Boosting Invariance and Efficiency in Supervised Learning

Andrea Vedaldi
Computer Science Dept.
UCLA, USA
vedaldi@ucla.edu
http://vision.ucla.edu/˜vedaldi

Paolo Favaro
Dept. of Electr. Eng. and Physics
Heriot-Watt University, UK
p.favaro@hw.ac.uk
http://www.eps.hw.ac.uk/˜pf21

Enrico Grisan
Dept. of Biomedical Eng.
University of Padua, Italy
enrico.grisan@unipd.it

Abstract

In this paper we present a novel boosting algorithm for supervised learning that incorporates invariance to data transformations and has high generalization capabilities. While one can incorporate invariance by adding virtual samples to the data (e.g., by jittering), we adopt a much more efficient strategy and work along the lines of vicinal risk minimization and tangent distance methods. As in vicinal risk minimization, we incorporate invariance to data by applying anisotropic smoothing along the directions of invariance. Moreover, as in tangent distance methods, we provide a simple local approximation to such directions, thus obtaining an efficient computational scheme. We also show that it is possible to automatically design optimal weak classifiers by using gradient descent. To increase efficiency at run time, such optimal weak classifiers are projected on a Haar basis. This results in designing strong classifiers that are more computationally efficient than in the case of exhaustive search. For illustration and validation purposes, we demonstrate the novel algorithm both on synthetic and on real data sets that are publicly available.

1. Introduction

In most computer vision applications we are interested in classifying objects despite changes in their scale, orientation, and location, and in their photometry, due to varying illumination and noise [6, 9, 10, 11, 17, 18]. For example, in a surveillance system one may be interested in detecting faces regardless of whether they face the camera, are close or far from it, or lie in the shade. One way to address such classification problem is *to incorporate invariance to these changes at run-time*. For instance, one can apply an object classifier to any possible transformation of the input images, and then design a strategy to select an answer among all responses. However, transformations may be continuous in nature, such as in the above examples, so that testing all of them is impossible. As the number of independent transformations grows, even discretization may be not sufficient to reduce the required tests to a workable number. An alternative is *to incorporate invariance during training*. One can enlarge the training set by adding the so-called *virtual samples*, i.e., samples obtained by morphing the original data set for a finite set of transformations [7, 13]. This approach is more effective at run-time than the previous one, but renders the training phase extremely slow and memory inefficient. A third alternative is *to incorporate invariance in the classification error function* used for training: Given a sample, one can compute analytically the error of a classifier against a distribution of a set of transformations. This is, for example, the approach used in *Vicinal Risk Minimization* (VRM) [2, 13].

In this paper we also incorporate invariance in the classification error function, because this has the advantage of being efficient both at run-time and during training. We attach probability distributions to transformations that are local rather than global, as the extrapolation of samples for large transformations may not be reliable. This allows us to introduce tangent vector methods [13, 16] in our algorithm. To further limit the amount of computations, we introduce two additional elements: gradient descent and Haar wavelet projection. Our gradient descent procedure allows to find good features automatically and efficiently so that exhaustive search over large databases can be avoided. Gradient methods have been also suggested in a scale-space scheme in VRM, but not in a boosting framework [2]. Haar wavelet projection allows to dramatically reduce the amount of computations at test time by mapping each feature to a finite set of Haar wavelets and therefore enabling the use of integral images [12, 20]. We develop this method in the boosting framework, which, to the best of our knowledge, has never been done before. Notice, however, that invariance to transformations has been incorporated before, for instance, in support vector machines [15].

In the next section we revisit the basics of the binary classification problem; then, in sec. 3 we introduce the tangent space approach and, finally, in sec. 4 we present the Parzen-AdaBoost approach.

2. Binary Classification

For the sake of simplicity, in this paper we discuss only the binary classification problem. Extensions to multiple classes can be easily obtained, for instance, by using AdaBoost.MH [14]. In binary classification we are given a collection of N i.i.d.[1] observations $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$ and labels $y_1, \ldots, y_N \in \{-1, +1\}$. The goal is to design a function $H : \mathcal{X} \mapsto \{-1, +1\}$, a *classifier*, that predicts the label y of a generic observation $\mathbf{x}$. As we deal with images, $\mathcal{X} \subseteq \mathbb{R}^d$, where d is the number of pixels of the image. We denote the unknown joint probability distribution density of $(\mathbf{x}, y)$ with $p(\mathbf{x}, y)$ and the *expectation* of a random variable w with respect to p with $E_p[w]$.

The *optimal classifier* H is the one that minimizes the so-called *01-loss expected risk*

$$\begin{aligned} \mathrm{Err}_p(H) &\doteq E_p[I(H(\mathbf{x}) \neq y)] \\ &= \sum_{y=\{-1,+1\}} \int p(\mathbf{x}, y) I(H(\mathbf{x}) \neq y)\, d\mathbf{x} \end{aligned} \tag{1}$$

where $I(\mathcal{A})$ is the indicator function of the event $\mathcal{A}$. Unfortunately, the optimal classifier cannot be computed as eq. (1) requires the distribution $p(\mathbf{x}, y)$, which is unknown. The common strategy is then to approximate $p(\mathbf{x}, y)$ by using the available set of samples $\{(\mathbf{x}_i, y_i),\ i = 1, \ldots, N\}$. One such approximation is to replace the true distribution $p(\mathbf{x}, y)$ with the *empirical distribution*

$$\hat{p}_e(\mathbf{x}, y) = \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i)\delta(y - y_i) \tag{2}$$

where $\delta(\mathbf{x})$ and $\delta(y)$ are Dirac's deltas. This choice leads to the minimization of the *empirical loss*[2]

$$\mathrm{Err}_{\hat{p}_e}(H) \propto -\frac{1}{N} \sum_{i=1}^{N} y_i H(\mathbf{x}_i). \tag{3}$$

Alternatively to eq. (2), one can consider *Parzen's windows* [3]

$$\hat{p}_g(\mathbf{x}, y) = \frac{1}{N} \sum_{i=1}^{N} g_\Sigma(\mathbf{x} - \mathbf{x}_i)\delta(y - y_i) \tag{4}$$

where $g_\Sigma(\mathbf{x})$ denotes a zero-mean Gaussian distribution with covariance Σ. In this case, the classification error is

$$\begin{aligned} \mathrm{Err}_{\hat{p}_g}(H) &\propto E_{\hat{p}_g}[-yH(\mathbf{x})] \\ &= -\frac{1}{N} \sum_{i=1}^{N} \int g_\Sigma(\mathbf{x} - \mathbf{x}_i) y_i H(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{5}$$

[1]The notation i.i.d. stands for *independent and identically distributed*.

