TRAINING BOOSTING-LIKE ALGORITHMS WITH SEMI-SUPERVISED SUBSPACE LEARNING

Jingsong Xu[⋆†] *Qiang Wu*[†] *Jian Zhang*[†] *Fumin Shen*[⋆] *Zhenmin Tang*[⋆]

[⋆] School of Computer Science and Technology, Nanjing University of Science and Technology, China
[†]Faculty of Engineering and Information Technology, University of Technology, Sydney

ABSTRACT

Boosting algorithms have attracted great attention since the first real-time face detector by Viola & Jones through feature selection and strong classifier learning simultaneously. On the other hand, researchers have proposed to decouple such two procedures to improve the performance of Boosting algorithms. Motivated by this, we propose a boosting-like algorithm framework by embedding semi-supervised subspace learning methods. It selects weak classifiers based on class-separability. Combination weights of selected weak classifiers can be obtained by subspace learning. Three typical algorithms are proposed under this framework and evaluated on public data sets. As shown by our experimental results, the proposed methods obtain superior performances over their supervised counterparts and AdaBoost.

Index Terms— AdaBoost, Boosting, Semi-supervised Discriminant Analysis, Semi-supervised Subspace Learning

1. INTRODUCTION AND BACKGROUND

As one of the most widespread of ensemble based learning algorithms, boosting has achieved great successes. The brief idea of boosting is to progressively combine weak classifiers to form a strong classifier. During the process, boosting pays more attention to those misclassified training samples [1].

Conventional boosting algorithm, e.g. AdaBoost, learns weak classifiers and combination coefficients simultaneously. Wu *et al.* [2] presented that these two procedures can be decoupled. They showed two alternatives to improve the performance and speed of AdaBoost. Motivated by this, Paisitkriangkrai *et al.* [3] proposed Greedy Sparse Linear Discriminant Analysis (GSLDA) which selects weak classifiers based on class-separability (i.e. LDA) other than training error used in AdaBoost. During weak classifier selection, coefficients for weak classifiers combination are learned simultaneously through eigen-decomposition in LDA. In order to keep the advantage of AdaBoost on re-weighting, they applied a similar re-weighting scheme into GSLDA and proposed Boosted GSLDA (BGSLDA). Better performances than AdaBoost were obtained by BGSLDA. Note that we use 'feature' and 'weak classifier' interchangeably when decision stump is adopted as weak classifier [4].

LDA is a popular supervised subspace learning algorithm. Extensive experiments show that LDA works fine with sufficient training samples [5]. This has also been verified by GSLDA which achieved excellent performances on face detection with thousands of training samples. However, the performance of LDA decreases greatly when it deals with small sample size problem [6]. We find that GSLDA also fails to train a reasonable classifier with less training samples. In this paper, unlabeled data is added to tackle the problems above. Actually, many semi-supervised boosting algorithms have been proposed to deal with small sample size problem for boosting, such as SSMBoost [7].

Inspired by the success of semi-supervised learning [6, 8], we propose a boosting-like algorithm framework by embedding semi-supervised subspace learning. In this framework, weak classifiers are selected based on class-separability other than training error in AdaBoost and SSMBoost. That is the selected weak classifiers can maximize between-class difference and minimize within-class differences. In particular, we show that unlabeled data plays two roles in this framework to support semi-supervised subspace learning and weak classifier candidates generation. Three algorithms are proposed under this framework in which the new algorithm achieves better than the previous ones. Experimental results show that they achieve better performances than their supervised counterparts (GSLDA/BGSLDA) and AdaBoost.

2. ALGORITHMS

In this section, we first give a brief review of Semi-supervised Discriminant Analysis (SDA). Then we use SDA as an alternative criterion to train boosting-like classifiers using both labeled and unlabeled data. In this paper, our focus is binary classification problems, while our methods can be extended to multi-class scenario.

