Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning

Boris Babenko, Piotr Dollár, Zhuowen Tu, Serge Belongie

Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning*

Boris Babenko[1] Piotr Dollár[1,2] Zhuowen Tu[3] Serge Belongie[1,2]

[1] Comp. Science & Eng. Univ. of CA, San Diego {bbabenko,sjb}@cs.ucsd.edu

[2] Electrical Engineering California Inst. of Tech. pdollar@caltech.edu

[3] Lab of Neuro Imaging Univ. of CA, Los Angeles zhuowen.tu@loni.ucla.edu

Abstract. In object recognition in general and in face detection in particular, data alignment is necessary to achieve good classification results with certain statistical learning approaches such as Viola-Jones. Data can be aligned in one of two ways: (1) by separating the data into coherent groups and training separate classifiers for each; (2) by adjusting training samples so they lie in correspondence. If done manually, both procedures are labor intensive and can significantly add to the cost of labeling. In this paper we present a unified boosting framework for simultaneous learning and alignment. We present a novel boosting algorithm for *Multiple Pose Learning* (MPL), where the goal is to simultaneously split data into groups and train classifiers for each. We also review *Multiple Instance Learning* (MIL), and in particular MIL-BOOST, and describe how to use it to simultaneously train a classifier and bring data into correspondence. We show results on variations of LFW and MNIST, demonstrating the potential of these approaches.

1 Introduction

In object recognition in general and in face detection in particular, data alignment is often necessary to achieve good classification results. *E.g.*, consider the Viola-Jones face detector [1]. The detector works best when trained with images that come from a single coherent group (*e.g.* frontal faces) and lie in approximate correspondence (*e.g.* eyes aligned). Generalizing Viola-Jones to multiple poses and offset training data remains an active area of research.

In this work we are interested in strategies for simultaneous learning and alignment. Data can be aligned in one of two ways: (1) by separating the data into coherent groups and training separate classifiers for each; (2) by adjusting training samples so they lie in correspondence (see Fig. 1). The first strategy is appropriate for out of plane rotation of faces while the second strategy is appropriate for dealing with translational offset. If done manually, both alignment strategies are labor intensive and can significantly add to the cost of labeling.

In this paper we present a unified boosting framework for simultaneous learning and alignment derived using Friedman's gradient boosting methodology [2]. We present a novel boosting algorithm for simultaneously splitting data into groups and training classifiers for each group. We refer to this problem as *Multiple Pose Learning* (MPL) and our algorithm as MPL-BOOST. We also review

* Earlier version appeared as Technical Report CS2008, UCSD (2008)

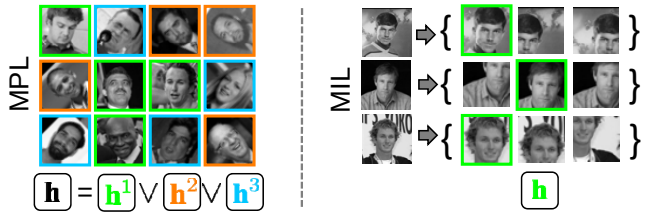

Fig. 1. We present two strategies for simultaneous learning and alignment. Data can be aligned by: (1) separating the data into coherent groups and training separate classifiers for each (MPL); (2) adjusting training samples so they lie in correspondence (MIL).

Multiple Instance Learning (MIL) [3], and describe how to use it to simultaneously train a classifier and bring data into correspondence. In particular, we re-derive and generalize MIL-BOOST, first presented in [4]. Throughout we show the deep connection between these two approaches.

We begin with a review of related work below. In Section 2 we give a detailed and accessible derivation of MPL-BOOST and MIL-BOOST. We show results on variations of the LFW face dataset [5] as well as MNIST digits [6] in Section 3, demonstrating the potential of these approaches. We conclude in Section 4.

1.1 Related Work

A number of papers have focused on grouping data either manually or automatically. *E.g.*, one approach for face detection is to train multiple detectors, each on a manually defined range of orientations [7, 8]. A pre-processing step can be used to predict orientation [7]. However, manual grouping of data can be labor intensive and suboptimal. Alternatively, data can be clustered prior to training [9], but the resulting clustering is likely suboptimal. Most related to this are methods that attempt to group the data during training [10–12], *e.g.* by learning a tree classifier that naturally splits the data [11, 12]. Although promising, these approaches often have heuristic splitting procedures and many parameters.

A number of papers have addressed the second form of alignment. The most direct approach is to try to bring a set of images into joint alignment via congealing [13–15] or thin plate splines [16]. This line of work has yielded promising results; however, alignment is independent of learning. Many weakly supervised recognition methods [17–19] perform alignment implicitly during training, but these alignment strategies are not easily transferable to boosted detection systems. Finally, MIL has been successfully applied to a number of vision problems [20, 21, 4]. MIL offers one way of simultaneously training a discriminative classifier and bringing data into correspondence. As mentioned, we re-derive and generalize MIL-BOOST [4]; details given in Sec. 2.4.

2 Multi-Pose & Multi-Instance Boosting

We begin with an overview of gradient boosting with the log likelihood in Section 2.1, followed by a discussion of smooth approximations of the max function in 2.2. This will serve as the foundation of MPL-BOOST and MIL-BOOST, presented in 2.3 and 2.4, respectively. Finally, we propose various extensions in 2.5.