[2]In our notation $A(\mathbf{x}) \propto B(\mathbf{x})$ if and only if there exist constants $a > 0, b$ such that $A(\mathbf{x}) = aB(\mathbf{x}) + b$.

An interesting observation in [2] that we exploit in our algorithm is that Parzen's windows method can be formulated as the empirical loss of a smoothed classifier H, i.e.,

$$\mathrm{Err}_{\hat{p}_g}(H) = \mathrm{Err}_{\hat{p}_e}(g_\Sigma * H). \tag{6}$$

2.1. Discrete AdaBoost

In boosting one builds a classifier H by combining additively several so-called *weak classifiers* [5, 8]. The key idea is that, as long as the weak classifiers do better than chance, it is possible to *boost* their performance by combining them linearly.

In Discrete AdaBoost M weak classifiers $f_m : \mathcal{X} \mapsto \{-1, +1\}$ are combined to yield an *auxiliary function* F_M and the corresponding *strong classifier* H_M

$$F_M(\mathbf{x}) = \textstyle\sum_{m=1}^{M} c_m f_m(\mathbf{x}), \quad H_M(\mathbf{x}) = \mathrm{sign}\,(F_M(\mathbf{x})) \tag{7}$$

with parameters $c_1, \ldots, c_M \in \mathbb{R}$. Rather than directly minimizing the empirical error (3), AdaBoost minimizes the *exponential loss*

$$E_{\hat{p}_e}[e^{-yF_M(\mathbf{x})}] \tag{8}$$

which bounds from above the empirical error (3) as $e^{-yF_M(\mathbf{x})} \geq I(H_M(\mathbf{x}) \neq y),\ \forall M$. To limit the computational burden the auxiliary function F_M is built iteratively. Given F_{m-1} one searches for the optimal update $c_m f_m$ such that the exponential loss (8) is minimized. Every iteration can be written recursively by means of weights $w(\mathbf{x}, y) = e^{-yF_{m-1}(\mathbf{x})}$, which automatically concentrate the error on the difficult samples. The algorithm is summarized in Algorithm 1.

3. Invariance and Tangent Spaces

As mentioned in the Introduction, in many computer vision applications one is interested in classifying objects in images irrespectively of translations $t \in \mathbb{R}^2$, rotations[3] $R \in SO(2)$, scalings $s \in [0, +\infty)$, and changes in intensity due to contrast $b \in [0, +\infty)$. Let $\alpha = [t\ R\ s\ b] \in \mathcal{L}$ be a vector lumping all the transformation parameters. Then, given a sample $\mathbf{x}$, the transformed sample $\mathbf{x}_\alpha$ is defined as

$$\mathbf{x}_\alpha \doteq T(\mathbf{x}, \alpha) \tag{9}$$

where $T : \mathcal{X} \times \mathcal{L} \mapsto \mathcal{X}$ is the *morphing function* defined via

$$T(\mathbf{x}, \alpha)(x) \doteq b\mathbf{x}(sRx + t) \tag{10}$$

and $x \in \mathbb{R}^2$ denotes the 2-D coordinates of $\mathbf{x}$. Let $p(\alpha)$ be the probability density of a transformation α. Then, one could incorporate invariance to transformations in the empirical error as follows

$$\mathrm{Err}^{inv}_{\hat{p}_e}(H) \propto -\frac{1}{N} \sum_{i=1}^{N} \int p(\alpha) y_i H(T(\mathbf{x}_i, \alpha)) d\alpha. \tag{11}$$

[3]$SO(2)$ denotes the *special orthogonal* group of 2-D rotation matrices.

Algorithm 1 Discrete AdaBoost

```
1: Initialize $F_0(\mathbf{x}) = 0$ for all $x \in \mathcal{X}$.
2: Initialize $w(\mathbf{x}_i, y_i) = 1/N$ for all $i = 1, 2, \ldots, N$.
3: for $m = 1$ to $M$ do
4:     Find the weak classifier $f_m \in \mathcal{F}$ that minimizes
           $$\mathrm{Err}_q(f) \propto \sum_{i=1}^{N} w(\mathbf{x}_i, y_i) I(f(\mathbf{x}_i) \neq y_i)$$
       where the distribution $q$ is defined as
           $$q(\mathbf{x}, y) \propto \sum_{i=1}^{N} w(\mathbf{x}, y)\delta(\mathbf{x} - \mathbf{x}_i)\delta(y - y_i);$$
5:     Let
           $$c_m \leftarrow \frac{1}{2}\log\frac{1 - \mathrm{Err}_q(f_m)}{\mathrm{Err}_q(f_m)};$$
6:     Update the weights
           $$w(\mathbf{x}_i, y_i) \leftarrow w(\mathbf{x}_i, y_i)e^{-y_i c_m f_m(\mathbf{x}_i)};$$
7:     Update the auxiliary function $F_m = F_{m-1} + c_m f_m$.
8: end for
```