2.1. Overview of SDA

Given M labeled training samples $\mathbf{X}_{\mathcal{L}} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M\}$ and N unlabeled samples $\mathbf{X}_{\mathcal{U}} = \{\mathbf{x}_{M+1}, \mathbf{x}_{M+2}, \cdots, \mathbf{x}_{M+N}\}$,

y_i is the label, $y_i \in \{+1,-1\}, 1 \le i \le M$. Let $\mathbf{X} = \mathbf{X}_\mathcal{L} \bigcup \mathbf{X}_\mathcal{U}$ denote the set of all training samples. $\mathbf{X} \in \mathbb{R}^{K\times(M+N)}$, K is the dimension of feature vectors. The objective function of Linear Discriminant Analysis (LDA) is to find the projection vector $\mathbf{w}$ which can maximize between-class difference and minimize within-class differences [5]:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\mathrm{T}\mathbf{S}_b\mathbf{w}}{\mathbf{w}^\mathrm{T}\mathbf{S}_w\mathbf{w}} \tag{1}$$

where $\mathbf{S}_b$ is between-class scatter matrix and $\mathbf{S}_w$ is within-class scatter matrix. The total scatter matrix is $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w$. The objective function of LDA (1) can be cast as a generalized eigenvalue decomposition problem [9]: $\mathbf{S}_b\mathbf{w} = \lambda\mathbf{S}_t\mathbf{w}$. The solutions are projection vector $\mathbf{w}$ and eigenvalue λ.

From the view of manifold learning [10], above relationship can be represented with matrixes. We can define matrix $\mathbf{W}$ as the weight of the edge $(\mathbf{x}_i, \mathbf{x}_j)$:

$$W_{i,j} = \begin{cases} 1/M_{y_i}, & \text{if } y_i = y_j \\ 0, & \text{if } y_i \neq y_j, \end{cases} \tag{2}$$

where M_{y_i} denotes the number of labeled samples in class y_i. Based on $\mathbf{W}$, we can obtain the following Laplacian matrixes: $\mathbf{L}^{sw} = \mathbf{I} - \mathbf{W}, \mathbf{L}^{sb} = \mathbf{W} - \frac{1}{M}\mathbf{e}\mathbf{e}^\mathrm{T}$ and $\mathbf{L}^{st} = \mathbf{I} - \frac{1}{M}\mathbf{e}\mathbf{e}^\mathrm{T}$ and the corresponding $\mathbf{S}_w = \mathbf{X}_\mathcal{L}\mathbf{L}^{sw}\mathbf{X}_\mathcal{L}^\mathrm{T}, \mathbf{S}_b = \mathbf{X}_\mathcal{L}\mathbf{L}^{sb}\mathbf{X}_\mathcal{L}^\mathrm{T}$ and $\mathbf{S}_t = \mathbf{X}_\mathcal{L}\mathbf{L}^{st}\mathbf{X}_\mathcal{L}^\mathrm{T}$, where $\mathbf{e} = (1,1,\cdots,1)^\mathrm{T}$ is a M-dimensional vector.

The performance of LDA tends to be degraded when there are no sufficient training samples. In order to improve the performance, Cai *et. al* [9] proposed Semi-supervised Discriminant Analysis (SDA) to prevent overfitting of LDA with less labeled data. SDA applies p-nearest of each sample to model the relationships of all training samples including labelled and unlabelled training samples, forming a graph. The weight of the edge in the graph encodes this relationship, defined by matrix $\mathbf{S}$:

$$S_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in N_p(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_p(\mathbf{x}_i) \\ 0, & otherwise, \end{cases} \tag{3}$$

where $N_p(\mathbf{x}_i)$ denotes the set of p nearest neighbors of $\mathbf{x}_i$. SDA defines a regularizer $\mathbf{J}(\mathbf{w})$ as:

$$\mathbf{J}(\mathbf{w}) = \sum_{ij}(\mathbf{w}^\mathrm{T}\mathbf{x}_i - \mathbf{w}^\mathrm{T}\mathbf{x}_j)^2 S_{ij} = \mathbf{w}^\mathrm{T}\mathbf{X}\mathbf{L}\mathbf{X}^\mathrm{T}\mathbf{w} \tag{4}$$

$\mathbf{L} = \mathbf{D} - \mathbf{S}$ is the Laplacian matrix [11]. $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j S_{ij}$. The underlying explanation is that if two samples are close, they are likely to be in the same class. The objective function of SDA is:

$$\max_{\mathbf{w}} \frac{\mathbf{w}^\mathrm{T}\mathbf{S}_b\mathbf{w}}{\mathbf{w}^\mathrm{T}\mathbf{S}_t\mathbf{w} + \alpha\mathbf{J}(\mathbf{w})} = \max_{\mathbf{w}} \frac{\mathbf{w}^\mathrm{T}(\mathbf{X}_\mathcal{L}\mathbf{L}^{sb}\mathbf{X}_\mathcal{L}^\mathrm{T})\mathbf{w}}{\mathbf{w}^\mathrm{T}(\mathbf{X}_\mathcal{L}\mathbf{L}^{st}\mathbf{X}_\mathcal{L}^\mathrm{T} + \alpha\mathbf{X}\mathbf{L}\mathbf{X}^\mathrm{T})\mathbf{w}} \tag{5}$$

Parameter α controls the trade off between model complexity and empirical loss. It is clear that (5) is similar to (1) except $\mathbf{S}_w$ is replaced. Thus it can be solved in the same way as (1) [9]. With this regularizer, the output of SDA, $\mathbf{w}$ not only considers the discriminant power among labeled data but the intrinsic geometrical structure among all training samples.