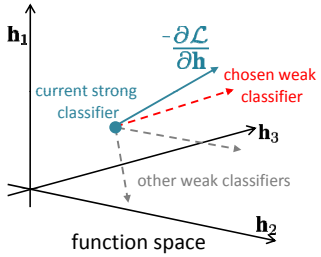

Fig. 2. Illustration of gradient boosting, see text.

2.1 Gradient Boosting Overview

We begin with an overview of the standard setting for supervised learning. Training data consists of n samples $x_i \in \mathcal{X}$ and n corresponding labels $y_i \in \mathcal{Y}$, where typically $\mathcal{X} = \mathbb{R}^d$ and in binary classification $\mathcal{Y} = \{-1, 1\}$. The goal is to learn a classification function $h : \mathcal{X} \to \mathcal{Y}$ that generalizes well to unseen data.

Boosting has been shown to be a very successful approach for binary classification. In boosting, the goal is to train a classifier of the form:

$$\mathbf{h}(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \tag{1}$$

where each $h_t : \mathcal{X} \to \mathcal{Y}$ is a *weak learner* whose performance may only be slightly above chance, and the α_t weigh the weak learners' relative importance. Boosting combines multiple weak learners into a single *strong* classifier with low error. Training proceeds sequentially. In each phase incorrectly classified examples receive more weight; details vary according to boosting algorithm.

Friedman [2] proposes an elegant method for deriving boosting algorithms for a wide range of loss functions. The general idea is to optimize a loss function $\mathcal{L}(\mathbf{h})$ by performing gradient descent on $\mathbf{h}$ (known as gradient descent in function space). Intuitively, the idea is to always select the h_t that most reduces the loss on the training data. Finding the best h_t proceeds in two stages: (1) compute the *optimal* weak classifier response and (2) from the available candidates select the weak classifier that best approximates the optimal response.

More formally, we can consider $\mathbf{h}$ as an n vector whose i^{th} component $\mathbf{h}_i$ has the value $\mathbf{h}(x_i)$. The loss $\mathcal{L}$ is a function over $\mathbf{h}$, and the goal is to minimize $\mathcal{L}(\mathbf{h})$ w.r.t. to $\mathbf{h}$ via gradient descent. In other words to find the optimal weak classifier response in each phase t we compute $-\frac{\partial \mathcal{L}}{\partial \mathbf{h}}$, which is a vector with components $w_i \equiv -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i}$. Ideally we would select h_t such that $w_i = h_t(x_i)\ \forall i$; however, in practice we are limited in the choice of h_t. Therefore, Friedman proposes to find the h_t which is as close as possible to the gradient in function space:

$$h_t = \operatorname*{argmax}_h \sum_{i=1}^{n} w_i h(x_i) \tag{2}$$

The general process is illustrated in Fig. 2. Once we compute h_t, the step size α_t can be found via a line search (again minimizing $\mathcal{L}(\mathbf{h})$). Putting everything together, we obtain the boosting procedure in Fig. 3. Remaining details follow.

Input: Dataset $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$,$y_i \in \{-1, 1\}$
1: **for** $t = 1$ to T **do**
2: Compute weights $w_i = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i}$
3: Train weak classifier h_t using weights $|w_i|$
$h_t = \text{argmin}_h \sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i|$
4: Find α_t via line search to minimize $\mathcal{L}(\mathbf{h})$
$\alpha_t = \text{argmin}_\alpha \mathcal{L}(\mathbf{h} + \alpha h_t)$
5: Update strong classifier $\mathbf{h} \leftarrow \mathbf{h} + \alpha_t h_t$.
6: **end for**

Fig. 3. Boosting via gradient descent in function space.

Log Likelihood: In the remainder of this paper we use the negative log likelihood as the loss function. If we define $p_i \equiv p(y_i = 1|x_i)$, we can write:

$$\mathcal{L}(\mathbf{h}) = -\sum_{i=1}^{n} \Big(\mathbf{1}(y_i = 1) \log p_i + \mathbf{1}(y_i = -1) \log(1 - p_i) \Big). \tag{3}$$

We begin by deriving a standard boosting algorithm using this loss function and gradient descent boosting. This derivation will serve as the foundation for the derivations for MPL-BOOST and MIL-BOOST in Sections 2.3 and 2.4. Note that a related derivation is done in [2]; furthermore, the popular LogitBoost algorithm [22] uses similar criteria, though the actual optimization procedure differs.

First, we define p_i in terms of $\mathbf{h}_i = \mathbf{h}(x_i)$. We follow [2, 22] and define:

$$p_i = \sigma\left(2\mathbf{h}_i\right), \tag{4}$$

where $\sigma(v) = \frac{1}{1+exp(-v)}$ is the sigmoid. Note that $\sigma(v) \in [0, 1]$ and $\frac{\partial \sigma}{\partial v} = \sigma(v)(1 - \sigma(v))$. Finally, taking the derivatives:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \begin{cases} \frac{-1}{p_i} & \text{if } y_i = 1 \\ \frac{1}{1-p_i} & \text{if } y_i = -1 \end{cases} \tag{5}$$

$$\frac{\partial p_i}{\partial \mathbf{h}_i} = 2p_i(1 - p_i), \tag{6}$$

and using the chain rule, we get $w_i = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i} = -\frac{\partial \mathcal{L}}{\partial p_i} \frac{\partial p_i}{\partial h_i}$:

$$w_i = \begin{cases} 2(1 - p_i) & \text{if } y_i = 1 \\ -2p_i & \text{if } y_i = -1 \end{cases} \tag{7}$$

Intuitively, negatives with large p_i and positives with small p_i receive high weight.