Virtual samples can be easily generated by letting $p(\alpha) = \sum_{j=1}^{K}\delta(\alpha - \alpha_j)$ for a certain set of transformations $\{\alpha_j\}_{j=1,2,\ldots,K}$. However, this has the immediate effect of multiplying the size of the data set of the samples by a factor K. Moreover, notice that virtual samples generated for large transformations may not reliably substitute real samples due to missing data, sampling, and quantization. This is the case, for instance, when we scale, translate, or rotate an image. Therefore, we consider incorporating invariance to transformations only locally at each sample. To do so, we use the tangent vector approach [16], i.e., we approximate the global transformation at each sample as

$$T(\mathbf{x}, \alpha) \simeq \mathbf{x} + \sum_{k=1}^{K} L_k(\mathbf{x}, \alpha_k^0)(\alpha_k - \alpha_k^0) \tag{12}$$

where α^0 is the identity transformation, which we can assume identically zero without loss of generality, and $L_k : \mathcal{X} \times \mathcal{L} \mapsto \mathcal{X}$ are local transformations defined as

$$L_k(\mathbf{x}, \alpha^0) \doteq \frac{\partial T}{\partial \alpha_k}(\mathbf{x}, \alpha)\bigg|_{\alpha=\mathbf{0}}. \tag{13}$$

Note that L_k are operators that generate the whole space of local transformations (a Lie algebra of local transformations). Such operators can be computed analytically or, more easily, by using finite differences. Finally, to enforce locality we assume that the prior $p(\alpha)$ is Gaussian with mean $\alpha_0 = \mathbf{0}$ and diagonal covariance matrix Ψ.[4] By substituting this prior in eq. (11) and by using the tangent vector approximation we obtain

$$\mathrm{Err}_{\hat{p}_e}^{inv}(H) \propto -\frac{1}{N}\sum_{i=1}^{N} y_i \left(g_{\Sigma_i} * H\right)(\mathbf{x}_i). \tag{14}$$

where $\Sigma_i = L(\mathbf{x}_i, \mathbf{0})\Psi L(\mathbf{x}_i, \mathbf{0})^T$ and we have defined $L(\mathbf{x}_i, \mathbf{0}) = [L_1(\mathbf{x}_i, 0)\ L_2(\mathbf{x}_i, 0)\ \ldots\ L_K(\mathbf{x}_i, 0)]$. Notice that the above equation can also be readily interpreted as Parzen's windows error where the covariance of the Gaussian kernel in eq. (3) is $\Sigma = \Sigma_i$, i.e.,

$$\mathrm{Err}_{\hat{p}_e}^{inv}(H) = \mathrm{Err}_{\hat{p}_e}(g_\Sigma * H) = \mathrm{Err}_{\hat{p}_g}(H). \tag{15}$$

Let us now consider the case of additive Gaussian noise $w \sim p(w)$. This case is particularly interesting as it corresponds to no a-priori knowledge where every pixel of the image $\mathbf{x}$ is affected by an unknown disturbance. In this case, the tangent vectors cover the whole space $\mathcal{X}$ and Σ becomes a diagonal matrix, thus yielding *isotropic smoothing*. In other words, when samples are affected by additive Gaussian noise that is independent at each pixel, virtual samples lie in a sphere around the original image sample. This has the effect of increasing the classifier margin [19].

4. Parzen-AdaBoost

So far, AdaBoost has been based on the empirical distribution in eq. (2). We now look at the extension of AdaBoost to Parzen's windows eq. (4) because, as we have seen in sec. 3, it allows us to incorporate invariance to a prescribed set of transformations.

Similarly to Discrete AdaBoost, our algorithm is based on the following inequality

$$\frac{1}{2}E_{\hat{p}_e}\left[1 - y\mathrm{sign}(g_\Sigma * F(\mathbf{x}))\right] \leq E_{\hat{p}_e}[e^{-yg_\Sigma * F(\mathbf{x})}]. \tag{16}$$

The auxiliary function F of a strong classifier $H(\mathbf{x}) = \mathrm{sign}\ F(\mathbf{x})$ is written as a summation $F_M = \sum_{m=1}^{M} c_m f_m$, so that $g_\Sigma * F_M = \sum_{m=1}^{M} c_m\left(g_\Sigma * f_m\right)$ and this corresponds to smoothing each weak classifier. This approach has three advantages: First, eq. (16) guarantees that by minimizing the right-hand side one improves the empirical error eq. (3) of the strong classifier $\mathrm{sign}(g_\Sigma * F)$. Second, by selecting weak classifiers based on eq. (16) one automatically incorporates invariance.[5] Third, the minimization of

[4] One way to estimate Ψ is, for example, via cross-validation. However, in our experiments we manually fix Ψ to the maximum amount possible and within the limits imposed by linearization.

[5] Later, we will see that the strong classifier F approximately minimized Parzen's loss (6). When F is a single weak classifier of the form $F(\mathbf{x}) = c\,\mathrm{sign}(\gamma_1^T\mathbf{x} + \gamma_0)$ the bound $E_{\hat{p}_e}\left[1 - y\mathrm{sign}(g_\Sigma * F(\mathbf{x}))\right]/2 \leq E_{\hat{p}_e}[e^{-yg_\Sigma * F(\mathbf{x})}]$ is guaranteed for any $c < 2.678$. When F consists of more than one weak classifier, we assume that only one weak classifier is changing sign "close to" a sample. In this context, the notion of distance from a sample is based on Σ. In this case, we have that $F(\mathbf{x}) = c\,\mathrm{sign}(\gamma_1^T\mathbf{x} + \gamma_0) + \theta$ where θ is constant (near a sample) and collects all the decisions from the other weak classifiers. The above bound is again guaranteed for $c < 0.693$.

eq. (16) results in a very efficient algorithm, as few weak classifiers are required.[6] We call this novel method *Parzen-AdaBoost*.