Algorithm 1: Training Boosting-like Algorithm with Semi-supervised Discriminant Analysis (BSDA)

```
  Input: Training samples: $\mathbf{X}$;
  The number of weak classifiers: $T_{max}$;
  Parameters: $\alpha$ and $p$ for SDA.
1 Initialization: $\mathbf{H} = \emptyset$.
2 for $K$ features do
3     1. Find a weak classifier $h_k$ (decision stump) with the
      least error on labeled training samples.
4     2. Calculate the predictions of $h_k$ on all training samples
      and add the predictions into $\mathbf{H}$.
5 end
6 for $t = 1$ to $T_{max}$ do
7     Select the best weak classifier $h_t^*$ that yields the largest
      objective value in (6), and keep the projection vector $\mathbf{w}_t$.
8 end
  Output: The strong classifier $f(x) = \sum_{t=1}^{T_{max}} h_t^*(x)w_t$.
```

2.2. SDA for Boosting-like Algorithm

When training samples are insufficient, distribution of the whole data cannot be modeled correctly. The selected weak classifiers by GSLDA [3] are not discriminative and more likely overfitted. To solve this problem, we choose SDA as criterion instead of LDA used in GSLDA which selects weak classifiers by taking advantage of unlabeled data.

The framework of our proposed method Boosting-like algorithm with SDA (BSDA) is presented in Algorithm 1.

The first stage (lines 2 - 5) generates weak classifier candidates according to labeled samples only. These weak classifier candidates are to be selected and then combined to form a strong classifier. We adopt decision stump [4] to train weak classifier candidates. In the work, we train K decision stumps based on the original labeled training set $\mathbf{X}_\mathcal{L}$. The weak classifier candidates will be generated based on the minimal error from labelled training samples. We denote that all weak classifier candidates compose a weak classifier family $\{h_k(\cdot) : x \to \{1,-1\}, k = 1,...,K\}$. Then, all the classification results for the training samples can be obtained and stored in matrix $\mathbf{H} \in \mathbb{R}^{K\times(M+N)}$ with $H_{k,i} = h_k(x_i)$. H_k is the k-th row of $\mathbf{H}$ which stores the classification results of weak classifier h_k on all training samples.

$$\{k, \mathbf{w}\} = \arg\max_{k,\mathbf{w}} \frac{\mathbf{w}^\mathrm{T}(\mathbf{A}_\mathcal{L}^t\mathbf{L}^{sb}(\mathbf{A}_\mathcal{L}^t)^\mathrm{T})\mathbf{w}}{\mathbf{w}^\mathrm{T}(\mathbf{A}_\mathcal{L}^t\mathbf{L}^{st}(\mathbf{A}_\mathcal{L}^t)^\mathrm{T} + \alpha\mathbf{A}^t\mathbf{L}(\mathbf{A}^t)^\mathrm{T})\mathbf{w}}$$
$$\text{s.t. } \mathbf{A}^t = \begin{bmatrix}\mathbf{A}^{t-1} \\ H_k\end{bmatrix}, \ 1 \le k \le K, \ \mathbf{A}^0 = \emptyset \tag{6}$$

The second stage (lines 6 - 8) is used for feature selection. In each iteration t, every feature h_k is picked up from the available feature candidates and spanned in addition to the previously selected feature(s). The training sample is described by this spanned feature set, i.e. $\mathbf{A}^t$ in (6). Compared with (5), we update the description $\mathbf{X}$ during the iterations in (6). Matrix $\mathbf{A}^{t-1}$ stores the predictions of selected weak classifiers before iteration t. Matrix $\mathbf{A}^t \in \mathbb{R}^{t\times(M+N)}$ is made up of two parts: the predictions for labeled samples $\mathbf{A}^t_{\mathcal{L}} \in \mathbb{R}^{t\times M}$ and unlabeled samples $\mathbf{A}^t_{\mathcal{U}} \in \mathbb{R}^{t\times N}$, i.e. $\mathbf{A}^t = [\mathbf{A}^t_{\mathcal{L}}|\mathbf{A}^t_{\mathcal{U}}]$. The best weak classifier h^*_t which yields the largest objective value in (6) will be selected. Repeat the process until there are T_{max} selected features.