Training h_t: In the original gradient boosting paper weak classifiers are chosen by optimizing Eqn. (2), while in most boosting algorithms weak classifiers are chosen by optimizing weighted error:

$$h_t = \arg\min_h \sum_i \mathbf{1}(h(x_i) \neq y_i) w_i, \tag{8}$$

where $y_i \in \{-1, 1\}$. The criterion in Eqn. (8) is more common and many existing learning algorithms approximately minimize this cost function. Here we show that for binary classifiers the first cost function can be converted to the second. Let $y'_i = \text{sign}(w_i)$ (typically $y'_i = y_i$), and $w'_i = |w_i| / \sum_i (|w_i|)$. Then:

$$\begin{aligned} h_{t+1} &= \arg\min_h \sum_i -h(x_i) w_i \\ &= \arg\min_h \sum_i (2 \cdot \mathbf{1}(h(x_i) \neq y'_i) - 1)|w_i| \\ &= \arg\min_h \sum_i 2 \cdot \mathbf{1}(h(x_i) \neq y'_i)|w_i| \\ &= \arg\min_h \sum_i \mathbf{1}(h(x_i) \neq y'_i) w'_i \end{aligned}$$

This transformation allows us to use existing learning algorithms, *e.g.* decision stumps, that minimize weighted error or some approximation to it to train h_t.

Initial distribution: It is often useful to train with an initial (prior) distribution over the data, *e.g.* if more negative than positive training examples are available. Let ω_i be the prior on the i^{th} example. We can modify $\mathcal{L}$ as follows:

$$\mathcal{L}(\mathbf{h}) = -\sum_{i=1}^{n} \omega_i \Big(\mathbf{1}(y_i = 1) \log p_i + \mathbf{1}(y_i = -1) \log(1 - p_i) \Big). \tag{9}$$

Observe that doubling the weight of x_i is the same as having two copies of x_i in the training data. The derivative of $\mathcal{L}$ becomes $\frac{\partial \mathcal{L}}{\partial p_i} = \frac{-\omega_i}{p_i}$ if $y_i = 1$ and $\frac{\omega_i}{1-p_i}$ if $y_i = -1$. Modifying the corresponding equations for w_i is trivial.

Real vs. Discrete h_t: Thus far we have assumed that the weak classifiers are binary, *i.e.* $h_t(x) \in \{-1, 1\}$. If, however, h_t outputs a real valued confidence or score, we can use this score directly. The procedure outlined in Fig. 3 remains unchanged. The alternative is to just use $\text{sign}(h_t(x))$. Using the terminology of [22], we refer to these variations as *real* and *discrete*, respectively.

2.2 Softmax Functions

We now present an overview of differentiable approximations of the *max*, which we refer to as the *softmax*. The general idea is to approximate the max over $\{v_1, \ldots, v_m\}$ by a differentiable function $g_\ell(v_\ell)$, such that:

$$g_\ell(v_\ell) \approx \max_\ell(v_\ell) = v_* \tag{10}$$

$$\frac{\partial g_\ell(v_\ell)}{\partial v_i} \approx \frac{\mathbf{1}(v_i = v_*)}{\sum_\ell \mathbf{1}(v_\ell = v_*)} \tag{11}$$

Intuitively, if v_i is the unique max, then changing v_i changes the max by the same amount, otherwise changing v_i does not affect the max.

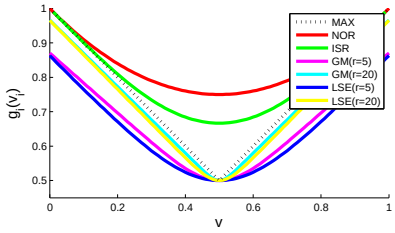

Fig. 4. Various models $g_\ell(v_\ell)$ applied to $v_\ell = (v, 1-v)$.

A number of approximations for g have been proposed. We summarize the choices used here in Table 1: a variant of log-sum-exponential (LSE) [23, 24], generalized mean (GM), noisy-or (NOR) [4], and the ISR model [20, 4]. In Fig. 4 we show the different models applied to $(v, 1-v)$ for $v \in [0, 1]$.

LSE and GM each have a parameter r the controls their sharpness and accuracy; $g_\ell(v_\ell) \to v_*$ as $r \to \infty$ (note that large r can lead to numerical instability). For LSE one can show that $v_* - \log(m)/r \leq g_\ell(v_\ell) \leq v_*$ [23] and for GM that $(1/m)^{1/r} v_* \leq g_\ell(v_\ell) \leq v_*$, where $m = |v_\ell|$. NOR and ISR are only defined over $[0, 1]$. Both have probabilistic interpretations and work well in practice; however, these models are best suited for small m as $g_\ell(v_\ell) \to 1$ as $m \to \infty$. Finally, all models are exact for $m = 1$, and if $\forall \ell \; v_\ell \in [0, 1]$, then $0 \leq g_\ell(v_\ell) \leq 1$ for all models.