Our strategy is to find a recursive iteration along the lines of Discrete AdaBoost to minimize Parzen's windows loss in eq. (5). Similarly to Discrete AdaBoost we have a strong classifier $H(\mathbf{x}) = \text{sign}(g_\Sigma * F_M(\mathbf{x}))$, where $F_M(\mathbf{x}) = \sum_{m=1}^M c_m f_m(\mathbf{x})$, and define Parzen's windows exponential loss as

$$E_{\hat{p}_e}\left[e^{-y\ g_\Sigma * F_M(\mathbf{x})}\right]. \tag{17}$$

Then, given a classifier F_{m-1} we search for the optimal update $c_m f_m$ such that eq. (16) is minimized. Thanks to the exponential form, we can separate the update from the current classifier so that

$$E_{\hat{p}_e}\left[e^{-y\ g_\Sigma * (F_{m-1}+c_m f_m)(\mathbf{x})}\right] = E_q\left[e^{-y\ c_m g_\Sigma * f_m(\mathbf{x})}\right] \tag{18}$$

where

$$\begin{array}{rcl} q(\mathbf{x},y) & = & \frac{1}{N}\sum_{i=1}^N w(\mathbf{x}_i,y_i)\delta(\mathbf{x}-\mathbf{x}_i)\delta(y-y_i) \\ w(\mathbf{x}_i,y_i) & = & e^{-y_i\ g_{\Sigma_i} * F_{m-1}(\mathbf{x}_i)}. \end{array} \tag{19}$$

Thus, at the m-th iteration we need to minimize

$$\sum_{i=1}^N w(\mathbf{x}_i,y_i) e^{-y_i c_m g_\Sigma * f_m(\mathbf{x}_i)} \tag{20}$$

with respect to f_m and c_m. Finding a closed form solution for this minimization problem is not straightforward. In this paper we follow an approach proposed by Friedman [4] and consider the minimization of eq. (20) in function space. First, we compute the derivative of eq. (20) along an arbitrary weak classifier $h : \mathcal{X} \mapsto \mathbb{R}$. This yields

$$\lim_{\epsilon \to 0} \frac{E_q\left[e^{-y\ g_\Sigma * \epsilon h(\mathbf{x})}\right] - E_q\left[e^0\right]}{\epsilon} = -E_q\left[y\ g_\Sigma * h(\mathbf{x})\right]. \tag{21}$$

Then we search for the weak classifier f_m that results in the steepest descent, i.e., that maximizes

$$E_q\left[y\ g_\Sigma * f_m(\mathbf{x})\right] \propto -\text{Err}_q(f_m) \tag{22}$$

where

$$\text{Err}_q(f_m) = E_q\left[\frac{1 - y\ g_\Sigma * f_m(\mathbf{x})}{2}\right]. \tag{23}$$

Thus we are searching for the weak classifier f_m that, after smoothing, has the minimum weighed risk. Once f_m has been chosen, we need to compute the optimal step c_m. To this end, we consider the first and second derivative of eq. (20) with respect to c_m, i.e.,

$$\begin{array}{rcl} \frac{\partial}{\partial c_m} E_q\left[e^{-y\ c_m g_\Sigma * f_m(\mathbf{x})}\right] & = & E_{q'}[-y\ g_\Sigma * f_m(\mathbf{x})] \\ \frac{\partial^2}{\partial c_m^2} E_q\left[e^{-y\ c_m g_\Sigma * f_m(\mathbf{x})}\right] & = & E_{q'}[(g_\Sigma * f_m)^2(\mathbf{x})] \end{array} \tag{24}$$

where $q'(\mathbf{x}) = q(\mathbf{x})e^{-y\ c_m g_\Sigma * f_m(\mathbf{x})}$ is the iteratively updated weighed distribution. Thus, starting from

$$c_m \leftarrow \frac{1}{2}\log\frac{1-\text{Err}_{q'}(f_m)}{\text{Err}_{q'}(f_m)} = \frac{1}{2}\log\frac{1+E_{q'}[y\ g_\Sigma * f_m(\mathbf{x})]}{1-E_{q'}[y\ g_\Sigma * f_m(\mathbf{x})]} \tag{25}$$

we get the Gauss-Newton update

$$c_m \leftarrow \frac{E_{q'}[y\ g_\Sigma * f_m(\mathbf{x})]}{E_{q'}[(g_\Sigma * f_m)^2(\mathbf{x})]}. \tag{26}$$

The algorithm is summarized in Algorithm 2.

Algorithm 2 Parzen-AdaBoost

```
1: Initialize $F_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{X}$.
2: Initialize $w(\mathbf{x}_i, y_i) = 1/N$ for all $i = 1, 2, \dots, N$.
3: for $m = 1$ to $M$ do
4:    Search for the weak classifier $f_m \in \mathcal{F}$ that minimizes
      $\text{Err}_q(f_m)$ given in eq. (23), where $q(\mathbf{x}, y)$ is given in
      eq. (19). In alternative to the exhaustive search, use the
      gradient descent method described in sec. 4.2.
5:    Initialize $q' \leftarrow q$.
6:    Initialize
         $c_m \leftarrow \frac{1}{2}\log\frac{1-\text{Err}_{q'}(f_m)}{\text{Err}_{q'}(f_m)}$.   (27)
7:    while not converged do
8:       Calculate $E_{q'}[(g_\Sigma * f_m)^2(\mathbf{x})]$.
9:       Calculate $E_{q'}[y\ g_\Sigma * f_m(\mathbf{x})]$.
10:      Set $\delta \leftarrow E_{q'}[y\ g_\Sigma * f_m(\mathbf{x})]/E_{q'}[(g_\Sigma * f_m)^2(\mathbf{x})]$.
11:      Update $c_m \leftarrow c_m + \delta$.
12:      Update $q'(\mathbf{x}, y) \leftarrow q'(\mathbf{x}, y)e^{-\delta y\ g_\Sigma * f_m(\mathbf{x})}$.
13:   end while
14:   Update the auxiliary function $F_m \leftarrow F_{m-1} + c_m f_m$.
15: end for
```

