of how the combinations of weak learners learned by SOP- and POS-Boost can overcome the limitations of these weak learners in a manner that is not possible with the *linear* combinations of Lin-Boost.

4.2. Real data

In a second set of experiments, we compared SOP-Boost, POS-Boost, and Lin-Boost on 7 UCI data sets ("Banana", "Titanic", "Waveform", "Twonorm", "Thyroid", "Splice" and "Ringnorm"). These are provided with $20 - 100$ predefined train/test splits [1], for which we report average results. We again used second order TaylorBoost, but considered four loss functions: the exponential of [5], logistic of [6], canonical boosting loss (CBL) of [11], and Laplace loss of [18]. As in the previous section, weak learners were regressors on example coordinates. All classifiers were trained with 100 iterations.

Table 1 lists the performances of the different methods. Comparing SOP-Boost with Lin-Boost, the former produced more accurate classifiers in 24 datasets out of 28. In fact, Lin-Boost had better performance only on "Twonorm", and only about $1 - 2\%$ improvement. In some of the other datasets, SOP-Boost produced very significant gains over Lin-Boost, e.g. 47.1% vs 11.7% on "Banana". Comparing POS-Boost to Lin-Boost, the former had better rates in 4 datasets, while Lin-Boost was more effective in 15. In 9 datasets the two methods had the same accuracy. Overall, SOP-Boost had the best performance among the three methods, the next best was Lin-Boost, and the third POS-Boost.

The weaker performance of POS-Boost is probably due to its lack of guarantee of a globally optimal solution. While the optimization problems of the first two methods are convex, that of POS-Boost is not. Although this was not a problem for XOR data, there may be many more local minima in the UCI problems, on which POS-Boost could get trapped. On the other hand, because SOP-Boost solves a convex optimization problem, it guarantees convergence to the optimal solution. Its superiority over Lin-Boost is explained by the fact that $\Omega_{\mathcal{G}} \subset \Omega_{\mathcal{G}}^{sop}$, i.e. SOP-Boost searches a (much) larger space of predictors.

We next compared the performance of SOP-Boost and implementations of Lin-Boost with various weak learners. These experiments were based on the exponential loss, second order TaylorBoost, and 100 training iterations. Table 2 compares the results of SOP-Boost+regression weak learners and Lin-Boost with weak learners based on regression, decision stumps, and decision trees of depth 3 and 5. For Lin-Boost, the more complex trees lead to the same or worse performance in $5/7$ datasets. Overall, SOP-Boost had the best performance in $4/7$ datasets. In a pairwise comparison, it beats Lin-Boost with regression learners on $6/7$ datasets, Lin-Boost+stumps on $5/7$ (one tie), Lin-Boost+tree(3) on $4/7$, and Lin-Boost+tree(5) on $5/7$.

These results show that SOP-Boost with regression weak

[1] It is possible to use stronger weak learners with regular boosting to address this problem. A comparison to these methods is presented in Section 4.2

Table 1. The average error rate (%) of different Boosting methods on various data sets using different loss functions

	Exp			*Log*			*CBL*			*Lap*		
	Lin	SOP	POS	Lin	SOP	POS	Lin	SOP	POS	Lin	SOP	POS
Banana	$47.1 \pm .5$	$11.7 \pm .1$	$22.4 \pm .5$	$47.0 \pm .4$	$\mathbf{11.0 \pm .1}$	$20.9 \pm .5$	$47.0 \pm .4$	$11.1 \pm .1$	$21.1 \pm .3$	$47.1 \pm .5$	$11.1 \pm .1$	$21.4 \pm .3$
Titanic	$22.7 \pm .1$	$22.4 \pm .1$	$23.4 \pm .4$	$22.7 \pm .1$	$22.4 \pm .1$	$23.0 \pm .4$	$22.7 \pm .1$	$22.4 \pm .1$	$23.7 \pm .5$	$22.7 \pm .1$	$\mathbf{22.3 \pm .1}$	$23.6 \pm .5$
Waveform	$13.8 \pm .1$	$\mathbf{12.9 \pm .1}$	$13.8 \pm .1$	$13.5 \pm .1$	$13.2 \pm .1$	$13.5 \pm .1$	$13.5 \pm .1$	$13.1 \pm .1$	$13.5 \pm .1$	$13.5 \pm .1$	$\mathbf{12.9 \pm .1}$	$13.5 \pm .1$
Twonorm	$\mathbf{3.6 \pm .0}$	$4.6 \pm .1$	$\mathbf{3.6 \pm .0}$	$\mathbf{3.6 \pm .0}$	$4.5 \pm .1$	$\mathbf{3.6 \pm .0}$	$3.6 \pm .0$	$5.1 \pm .1$	$\mathbf{3.6 \pm .0}$	$\mathbf{3.6 \pm .0}$	$4.6 \pm .1$	$\mathbf{3.6 \pm .0}$
Thyroid	$11.3 \pm .3$	$\mathbf{5.4 \pm .3}$	$15.8 \pm .4$	$10.5 \pm .2$	$5.7 \pm .3$	$14.6 \pm .4$	$10.2 \pm .3$	$6.7 \pm .4$	$14.6 \pm .4$	$10.4 \pm .3$	$5.7 \pm .3$	$13.4 \pm .5$
Splice	$16.4 \pm .2$	$7.9 \pm .2$	$16.6 \pm .1$	$16.2 \pm .2$	$8.0 \pm .2$	$16.4 \pm .2$	$16.3 \pm .2$	$\mathbf{7.8 \pm .2}$	$16.5 \pm .2$	$16.3 \pm .2$	$7.9 \pm .2$	$16.3 \pm .2$
Ringnorm	$26.3 \pm .1$	$\mathbf{6.8 \pm .1}$	$33.8 \pm .7$	$25.3 \pm .1$	$7.6 \pm .1$	$34.7 \pm .7$	$25.4 \pm .1$	$7.6 \pm .1$	$34.9 \pm .7$	$25.4 \pm .1$	$7.5 \pm .1$	$32.4 \pm .7$