Stage 3, the output of the algorithm, is to build strong classifier learning. When selecting best h^*_t in stage 2, its corresponding projection vector $\mathbf{w}_{T_{max}}$ is obtained as the combination weights. The final strong classifier is $f(x) = \sum_{t=1}^{T_{max}} h^*_t(x)w_t$, w_t denotes the t-th element in $\mathbf{w}_{T_{max}}$.

Algorithm 2: Training Steps for WBSDA

```
   Input: Training samples: $\mathbf{X}$;
   The number of weak classifiers: $T_{max}$;
   Parameters: $\alpha$ and $p$ for SDA.
1  Initialization: $\mathbf{H} = \emptyset$.
2  for $t = 1$ to $T_{max}$ do
3      $\mathbf{H} = \emptyset$.
4      for $K$ features do
5          1. Find a weak classifier $h_k$ (decision stump) with
           the least error on labeled training samples.
6          2. Calculate the predictions of $h_k$ on all training
           samples and add the predictions into $\mathbf{H}$.
7      end
8      Select the best weak classifier $h^*_t$ that yields the largest
       objective value in (6), and keep the projection vector $\mathbf{w}_t$.
9      Update the weights for labeled training samples [4].
10 end
   Output: The strong classifier $f(x) = \sum_{t=1}^{T_{max}} h^*_t(x)w_t$.
```

2.3. SDA for Weighted Boosting-like Algorithm

From Algorithm 1, weak classifier candidates remain unchanged during the T_{max} iterations. It should be better to obtain more informative weak classifier candidates. In BSDA, decision stumps are generated without considering an important factor of AdaBoost: re-weighting scheme. [12] showed that performances of boosting algorithms degraded significantly if weighting scheme was omitted. During AdaBoost training, each training sample is assigned a weight. AdaBoost finds the best weak classifier with the least weighted error based on training samples [1]. Then the weights of the training samples are updated in which the misclassified samples are given larger weights. Here we apply the re-weighting scheme to improve the performance of BSDA.

See Algorithm 2 for this Weighted Boosting-like algorithm with SDA (WBSDA). After selecting the best weak classifier at current iteration t in line 8, the samples weights are updated in line 9. A new feature pool (feature candidates) will be re-generated in lines 4 - 7 before the next iteration. That is to say, the weak classifier family is updated in each iteration, which changes $\mathbf{H}$ accordingly. This is a key difference compared with BSDA in which $\mathbf{H}$ remains unchanged.

2.4. SDA for Semi-supervised Weighted Boosting-like Algorithm (SemiWBSDA)

In section 2.2 and 2.3, unlabeled data serves as prior or underlying knowledge to guide subspace learning. The weights of unlabeled samples are not updated. We will show that unlabeled data can help generate better weak classifier candidates through semi-supervised boosting.

For semi-supervised boosting, the objective function in this paper adds a cost function for unlabeled data. Then this problem can be cast as conventional boosting problem [13, 14, 15, 7]. We adopt Semi-supervised MarginBoost (SSMBoost) [7] to generate weak classifier candidates. However, in each round of feature selection, the weights for all training sample including labeled and unlabeled data are updated [7] (instead of updating labeled data only as in WBSDA). That is, it changes the training steps in lines 5 and 9 in Algorithm 2 by considering unlabeled samples. This new feature pool contains more informative weak classifier candidates. In this new algorithm, unlabeled data is used for two purposes: one is used for generating weak classifier candidates (i.e. 'Semi-') and the other is for subspace learning (i.e. '-SDA').

3. EXPERIMENTS

3.1. Experimental Setup

In this section, we evaluate seven algorithms, namely AdaBoost [1], SSMBoost [7], GSLDA [3], BGSLDA [3] and our proposed BSDA, WBSDA and SemiWBSDA on a handwritten digit recognition set: MNIST and a computer vision set: face detection. The maximum number of iterations T_{max} is limited to 200. The parameters α for SDA in (6) is validated from $\{2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}\}$. The number of nearest neighbors p in SDA is set to 5. The best performance with the highest accuracy on validation set is selected for all algorithms for comparison. We randomly partition the training examples into disjoint sets of labeled and unlabeled samples. The size of the unlabeled partition is set to be r =50%, 75%, 85% and 95% of the number of all training examples. The experiments are conducted for 10 times, and average test error rates with standard deviations are reported.