	$g_\ell(v_\ell)$	$\partial g_\ell(v_\ell)/\partial v_i$	domain
LSE	$\frac{1}{r} \ln\left(\frac{1}{m} \sum_\ell \exp(r v_\ell)\right)$	$\frac{\exp(r v_i)}{\sum_\ell \exp(r v_\ell)}$	$[-\infty, \infty]$
GM	$\left(\frac{1}{m} \sum_\ell v_\ell^r\right)^{\frac{1}{r}}$	$g_\ell(v_\ell) \frac{v_i^{r-1}}{\sum_\ell v_\ell^r}$	$[0, \infty]$
NOR	$1 - \prod_\ell (1 - v_\ell)$	$\frac{1 - g_\ell(v_\ell)}{1 - v_i}$	$[0, 1]$
ISR	$\frac{\sum_\ell v'_\ell}{1 + \sum_\ell v'_\ell}$, $v'_\ell = \frac{v_\ell}{1 - v_\ell}$	$\left(\frac{1 - g_\ell(v_\ell)}{1 - v_i}\right)^2$	$[0, 1]$

Table 1. Four softmax approximations $g_\ell(v_\ell) \approx \max_\ell(v_\ell)$.

2.3 Multiple Pose Learning

We now present a boosting approach for simultaneously splitting data into groups and training a classifier for each group. Similar to standard supervised learning, we are given n samples $x_i \in \mathcal{X}$ and n corresponding labels $y_i \in \{-1, +1\}$. We assume, however, that there are K latent variables $y_i^k \in \{-1, +1\}$ associated with each sample. Each latent variable defines membership to one of the K groups. A sample is considered positive if it belongs to at least one of these groups, which can be expressed as follows:

$$y_i = \max_k (y_i^k) \tag{12}$$

Our goal is to simultaneously split the positive data into K groups and train K classifiers $\mathbf{h}^1, \ldots, \mathbf{h}^K$, one per group, so that $\max_k(\mathbf{h}^k(x_i)) = y_i$. We call this problem *Multiple Pose Learning*. Note that the latent variables y_i^k are not known; if they were, we could train each $\mathbf{h}^k$ using standard learning approaches.

As before, we would like to optimize the negative log likelihood $\mathcal{L}$. However, we need to modify our definition of the probability of x_i. We define the probability $p_i^k \equiv p^k(y_i = 1|x_i)$ according to a single classifier $\mathbf{h}^k$ as $p_i^k = \sigma(2\mathbf{h}_i^k)$, where $\mathbf{h}_i^k = \mathbf{h}^k(x_i)$. This is similar to Eqn. (4). Given all of the classifiers, we define the probability $p_i \equiv p(y_i = 1|x_i)$ as the *maximum* over the probabilities p_i^k. Using one of the approximations g of the max from Section 2.2, we write:

$$p_i = g_k(p_i^k) = g_k(\sigma(2\mathbf{h}_i^k)) \tag{13}$$

This can be viewed as the probabilistic approximation of Eqn. (12). Note that p_i and consequently $\mathcal{L}$ depend on each $\mathbf{h}_i^k$; to make this dependence explicit we write $\mathcal{L}(\mathbf{h}^1, \cdots, \mathbf{h}^K)$. We optimize $\mathcal{L}(\mathbf{h}^1, \cdots, \mathbf{h}^K)$ by coordinate descent, cycling through k, where in each phase we add a weak classifier to $\mathbf{h}^k$ while keeping all other weak classifiers fixed. The algorithm is summarized in Figure 5.

Input: Dataset $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$,$y_i \in \{-1, 1\}$, K

for $t = 1$ to T **do**
  for $k = 1$ to K **do**
    Compute weights $w_i^k = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^k}$
    Train weak classifier h_t^k using weights $|w_i^k|$
      $h_t^k = \text{argmin}_h \sum_i \mathbf{1}(h(x_i) \neq y_i)|w_i^k|$
    Find α_t via line search to minimize $\mathcal{L}(\cdot, \mathbf{h}^k, \cdot)$
      $\alpha_t = \text{argmin}_\alpha \mathcal{L}(\cdot, \mathbf{h}^k + \alpha h_t^k, \cdot)$
    Update strong classifier $\mathbf{h}^k \leftarrow \mathbf{h}^k + \alpha_t^k h_t^k$.
  end for
end for

Fig. 5. MPL-BOOST

All that remains is to derive $w_i^k = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^k}$. Using the chain rule we get:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^k} = \frac{\partial \mathcal{L}}{\partial p_i} \frac{\partial p_i}{\partial p_i^k} \frac{\partial p_i^k}{\partial \mathbf{h}_i^k}. \tag{14}$$

All other terms cancel. Plugging in $\frac{\partial \mathcal{L}}{\partial p_i}$ and $\frac{\partial p_i^k}{\partial \mathbf{h}_i^k}$, derived in Eqns. (5-6), we get:

$$w_i^k = \begin{cases} \frac{2p_i^k(1-p_i^k)}{p_i} \frac{\partial p_i}{\partial p_i^k} & \text{if } y_i = 1 \\ \frac{-2p_i^k(1-p_i^k)}{1-p_i} \frac{\partial p_i}{\partial p_i^k} & \text{if } y_i = -1 \end{cases} \tag{15}$$

The form of $\frac{\partial p_i}{\partial p_i^k}$ depends on the choice of softmax model g, see Table 1. We list the equations for w_i^k for different choices of the softmax in Table 2.