Table 2. The average error rate (%) of different Boosting methods on various data sets using different type of weak learners

	Lin-Boost				SOP-Boost
	regression	stump	tree(3)	tree(5)	regression
Banana	$47.1 \pm .5$	$21.9 \pm .2$	$13.1 \pm .1$	$13.4 \pm .1$	$\mathbf{11.7 \pm .1}$
Titanic	$22.7 \pm .1$	$23.5 \pm .2$	$25.1 \pm .3$	$44.2 \pm .6$	$\mathbf{22.4 \pm .1}$
Waveform	$13.8 \pm .1$	$12.9 \pm .1$	$11.7 \pm .1$	$\mathbf{11.3 \pm .1}$	$12.9 \pm .1$
Twonorm	$\mathbf{3.6 \pm 0}$	4.7 ± 0	4.1 ± 0	4.0 ± 0	$4.6 \pm .1$
Thyroid	$11.3 \pm .3$	$7.6 \pm .3$	$8.5 \pm .8$	$12 \pm .4$	$\mathbf{5.4 \pm .3}$
Splice	$16.4 \pm .2$	$\mathbf{7.0 \pm .1}$	$7.6 \pm .9$	$33.1 \pm .6$	$7.9 \pm .2$
Ringnorm	$26.3 \pm .1$	$8.6 \pm .1$	$8.2 \pm .1$	$8.2 \pm .2$	$\mathbf{6.8 \pm .1}$

learners has a better bias-variance trade-off than the variants of Lin-Boost. In particular, Lin-Boost+regressors has too much bias, which decreases with stumps and trees of depth 3. This method has the best overall performance among Lin-Boost variants. With trees of depth 5, Lin-Boost has too much variance and starts to severely over-fit in some datasets. It should be said, however, that for some datasets the over-fitting starts to be visible even for stumps, see e.g. the second row of table 2, where increasing tree depth results in higher error rates. In SOP-Boost, the base learners are simple regressors, which are used first and latter combined into higher order terms *if and only if* this leads to better performance. This introduces some "resistance" to unduly complex weak learners, which increase the risk of over-training.

In summary, SOP-Boost has the best overall performance because 1) it builds the complex combinations of weak learners needed for accurate detection, but 2) only uses such combinations when truly necessary, reducing the risk of over-fitting.

4.3. Object detection

In this section, we report on experiments involving the detection of faces, cars, and pedestrians. In all cases, the features were Haar wavelets. The face dataset contains $9,000$ face and $9,000$ non-face images, of size 24×24. The car data is based on the UIUC dataset [2] of $1,100$ positives and $10,000$ negatives, of size 20×50. The pedestrian data is based on the MIT Pedestrian dataset [14] of $1,000$ positives and $10,000$ negatives, of size 40×20. In all cases, the data was split into five folds, four of which were used for training and one for testing. All experiments were repeated with each fold taking the role of test set, and the results averaged.

Table 3. The average error rate (%) of different Boosting methods on face, car and pedestrian detection data sets.

	Face	Car	Pedestrian
AdaBoost	5.7 ± 0.1	3.3 ± 1.9	3.9 ± 0.1
GentleBoost	9.0 ± 0.3	2.5 ± 0.3	6.9 ± 0.2
LogitBoost	8.9 ± 0.3	2.2 ± 0.3	5.4 ± 0.2
SOP+Exp Loss	5.1 ± 0.3	1.7 ± 0.3	3.4 ± 0.3
SOP+Log Loss	$\mathbf{4.8 \pm 0.3}$	$\mathbf{1.5 \pm 0.3}$	$\mathbf{2.9 \pm 0.2}$

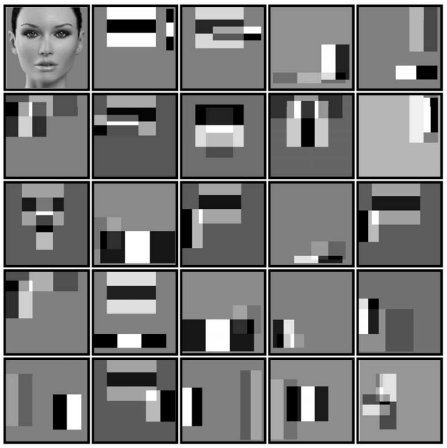

Figure 3. Examples of features selected by SOP-Boost.