4.1. Linear Classifiers

In the simplest instance of a classifier $H(\mathbf{x}) = \text{sign}\ f_1(\mathbf{x})$ the auxiliary function f_1 is linear, i.e.,

$$f_1(\mathbf{x}) = \text{sign}(\gamma_0 + \langle\gamma_1, \mathbf{x}\rangle) \tag{28}$$

where $\gamma_0 \in \mathbb{R}$ and $\gamma_1 \in \mathcal{X}$. Graphically, this corresponds to defining a hyper-plane that separates the space of the input images $\mathbf{x}$ into two complementary hyper-volumes. The vector γ_1 defines the normal to the hyper-plane. In practice, γ_1 can be rearranged as an image and be seen as a *feature*. As we will see in the later sections, we can approximate γ_1 with Haar wavelets and improve the computational efficiency of the classifier. In the case of Parzen's windows in

[6]This algorithm can also be cast as a Real-Boost method where we search for the optimal weak classifiers among the smooth and transformation invariant ones.

eq. (4) we immediately find that

$$(g_\Sigma * H)(\mathbf{x}) = \text{erf}\left(\frac{\gamma_0 + \langle\gamma_1, x\rangle}{\sqrt{2\gamma_1^\top \Sigma \gamma_1}}\right) \qquad (29)$$

and thus the approximate Parzen's windows loss eq. (5) becomes

$$\text{Err}_{p_w}(H) = \frac{1}{2} - \frac{1}{2N}\sum_{i=1}^{N} y_i \text{erf}\left(\frac{\gamma_0 + \langle\gamma_1, x_i\rangle}{\sqrt{2\gamma_1^T \Sigma_i \gamma_1}}\right) \qquad (30)$$

where erf is the *error function* and is defined as

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt. \qquad (31)$$

Tangent vectors can then be readily incorporated as suggested in sec. 3 by defining $\Sigma_i = L(\mathbf{x}_i, \mathbf{0})\Psi L(\mathbf{x}_i, \mathbf{0})^T$.

4.2. Optimizing Linear Weak Classifiers

The first step in Parzen-AdaBoost is to search for a weak classifier f_m that minimizes eq. (23). Typically, one defines a very large set of weak classifiers and then performs an exhaustive search to determine the optimal one. In addition to being rather time-consuming, this procedure yields cumbersome and computationally inefficient strong classifiers when the set of weak classifiers is not chosen purposefully. In this section we suggest a method to automatically design weak classifiers via a gradient descent procedure.

We restrict the weak classifiers to be linear so that, as shown in sec. 4.1,

$$f_m(\mathbf{x}) = \text{sign}\left(\gamma_0 + \langle\gamma_1, \mathbf{x}\rangle\right) \qquad (32)$$

where $\gamma_0 \in \mathbb{R}$ and $\gamma_1 \in \mathcal{X}$. The initialization of the parameters (γ_0, γ_1) is done by selecting a random vector γ_1 and then by a simple line search on the other parameter γ_0. In our algorithm, however, we implement a more efficient method for the initialization of γ_0 based on sorting the responses $\langle\gamma_1, \mathbf{x}_i\rangle$, that we do not report here for lack of space. Once the parameters have been initialized, we compute the gradient of

$$\Phi(\gamma_0, \gamma_1) \doteq \sum_{i=1}^{N} w(\mathbf{x}_i, y_i)\text{erf}\left(\frac{\gamma_0 + \langle\gamma_1, \mathbf{x}_i\rangle}{\sqrt{2\gamma_1^\top \Sigma_i \gamma_1}}\right). \qquad (33)$$

with respect to (γ_0, γ_1). This results in

$$\begin{aligned}\frac{\partial\Phi}{\partial\gamma_0} &= \textstyle\sum_{i=1}^{N} w(\mathbf{x}_i, y_i)\dot{\text{erf}}\left(\frac{\gamma_0+\langle\gamma_1,\mathbf{x}\rangle}{\sqrt{2\gamma_1^\top\Sigma_i\gamma_1}}\right)\frac{1}{\sqrt{2\gamma_1^\top\Sigma_i\gamma_1}}\\ \frac{\partial\Phi}{\partial\gamma_1^\top} &= \textstyle\sum_{i=1}^{N} w(\mathbf{x}_i, y_i)\dot{\text{erf}}\left(\frac{\gamma_0+\langle\gamma_1,\mathbf{x}\rangle}{\sqrt{2\gamma_1^\top\Sigma_i\gamma_1}}\right)\frac{1}{\sqrt{2\gamma_1^\top\Sigma_i\gamma_1}}\\ &\quad\cdot\left(\mathbf{x} - \frac{\gamma_0+\langle\gamma_1,\mathbf{x}\rangle}{\gamma_1^\top\Sigma_i\gamma_1}\Sigma_i\gamma_1\right)\end{aligned} \qquad (34)$$

where $\dot{\text{erf}}(z)$ denotes the derivative of $\text{erf}(z)$ with respect to z. Let $\nu_0 \doteq \frac{\partial\Phi}{\partial\gamma_0}$ and $\nu_1 \doteq \frac{\partial\Phi}{\partial\gamma_1}$ be the update directions of γ_0 and γ_1. Then, we can update the weak classifier via

$$\begin{aligned}\gamma_0 &\leftarrow \gamma_0 + \lambda\nu_0\\ \gamma_1 &\leftarrow \gamma_1 + \lambda\nu_1\end{aligned} \qquad (35)$$

given a step $\lambda > 0$. To determine the optimal update step λ, consider

$$\begin{aligned}&\Phi(\gamma_0 + \lambda\nu_0, \gamma_1 + \lambda\nu_1) =\\ &\qquad \textstyle\sum_{i=1}^{N} w(\mathbf{x}_i, y_i)\text{erf}\left(\frac{\beta_{1,i}+\lambda\beta_{2,i}}{\sqrt{2\beta_{3,i}+4\beta_{4,i}\lambda+2\beta_{5,i}\lambda^2}}\right)\end{aligned} \qquad (36)$$

where

$$\beta_{1,i} = \gamma_0 + \langle\gamma_1, \mathbf{x}_i\rangle \quad \beta_{2,i} = \nu_0 + \langle\nu_1, \mathbf{x}_i\rangle$$
$$\beta_{3,i} = \gamma_1^\top\Sigma_i\gamma_1 \qquad \beta_{4,i} = \nu_1^\top\Sigma_i\gamma_1 \qquad \beta_{5,i} = \nu_1^\top\Sigma_i\nu_1$$