	$y_i = -1$	$y_i = 1$
LSE	$\frac{-2p_i^k(1-p_i^k)}{1-p_i}\frac{\exp(rp_i^k)}{\sum_\ell \exp(rp_i^\ell)}$	$\frac{2p_i^k(1-p_i^k)}{p_i}\frac{\exp(rp_i^k)}{\sum_\ell \exp(rp_i^\ell)}$
GM	$\frac{-2p_i}{1-p_i}\left(\frac{(p_i^k)^r-(p_i^k)^{r+1}}{\sum_\ell (p_i^\ell)^r}\right)$	$2\left(\frac{(p_i^k)^r-(p_i^k)^{r+1}}{\sum_\ell (p_i^\ell)^r}\right)$
NOR	$-2p_i^k$	$\frac{2p_i^k(1-p_i)}{p_i}$
ISR	$\frac{-2\chi_i^k p_i}{\sum_\ell \chi_i^\ell}, \chi_i^k = \frac{p_i^k}{1-p_i^k}$	$\frac{2\chi_i^k(1-p_i)}{\sum_\ell \chi_i^\ell}, \chi_i^k = \frac{p_i^k}{1-p_i^k}$

Table 2. MPL equations for w_i^k for different choices of g.

2.4 Multiple Instance Learning

We now turn to Multiple Instance Learning (MIL), and re-derive and generalize MIL-BOOST [4]. MIL was introduced in [3] for scenarios where a training sample cannot be represented as a single feature vector, but rather as a set of m vectors, often called a 'bag', $X_i = \{x_{i1}, \ldots, x_{im}\}$. It is assumed that every instance has some true label $y_{ij} \in \{-1, +1\}$, but its value is not known during training (it is latent, much like y_i^k in MPL). A bag is labeled positive if at least one of its instances is positive, which can be expressed as follows:

$$y_i = \max_j(y_{ij}) \tag{16}$$

Here our goal is to learn a classifier $\mathbf{h}$ using only bag labels y_i, such that $\max_j(\mathbf{h}(x_{ij})) = y_i$. Similar to MPL, if the latent variables y_{ij} were known this classifier could be trained with standard supervised learning.

As in the previous cases, we would like to optimize the negative log likelihood $\mathcal{L}$. To do so we must define $p_i \equiv p(y_i = 1|X_i)$, the probability of a bag X_i. We begin by defining the probability $p_{ij} \equiv p(y_{ij} = 1|x_{ij})$ of an instance x_{ij} in the same manner as before: $p_{ij} = \sigma(2\mathbf{h}_{ij})$, where $\mathbf{h}_{ij} = \mathbf{h}(x_{ij})$. Given the instance probabilities, we define the bag probability p_i as the *maximum* over the instance probabilities p_{ij}. Using the softmax g in place of the max we write:

$$p_i = g_j(p_{ij}) = g_j(\sigma(2\mathbf{h}_{ij})) \tag{17}$$

This can be viewed as the probabilistic approximation of Eqn. (16). Note the similarity between Eqns. (12) and (16), and also between Eqns. (13) and (17). Although the form of the input data and learned classifiers for MIL and MPL are different, the form of the weights w_{ij} is nearly identical to the form of the weights w_i^k. In fact, if we replace the superscript k with the subscript j in Eqns. (14) and (15), we get the equivalent MIL equations (not shown).

Algorithm details are given in Figure 6. The optimization procedure for MIL is similar to the optimization procedure for regular boosting described in Section 2.1; compare Figures 3 and 6.

Input: Dataset $\{X_1, \ldots, X_n\}$, $\{y_1, \ldots, y_n\}$,$y_i \in \{-1, 1\}$

1: **for** $t = 1$ to T **do**
2: Compute weights $w_{ij} = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{ij}}$
3: Train weak classifier h_t using weights $|w_{ij}|$
$h_t = \text{argmin}_h \sum_{ij} \mathbf{1}(h(x_{ij}) \neq y_i)|w_{ij}|$
4: Find α_t via line search to minimize $\mathcal{L}(\mathbf{h})$
$\alpha_t = \text{argmin}_\alpha \mathcal{L}(\mathbf{h} + \alpha h_t)$
5: Update strong classifier $\mathbf{h} \leftarrow \mathbf{h} + \alpha_t h_t$.
6: **end for**

Fig. 6. MIL-BOOST

2.5 Extensions

For completeness, we conclude by discussing a number of noteworthy extensions and variations to MPL and MIL, which we leave to future work.

Combining MPL and MIL Thus far we have considered the two solutions to the alignment problem independently. It is conceivable that in certain situation it would be beneficial to apply both alignment approaches simultaneously. Here we briefly describe such an approach.