MNIST: MNIST handwritten digits data set contains 10 categories of digits with 1000 for each digit. Each digit is represented by a 28×28 image. It has split the total data into a training set and a test set. Sub-sets of digits '2' vs. '3' and '5'

Table 1: Classification results (test error rates (%) ± standard deviations) for digit '2' vs. '3' on the MNIST data set.

Ratio	50%	75%	85%	95%
AdaBoost	8.4 ± 1.8	9.7 ± 2.3	9.7 ± 3.0	17.9 ± 5.3
SSMBoost	8.7 ± 1.9	9.9 ± 2.3	9.2 ± 1.5	33.2 ± 21.6
GSLDA	17.2 ± 3.7	18.4 ± 6.5	16.3 ± 5.8	20.4 ± 6.3
BGSLDA	16.8 ± 3.0	19.3 ± 5.7	16.8 ± 6.0	17.6 ± 4.6
BSDA	6.9 ± 1.8	7.9 ± 2.1	8.0 ± 2.2	9.3 ± 2.1
WBSDA	6.6 ± 1.1	7.7 ± 2.2	7.6 ± 1.9	9.6 ± 2.1
SemiWBSDA	$\mathbf{6.6 \pm 0.7}$	$\mathbf{7.3 \pm 2.1}$	$\mathbf{7.1 \pm 1.1}$	$\mathbf{9.3 \pm 1.5}$

Table 2: Classification results (test error rates (%) ± standard deviations) for digit '5' vs. '8' on the MNIST data set.

Ratio	50%	75%	85%	95%
AdaBoost	$\mathbf{8.7 \pm 0.8}$	11.4 ± 1.6	13.5 ± 4.3	20.0 ± 3.5
SSMBoost	9.2 ± 1.0	11.2 ± 1.4	12.8 ± 2.3	18.2 ± 2.9
GSLDA	12.9 ± 2.1	19.1 ± 3.3	21.2 ± 4.0	29.7 ± 10.0
BGSLDA	11.4 ± 2.6	18.3 ± 4.1	20.3 ± 2.4	30.4 ± 10.6
BSDA	10.3 ± 1.8	10.1 ± 1.8	10.3 ± 2.8	$\mathbf{13.4 \pm 2.5}$
WBSDA	10.0 ± 1.9	10.0 ± 1.8	9.6 ± 3.1	13.9 ± 4.5
SemiWBSDA	9.2 ± 1.3	$\mathbf{9.9 \pm 1.9}$	$\mathbf{9.5 \pm 1.7}$	13.6 ± 2.1

vs. '8' are used in our work for evaluating the performance of proposed methods. 220 examples are randomly selected as training examples (110 per class), and 200 examples are randomly selected as validation ones. The original test set is used for testing. The gray values of each image are used as samples feature. The classification errors and standard deviations are reported in Table 1 and 2.

Face: We randomly select a set of 1000 24×24 samples with 500 face images and 500 non-face images from [2]. 20% of the samples are used for training; 20% are for cross validation and the rest for testing. We use Haar-like feature [4] for its popularity in face detection. 16,233 features are uniformly sampled from the whole Haar-like feature space [2]. The classification results are shown in Table 3.

3.2. Results and Discussions

From the results above, it is shown that three proposed algorithms outperform other methods in most experiments.

As mentioned before, GSLDA and BGSLDA will suffer without enough effective training samples. Therefore, they perform worse in our case due to the lack of training samples.

Our experiments demonstrate the importance of unlabeled data in subspace learning. By taking the advantages of unlabeled data, the proposed methods have lower error rates than GSLDA and BGSLDA. In addition, we set a fixed ratio between the number of labeled and unlabeled samples, and gradually increase the number of two kinds of samples. The experiment is conducted on the digit '2' vs. '3' MNIST data set

Table 3: Classification results (test error rates (%) ± standard deviations) on face data set.