These equations can be used to compute the energy function for several values of λ rather efficiently. However, to improve efficiency even further we perform a single Gauss-Newton step. Then, we compute the gradient and the Hessian of eq. (33) with respect to λ and evaluate them in $\lambda = 0$

$$\begin{aligned}\frac{\partial\Phi}{\partial\lambda}(0) &= \dot{\Phi}(\gamma_0, \gamma_1)\frac{\beta_{2,i}}{\beta_{3,i}^{1/2}} - \frac{\beta_{1,i}\beta_{4,i}}{\beta_{3,i}^{3/2}}\\ \frac{\partial^2\Phi}{\partial\lambda^2}(0) &= 2\dot{\Phi}(\gamma_0, \gamma_1)\left(\frac{\beta_{2,i}\beta_{4,i}}{\beta_{3,i}^{1/2}} - 3\frac{\beta_{1,i}\beta_{4,i}^2}{\beta_{3,i}^{5/2}} - \frac{\beta_{1,i}\beta_{5,i}}{\beta_{3,i}^{3/2}}\right)\end{aligned} \qquad (37)$$

where

$$\dot{\Phi}(\gamma_0, \gamma_1) \doteq \sum_{i=1}^{N} w(\mathbf{x}_i, y_i)\text{erf}\left(\frac{\beta_{1,i}}{\sqrt{2\beta_{3,i}}}\right) \qquad (38)$$

Finally, $\lambda \leftarrow -\frac{\partial\Phi}{\partial\lambda}(0)\left(\frac{\partial^2\Phi}{\partial\lambda^2}(0)\right)^{-1}$.

4.3. Projection onto Haar Wavelets

In this section we suggest a simple step to considerably speed up the performance of the classifier at run-time. So far we have been concerned with finding the weak classifiers f_m that minimize a certain exponential loss. We have not considered, however, that the computation of the inner product $\langle\gamma_1, \mathbf{x}\rangle$ is rather intensive for a generic vector γ_1. This is because each element in γ_1 needs to be multiplied by the corresponding element of the image $\mathbf{x}$. Nevertheless, if the elements of γ_1 lie in a rectangular region and have a constant value, the computations can be dramatically reduced by using the well-known *integral image* method [12, 20]. More in general, one can use Haar wavelets to compose several rectangular regions and obtain more advanced weak classifiers. In our algorithm, we *project* the weak classifier f_m by truncating its Haar wavelet decomposition. We insert this projection immediately before the update step of the auxiliary function F_m. In the next section we will see that only a few Haar wavelets are required to achieve the desired classification performance.

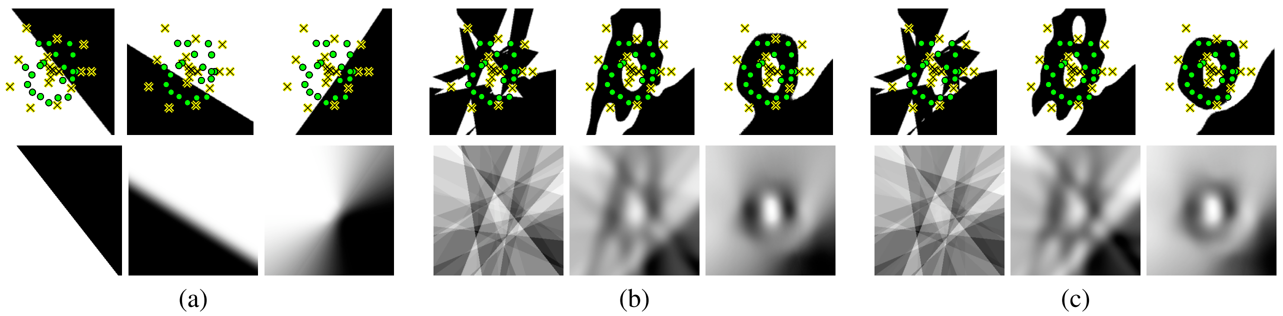
Figure 1. Smoothing and tangent vectors. Within each of the three groups (a), (b), and (c) we show the classification result of Discrete AdaBoost (left), AdaBoost with isotropic smoothing (middle), and AdaBoost with tangent vectors (right). Group (a) shows the classification result with 1 weak classifier, group (b) shows the classification results with 100 weak classifiers, and group (c) shows the classification results with 300 weak classifiers. The top row illustrates the decision region: White corresponds to the region where points are classified as crosses and black to the region where points are classified as circles. The bottom row shows the response of the auxiliary function F, i.e., before thresholding with sign. Notice that group (a) shows clearly the effect of introducing isotropic smoothing (middle image) and that of introducing an anisotropic smoothing (right image). Notice also how the standard AdaBoost method suffers from overfitting, while isotropic smoothing and, in particular, tangent vector methods do not.

5. Experiments

The proposed algorithm has been thoroughly tested on both synthetic and real data sets. In both cases we illustrate the effects of incorporating isotropic smoothing, gradient descent, tangent vectors, and Haar wavelet projection.

5.1. Synthetic Data Experiments

In Figure 1 we show a first experiment on two-dimensional (2-D) data to emphasize the efficacy of smoothing and the tangent vector method with few samples. The synthetic data is invariant to 2-D rotations. We divide the plots into 3 groups (a), (b), and (c) where: (a) corresponds to a strong classifier with only 1 weak classifier, (b) to 100 weak classifiers, and (c) to 300 weak classifiers. For each group the left image shows the classification result of Discrete AdaBoost, the middle image shows the result of AdaBoost with isotropic smoothing only, and the right image shows the result of AdaBoost with tangent vectors. The top row displays the decision region where white corresponds to points that are classified as crosses and black corresponds to points that are classified as circles. The bottom row shows the response of the auxiliary function F. One can immediately see that while Discrete AdaBoost suffers from overfitting, the other two methods can cope well with few samples. In particular, as this data is invariant to rotations, AdaBoost with tangent vectors obtains the best results.