We are given a training dataset consisting of bags $X_i = \{x_{i1}, \ldots, x_{im}\}$ and bag labels $y_i \in \{-1, +1\}$. We assume that there exist K latent variables y_{ij}^k for each sample x_{ij} which define group memberships as in MPL. A bag is labeled positive if at least one of its instances belongs to at least one of the K groups:

$$y_i = \max_k \max_j (y_{ij}^k) = \max_{jk} (y_{ij}^k) \tag{18}$$

The formulation of this seemingly more complex problem is actually similar to both MIL and MPL. We can define the probability of x_{ij} according to $\mathbf{h}^k$ in the same manner as before: $p_{ij}^k = \sigma(2\mathbf{h}_{ij}^k)$, where $\mathbf{h}_{ij}^k = \mathbf{h}^k(x_{ij})$. Then, we can define the bag probability p_i as the *maximum* over the instances *and* the classifiers:

$$p_i = g_{jk}(p_{ij}^k) = g_j(\sigma(2\mathbf{h}_{ij}^k)) \tag{19}$$

Note the similarity of Eqn. (19) to Eqns. (13) and (17). The form of the weights $w_{ij}^k = -\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{ij}^k}$ is again very similar to both MIL and MPL. Although we omit further details, derivation of the resulting boosting algorithm is straightforward.

Alternate MPL/MIL Formulation When deriving MPL, we defined $p^k(y_i = 1|x_i)$ as $p_i^k = \sigma(2\mathbf{h}_i^k)$, and the overall probability as $p_i = g_k(p_i^k)$. Instead of defining p_i in terms of the p_i^k, we could define $\mathbf{h}_i = g_k(\mathbf{h}_i^k)$ and p_i in terms of $\mathbf{h}_i$:

$$p_i = \sigma(2\mathbf{h}_i) = \sigma(2g_k(\mathbf{h}_i^k)) \tag{20}$$

Compare with Eqn. (13). Since $\mathbf{h}_i^k \in \mathbb{R}$, only the LSE softmax model is appropriate (see Table 1). Taking derivatives as before, we obtain:

$$w_i^k = \frac{\partial \mathcal{L}}{\partial p_i} \frac{\partial p_i}{\partial \mathbf{h}_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_i^k} = w_i \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_i^k} = w_i \frac{\exp(r\mathbf{h}_i^k)}{\sum_\ell \exp(r\mathbf{h}_i^\ell)}, \tag{21}$$

where w_i are the weights given in Eqn. (7). These weights can be used directly in our MPL boosting algorithm, see Fig. 5. Note that we could apply the same exact re-derivation for MIL, we omit details for space.

Negatives in MPL/MIL In MIL we defined $p_i = p(y_i = 1|X_i)$ in terms of p_{ij}, allowing us to write an expression for $\mathcal{L}$. We couldn't write $\mathcal{L}$ directly in terms of p_{ij} as we did not know the instance labels y_{ij}. However, for negatives, we know that $y_{ij} = -1$ for every j. We can thus write an alternate expression for $\mathcal{L}$:

$$\mathcal{L}(\mathbf{h}) = -\sum_{i=1}^{n} \Big(\mathbf{1}(y_i = 1) \log p_i + \mathbf{1}(y_i = -1) \sum_{j=1}^{m} \log(1 - p_{ij}) \Big). \tag{22}$$

For positives, w_{ij} remains unchanged. For negatives, $w_{ij} = -2p_{ij}$, as in Eqn. 7. All negative instances are treated identically regardless of the bag X_i they came from, just as in the NOR model. In MPL, we can use the same variant of $\mathcal{L}$, obtaining $w_i^k = -2p_i^k$ for negatives. This seems appropriate, as each classifier $\mathbf{h}^k$ has to correctly classify every negative.

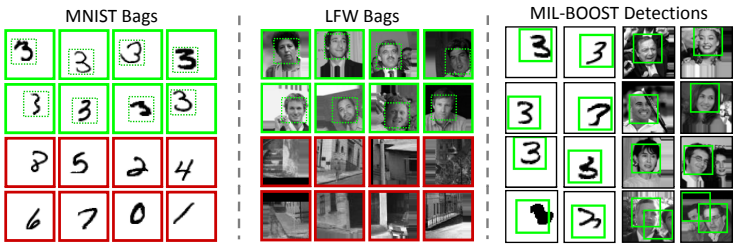

Fig. 7. Examples of positive (green) and negative (red) bags for the MIL experiments: *Left:* MNIST, *Middle:* LFW. For positive bags we highlight the true (unknown) object location. *Right*: Actual detections of a MIL-BOOST classifier (after non-maximal suppr.).

3 Experiments

We used two datasets: LFW [5] and MNIST [6]. MNIST is a simple optical digit recognition dataset; we use it to illustrate certain properties of the algorithms presented. The LFW dataset contains images of faces in the media. Although primarily intended for evaluating face recognition, we use it within a detection setting. For 'hard' non-face images we used false-positives returned by a Viola-Jones face detector. Fig. 7 shows example images from each datasets. In all experiments we use random Haar features [25] as the weak classifiers.

3.1 Multiple Instance Learning

As shown in [4], MIL-BOOST is well suited for discovering translational alignment. In fact, Viola *et al.* argue that delineating object boundaries is an inherently ambiguous task. Instead, we place several overlapping bounding boxes in a bag and use MIL to simultaneously train a classifier and select a good alignment.