Ratio	50%	75%	85%	95%
AdaBoost	6.5 ± 1.4	8.4 ± 1.2	10.0 ± 4.0	26.0 ± 6.4
SSMBoost	7.9 ± 1.3	10.4 ± 1.7	9.7 ± 2.0	23.4 ± 6.5
GSLDA	12.6 ± 2.2	16.4 ± 2.0	19.5 ± 3.3	27.5 ± 5.7
BGSLDA	10.1 ± 1.5	13.4 ± 3.9	22.7 ± 6.6	28.3 ± 7.1
BSDA	6.6 ± 1.1	6.8 ± 1.6	9.4 ± 3.0	$\mathbf{12.8 \pm 3.4}$
WBSDA	6.3 ± 0.7	6.7 ± 1.2	9.2 ± 3.2	12.9 ± 4.2
SemiWBSDA	$\mathbf{5.9 \pm 0.9}$	$\mathbf{6.0 \pm 0.8}$	$\mathbf{8.9 \pm 1.2}$	12.9 ± 3.3

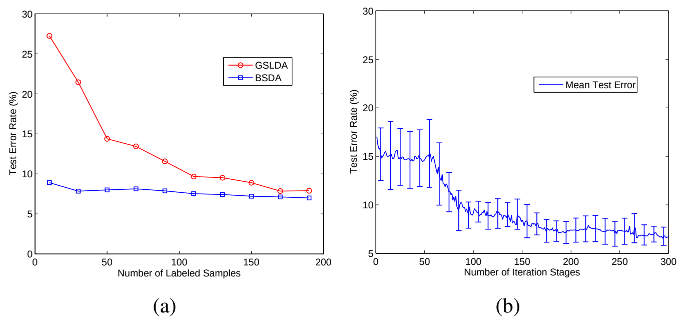

Fig. 1: (a) The mean test error rates for GSLDA and BSDA with different size of training samples. (b) The maximum number iteration is set to be 300.

with a ratio of 1:2. The classification results of GSLDA and BSDA are shown in Fig. 1 (a). It is shown that 15% of the total labeled samples are enough for BSDA to achieve a stable performance (in terms of test error rate in our case). The performances of BSDA and GSLDA are similar when increasing the number of labeled training samples to 200.

It also shows that (see Tables 1, 2 and 3) SemiWBSDA performs best among three proposed methods because of additional regularization introduced in SemiWBSDA for tuning both weak classifiers creation and feature selection.

Finally, we evaluate the sensibility of our proposed methods to overfitting. We run SemiWBSDA with $T_{max} = 300$ and 85% of unlabeled data on '2' vs. '3' MNIST data set. If overfitting happens, the error rate would increase after certain iteration. Fig. 1 (b) shows that no overfitting is observed and the test error can still decrease after 200 iterations.

4. CONCLUSION

In this paper, we propose a boosting-like algorithm framework by embedding semi-supervised subspace learning to deal with small labeled sample size problem. Three algorithms are proposed under this framework which show the two key roles of unlabeled data. Experiments on public data sets are verified. It is concluded that the proposed approaches outperform their supervised counterparts (GSLDA/BGSLDA) and AdaBoost under different situations.

5. REFERENCES

[1] J Friedman, T Hastie, and R Tibshirani, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[2] J Wu, S C Brubaker, M D Mullin, and J M Rehg, "Fast asymmetric learning for cascade face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 369–382, 2008.

[3] S Paisitkriangkrai, C Shen, and J Zhang, "Efficiently training a better visual detector with sparse eigenvectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1129–1136.

[4] P Viola and M J Jones, "Robust real-time face detection," *Int. J. Comp. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[5] Keinosuke Fukunaga, *Introduction to statistical pattern recognition*, Academic Pr, 1990.

[6] O. Chapelle, B. Schölkopf, A. Zien, et al., *Semi-supervised learning*, vol. 2, 2006.

[7] F. dAlché Buc, Y. Grandvalet, and C. Ambroise, "Semi-supervised marginboost," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, pp. 553–560, 2002.

[8] Ming Li, Hang Li, and Zhi-Hua Zhou, "Semi-supervised document retrieval," *Inform Process Manag*, vol. 45, no. 3, pp. 341–355, 2009.

[9] D Cai, X He, and J Han, "Semi-supervised discriminant analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–7.

[10] X. He, S. Yan, Y. Hu, P. Niyogi, and H.J. Zhang, "Face recognition using laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328–340, 2005.

[11] F.R.K. Chung, *Spectral graph theory*, Number 92. Amer Mathematical Society, 1997.

[12] M J Saberian, H Masnadi-Shirazi, and N Vasconcelos, "TaylorBoost: First and second-order boosting algorithms with explicit margin control," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 2929–2934.

[13] Ke Chen and Shihai Wang, "Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 129–143, Jan. 2011.

[14] K.P. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 289–296, 2002.

[15] Pavan Kumar Mallapragada, Student Member, and Rong Jin, "SemiBoost : Boosting for semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, 2009.