In the second experiment, the synthetic data set consists of 24×24 pixels images of 4 shapes: a circle, a triangle, a star, and a square. To each shape we apply all the transformations listed in sec. 3 and, in addition, skewness. Some samples from each data set are shown in Figure 2.

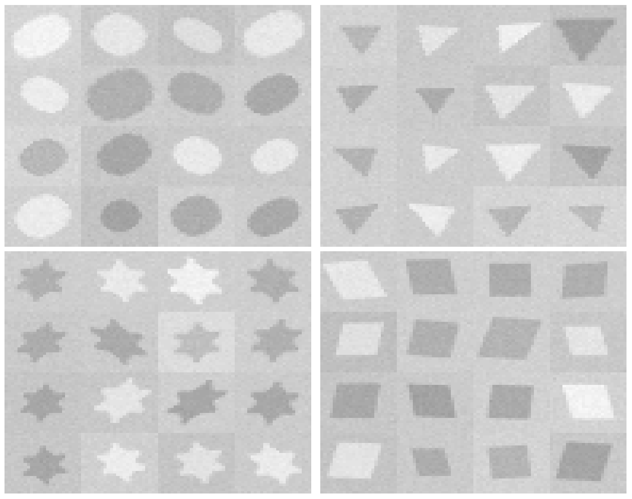
Figure 2. Some samples from the synthetic data set of the second experiment. Circles, triangles, stars, and squares are translated, rotated, scaled, skewed, undergo changes in contrast and brightness, and have additive Gaussian noise.

We lump triangles and squares into class 1 and stars and circles into class 2. Then, we train a strong classifier by changing the isotropic smoothing parameter, by enabling or not the gradient descent on weak classifiers, by incorporating or not the tangent vectors, and by changing the number of samples in the training set. The results of several combinations of these features are shown in Figures 3 and 4.

Notice that in Figure 3 the performance at run-time improves when tangent vectors are used and even more when gradient descent is enabled. All experiments share the same data set and the same initial set of weak classifiers. Furthermore, notice how the proposed algorithm can cope well with few data samples in the training set as the performance with 25 elements yields a 10% test error. As the test error converges almost immediately when the proposed method is used, only a few weak classifiers are needed to achieve

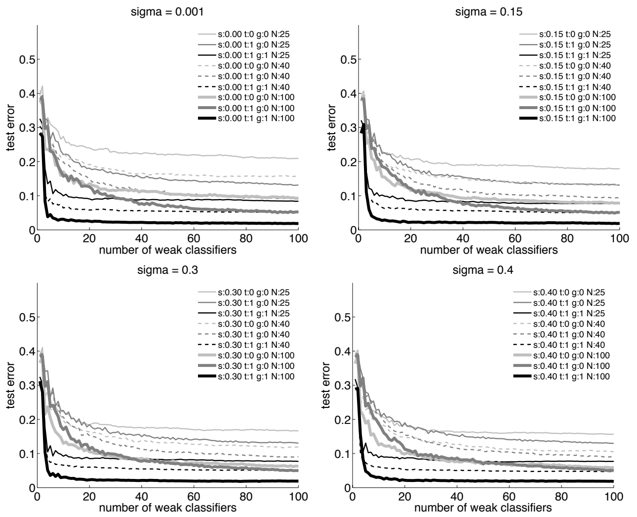

Figure 3. Classification results on synthetic data. The ordinate axis shows the test error of the proposed AdaBoost method for several configurations of number of samples in the training data set (N), usage of gradient descent (g), usage of the tangent vectors (t), and level of isotropic smoothing (s).

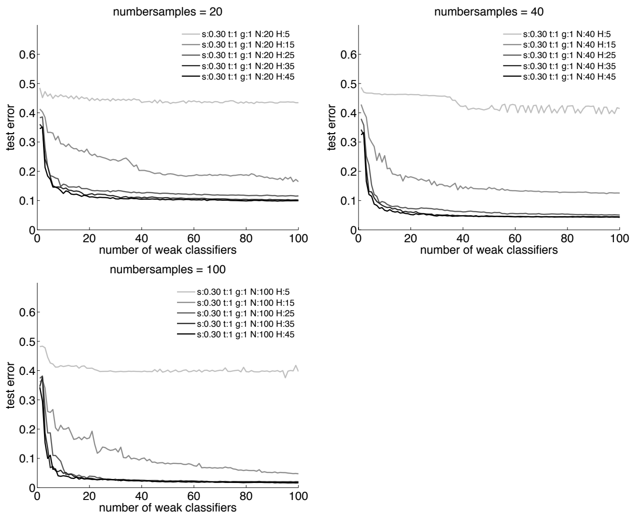

Figure 4. Classification results on synthetic data. The ordinate axis shows the test error of the proposed AdaBoost method with Haar wavelet projection. The plot shows how the performance changes as we increase the number of Haar wavelets of each weak classifier (H).

a very low test error (i.e., 8 weak classifiers for $\sim 2\%$ test error).

In Figures 4 we show the performance of Parzen-AdaBoost when each linear weak classifier is projected onto a Haar wavelet basis and a fixed number of components is retained. Notice that the algorithm achieves 10% test error already with 20 samples, and that on average 25 components are sufficient to capture the variability of the weak classifiers.