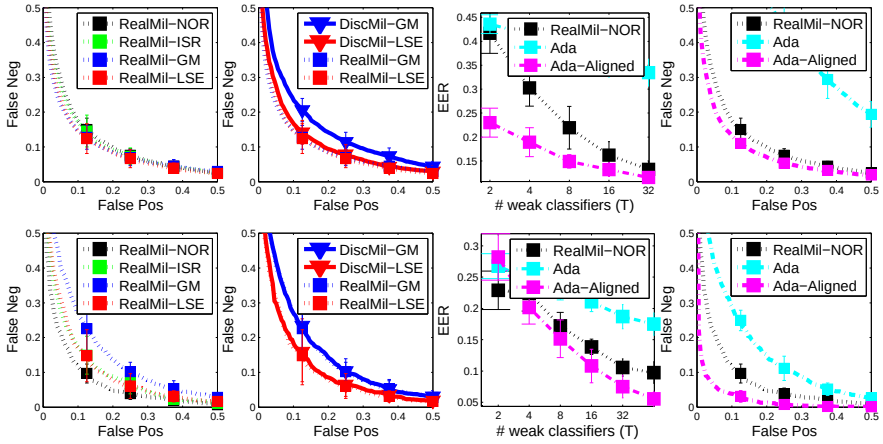

Fig. 8. MIL Results. *Top*: MNIST, *Bottom*: LFW. *(a)* Comparison of the different softmax functions. *(b)* Comparing the discrete and real versions of MIL-BOOST. *(c)* Equal error rate (EER) versus the number of weak classifiers. *(d)* MIL-BOOST compared to standard learning (AdaBoost) on aligned and non-aligned data.

LFW The LFW faces are aligned and centered. For the purpose of this experiment we 'corrupted' this alignment by randomly shifting the face 1-10 pixels within each 63×63 image. The same process was applied to negatives to avoid artifacts. Example images are shown in Figure 7. In each trial we used 400 images for training and 400 for testing; results are averages over 10 trial. For all algorithms tested we used boosting to select $T = 64$ weak stump classifiers from a pool of 1000 random Haar features.

Our results can be divided into 3 groups. First, we compare variations of MIL-BOOST. We show results obtained using different softmax function in Fig. 8(a) and using the real and discrete variations in 8(b). Second, we compare MIL-BOOST with standard supervised learning. We trained two additional classifiers using AdaBoost [22]: one using the unaligned data we used for MIL and one using cropped and aligned data. Results are shown in Fig. 8(c-d).

Finally, we seek to measure how well MIL-BOOST aligns the data. Specifically, we measure the *consistency* of the alignment w.r.t. to the ground truth (a constant offset in the predictions can easily be corrected). We define the alignment error ϵ as the standard deviation of the true and predicted locations l_i^*

and l_i: $\epsilon = \mathbf{std}(l_i^* - l_i)$. We compute l_i by smoothing the classifier response and recording the maximum location (but only for i s.t. $p_i > 25\%$).

For MIL-BOOST (the Real-LSE version) the alignment error averaged over 10 trials was $\epsilon = 5.07$. For comparison, we evaluated how well AdaBoost trained with (the small amount of) aligned data can perform alignment. The alignment error was $\epsilon = 5.97$, which was actually higher. The classifier response was rather diffuse, most likely due to the small amount of training data (note that AdaBoost with aligned data still had lower classification error). Nevertheless, this experiment suggests that MIL-BOOST does indeed recover a good alignment.

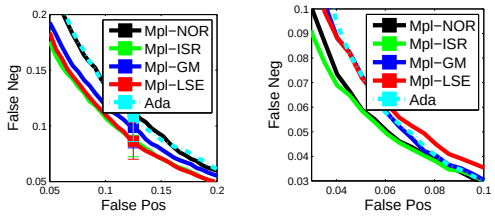

Fig. 9. MPL Results. ROC plots comparing four variants of MPL-BOOST and standard learning (AdaBoost) with: *left:* MNIST; *right:* LFW.

MNIST We arbitrarily chose the digit '3' to be the positive and the rest negative. We pad each 28×28 image with an 8 pixel border and randomly translated the digit within the resulting 44×44 image. All results are averaged over 10 trials and use 1000 images, 10% for training each time. We performed identical experiments as for the LFW data. Results are shown in Figure 8, top; the average alignment errors were $\epsilon = 3.08$ for AdaBoost and $\epsilon = 1.40$ for MIL-BOOST.

3.2 Multiple Pose Learning

The goal of MPL is to simultaneously group data and train a classifier for each group. Therefore a natural application of MPL is to train detectors for object categories that have several distinct views. *E.g.*, out of plane rotations make it difficult to place faces into correspondence, which can make it difficult to train a precise face detector. On the other hand, if we can accurately group data during training, we not only achieve higher accuracy but also potentially more meaningful output (*e.g.* a face detector that also predicts facial pose).

LFW We took 2000 LFW faces and synthesized three distinct groups: (1) faces kept as is, (2) faces rescaled by a factor of 2.5, and (2) faces rescaled by 2 and rotated 90°. In each case we use boosting to select $T = 64$ weak classifiers from 1000 candidates. We randomly split 4000 images into equal sized training and testing sets, and average all experiments over 10 trials. Figure 9(right) shows ROC curves comparing AdaBoost with four versions of MPL-BOOST. In Fig. 10 we

show randomly selected test images grouped according to the learned classifiers. We see that MPL-BOOST recovers the groups reasonably well. Since we have the ground truth memberships y_i^k, we can quantify the quality of the grouping (up to a permutation of the group ids). For the faces, the grouping accuracy for the four softmax models were: NOR 0.81%, ISR 0.84%, GM 0.83%, LSE 0.88%.