Table 3 compares performance of AdaBoost, GentleBoost, LogitBoost, and the combination of SOP-Boost (implemented with second order TaylorBoost) and two loss functions: exponential (also used by AdaBoost and GentleBoost) and logistic (by LogitBoost). The SOP+logistic combination had the higher detection rates in all three tasks, followed by SOP+exponential. In pairwise comparisons, the error rate of SOP+logistic was frequently close to half of that of the classical boosting methods. This is partly explained by the sophistication of the features learned by SOP-Boost. Some of the features learned for face detection are shown in Figure 3. The learned features are rather intuitive, mostly detectors of faces parts, such as eyes, mouth, or ears, in certain relative positions. This enables the rejection of face-like false-positive with a few feature evaluations.

Table 4. Classification accuracy for 15 scene categories.

Method	Accuracy%
Liu et al. [9]	63.32
Lazebnik et al. [8]	72.2
Rasiwasia et al. [16]	72.5
Liu et al. [9]	75.16
Rasiwasia et al. [15]	72.2
AdaBoost [10]	74.79
TangentBoost [10]	76.24
SOP-Boost+Exp loss	**76.96**

4.4. Scene Classification

We finish with results on scene classification using the 15-scene dataset. The most popular classifier for this problem is the SVM classifier of visual word histograms proposed in [8]. There are many variants of this method that implement spatial encodings, multiple kernels, different codebook learning methods, etc. Since our goal was to evaluate the performance of the learning algorithms, we present a comparison based on the basic visual word histogram, without any of these extensions. Good results on this problem have been reported for generative classifiers that represent images as vectors of posterior probabilities under the 15-classes. These probabilities can be computed with Gaussian [15] or Dirichlet [16] mixture models. Among discriminant methods, SVMs [15, 16], and various versions of boosting have also been used. Best current results are due to the TangentBoost method of [10]. As shown in Table 4, the combination of SOP-Boost and the exponential loss outperforms all previous methods. The comparison of the last four rows of the table is particularly interesting, because all these classifiers use as input the vectors of posterior probabilities under a Gaussian mixture. The only difference is the algorithm used to learn the final classification boundary, which is an SVM, AdaBoost, TangentBoost and SOP-Boost respectively.

5. Conclusion

In this work, we considered the problem of simultaneous feature creation and selection in classifier design. The proposed solution is based on boosting and a pool of simple features. At each boosting iteration, it searches for both the best feature to add to the current ensemble and the best possible combination of current and new features. This allows boosting to create weak learner combinations that are more sophisticated than the classical weighted sum. In particular, we have proposed algorithms for learning sums of weak learner products, SOP-Boost, and products of weak learners sums, POS-Boost. Extensive experiments on synthetic data, UCI data, and data from object detection and scene classification, has shown the superior performance of SOP-Boost over previous boosting methods.

References

[1] http://theoval.cmp.uea.ac.uk/ gcc/matlab/index.shtml. 6

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. PAMI*, 26:1475 –1490, 2004. 7

[3] O. Danielsson, B. Rasolzadeh, and S. Carlsson. Gated classifiers: Boosting under high intra-class variation. In *CVPR*, pages 2673 –2680, 2011. 2, 5

[4] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010. 1

[5] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995. 6

[6] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000. 2, 6

[7] B. Kégl and R. Busa-Fekete. Boosting products of base classifiers. In *ICML*, pages 497–504, 2009. 2, 5

[8] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006. 8

[9] J. Liu and M. Shah. Scene modeling using co-clustering. In *ICCV*, 2007. 8

[10] H. Masnadi-Shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *CVPR*, 2010. 8

[11] H. Masnadi-Shirazi and N. Vasconcelos. Variable margin losses for classifier design. In *NIPS*, 2010. 6

[12] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *NIPS*, 2000. 2

[13] T. Mita, T. Kaneko, B. Stenger, and O. Hori. Discriminative feature co-occurrence selection for object detection. *IEEE Trans. PAMI*, 30(7):1257 –1269, july 2008. 2

[14] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *CVPR*, pages 193–99, 1997. 7

[15] N. Rasiwasia and N. Vasconcelos. Scene classification with low-dimensional semantic spaces and weak supervision. In *CVPR*, 2008. 8

[16] N. Rasiwasia and N. Vasconcelos. Holistic context modeling using semantic co-occurrences. In *CVPR*, 2009. 8

[17] Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999. 1

[18] M. J. Saberian, H. Masnadi-Shirazi, and N. Vasconcelos. Taylorboost: First and second order boosting algorithms with explicit margin control. In *CVPR*, 2011. 2, 3, 6

[19] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 5(7):682–687, 2002. 1

[20] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004. 1, 2