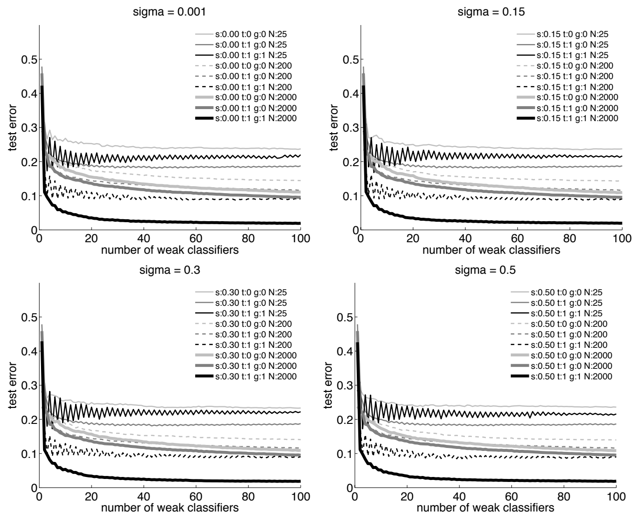

Figure 5. Faces data set. Performance evaluation for varying isotropic smoothing (s), gradient descent (g), tangent vectors (t), and number of samples in the training set (N).

5.2. Real Data Experiments

As real data set we use the Viola-Jones faces data set which is publicly available [1]. In Figure 5 we show the same tests performed in the previous section. Notice that the results resemble very closely the results obtained in the case of synthetic data. The only exception is for the case of 25 samples when both gradient descent and tangent vectors are used. In this case the method does still better than Discrete AdaBoost, but worse than using only tangent vectors. As this does not happen for larger training sets, we conjecture that gradient descent may overfit data on extremely small training sets.

As in the previous section, we test the performance of the proposed algorithm when the weak classifiers are projected onto a Haar wavelet basis. Figure 6 shows the results for 5, 15, 25, 35, 45, and 55 components in the truncated Haar wavelet series. Notice that the performance does not change visibly when more than 25 components are kept. In Figure 7 we show the result of training the method on $8,000$ samples (almost half of the database) and show a final comparison between the main features: optimization of the weak classifiers via gradient descent and usage of tangent vectors. Again the performance shows a consistent behavior. Not only AdaBoost based on tangent vectors and gradient descent achieves the best performance in these experiments by and large, but also it does so more quickly. Having quick convergence to the final test error means that fewer weak classifiers can be used, with considerable reduction of the computations at run-time.

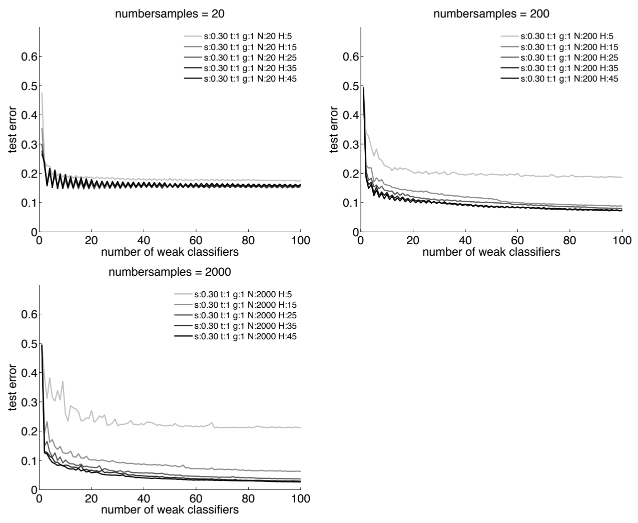

Figure 6. Faces data set. As in the synthetic data set, the ordinate axis shows how the performance changes as we vary the number of Haar wavelet components of each weak classifier (H).

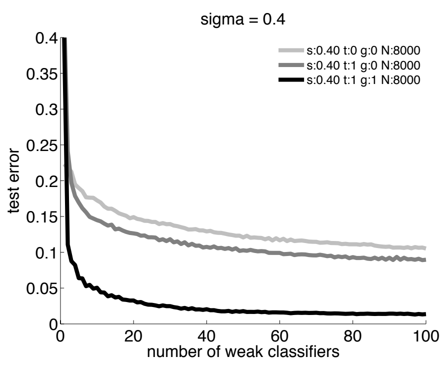

Figure 7. Faces data set. In this experiment we perform training on about half of the data set. The results are once again consistent with the performance in the previous cases, showing that the method scales well with the size of the training set.

6. Acknowledgments

The authors have been supported by AFOSR FA9550-06-1-0138 and ONR 67F-1080868.

7. Conclusions

We have presented a novel boosting method that incorporates several features. We start by extending the traditional AdaBoost framework to Parzen's windows and then show how this can be used to incorporate invariance to geometric and photometric transformations of the data set. Furthermore, our formulation allows the introduction of gradient descent to find optimal weak classifiers and of Haar wavelet projection to improve the computational efficiency at runtime. We demonstrate that our method has strong generalization properties on both synthetic and real data.

References

[1] http://www.cs.ubc.ca/˜pcarbo/viola-traindata.tar.gz. Viola-Jones face database.

[2] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization. In *NIPS*, pages 416–422, 2000.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley and Sons, Inc., 2nd edition, 2001.

[4] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 2001.

[5] J. H. Friedman, T. Hastie, and R. Tibshirani. Special invited paper. additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

[6] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):643–660, 2001.

[7] F. Girosi and N. T. Chan. Prior knowledge and the creation of "virtual" examples for rbf networks. In *Neural Networks for Signal Processing*, pages 201–21, 1995.

[8] T. Hastie, R. Tibishirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[9] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[10] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *Int. J. Comput. Vision*, 14(1):5–24, 1995.

[11] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[12] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision*, pages 555–562, 1998.

[13] A. Pozdnoukhov and S. Bengio. Invariances in kernel methods: From samples to objects. *Pattern Recogn. Lett.*, 27(10):1087–1097, 2006.

[14] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Computational learning theory*, pages 80–91, 1998.

[15] B. Scholkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In *Artificial Neural Networks*, pages 47–52, 1996.

[16] P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop-a formalism for specifying selected invariances in an adaptive network. *Advances in Neural Information Processing Systems 4*, pages 895–903, 1992.

[17] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, page 1470, Washington, DC, USA, 2003. IEEE Computer Society.

[18] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neurosci.*, 3:71–86, 1991.

[19] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, Inc., New York, NY, USA, 1995.

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