MNIST Our final experiment is similar to the one described above. We chose the digits 0-2 as positives and the others as negatives. The setup varied only slightly (1500 positives, 1500 negatives, 40% used for training, averaged over 10 trials). ROC curves are shown in Fig. 9 (left). We computed membership accuracy by assuming each positive digit forms its own group: NOR 0.83%, ISR 0.89%, GM 0.87%, LSE 0.87%.

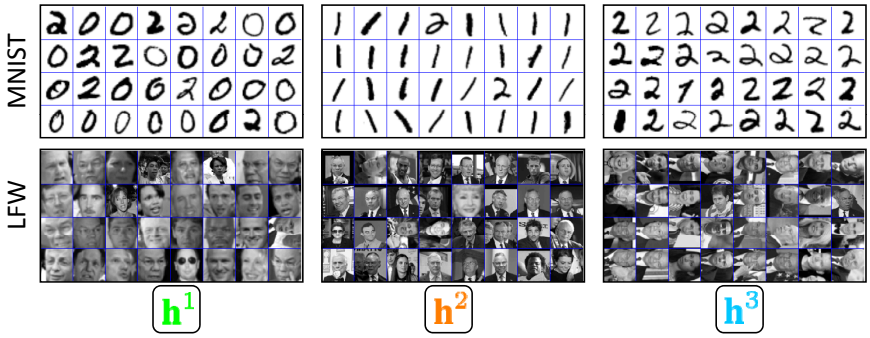

Fig. 10. MPL Alignment Results. Randomly selected test images grouped according to the 3 classifiers trained with MPL. *Top:* MNIST; *Bottom:* LFW.

4 Conclusion and Future Work

In this paper we discussed two simultaneous learning and alignment strategies that can potentially lead to higher accuracy detection systems. Specifically, we derived a novel framework we call Multiple Pose Learning and a unified boosting approach that encompasses both MPL and MIL. The results we presented are preliminary but promising. Significant testing remains to evaluate if these approaches can improve state of the art detection systems. We also described a number of extensions to MPL and MIL that warrant further investigation.

Acknowledgments

BB and PD were funded by NSF IGERT Grant DGE-0333451. SB was funded by NSF Career Grant #0448615 and the Alfred P. Sloan Research Fellowship. ZT was funded by NIH Grant U54RR021813 entitled Center for Computational Biology.

References

1. Viola, P., Jones, M.: Fast multi-view face detection. In: CVPR. (2001)
2. Friedman, J.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29**(5) (2001) 1189–1232
3. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple-instance problem with axis parallel rectangles. Artificial Intelligence (1997)
4. Viola, P., Platt, J.C., Zhang, C.: Multiple instance boosting for object detection. In: NIPS. (2005)
5. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller., E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusets, Amherst (2007)
6. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Gradient-based learning applied to document recognition. In: IEEE. (1998)
7. Jones, M., Viola, P.: Fast multi-view face detection. Technical Report 2003-96, MERL (2003)
8. Huang, C., Ai, H., Li, Y., Lao, S.: Vector boosting for rotation invariant multi-view face detection. In: ICCV. (2005)
9. Seemann, E., Leibe, B., Schiele, B.: Multi-Aspect Detection of Articulated Objects. In: CVPR. (2006)
10. Torralba, A., Murphy, K., Freeman, W.: Sharing visual features for multiclass and multiview object detection. PAMI **29**(5) (2007) 854–869
11. Tu, Z.: Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In: ICCV. (2005)
12. Wu, B., Nevatia, R.: Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In: ICCV. (2007)
13. Learned-Miller, E.: Data Driven Image Models through Continuous Joint Alignment. PAMI (2006) 236–250
14. Huang, G., Jain, V., Learned-Miller, E.: Unsupervised joint alignment of complex images. In: ICCV. (2007)
15. Vedaldi, A., Soatto, S.: A Complexity-Distortion Approach to Joint Pattern Alignment. In: NIPS. Volume 19. (2007)
16. Mori, G., Belongie, S., Malik, J.: Efficient Shape Matching Using Shape Contexts. PAMI (2005) 1832–1837
17. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: ECCV. (2000)
18. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: CVPR. (2003)
19. Crandall, D.J., Huttenlocher, D.P.: Weakly supervised learning of part-based spatial models for visual object recognition. In: ECCV. (2006)
20. Keeler, J.D., Rumelhart, D.E., Leow, W.K.: Integrated segmentation and recognition of hand-printed numerals. In: NIPS. (1990)
21. Nowlan, S., Platt, J.: A convolutional neural net. hand tracker. In: NIPS. (1995)
22. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Technical report, Stanford University (1998)
23. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge Univ. Press (2004)
24. Ramon, J., De Raedt, L.: Multi instance neural networks. ICML, Workshop on Attribute-Value and Relational Learning (2000)
25. Dollár, P., Tu, Z., Tao, H., Belongie, S.: Feature mining for image classification. In: CVPR. (2007)