Boosted-PCA for Binary Classification Problems

Seaung Lok, Ham and Nojun Kwak
School of Electrical and Computer Engineering
Ajou University
San 5, Woncheon-Dong, Yeungtong-Gu,
Suwon, 443-749 Korea
[hihiham|nojunk]@ajou.ac.kr

Abstract— **In this paper, a Boosted-PCA algorithm is proposed for efficient classification of two class data. Conventionally, in classification problems, the roles of feature extraction and classification have been distinct, i.e., a feature extraction method and a classifier are applied sequentially to classify input variable into several categories. In this paper, these two steps are combined into one resulting in a good classification performance. More specifically, each principal component is treated as a weak classifier in Adaboost algorithm to constitute a strong classifier for binary classification problems. The proposed algorithm is applied to UCI data set and showed better recognition rates than sequential application of feature extraction and classification methods such as PCA+1NN and PCA+SVM.**

I. INTRODUCTION

In pattern recognition and machine learning society, classification has been a classical problem and the most widely used classifiers include support vector machines (SVM) [1], k-nearest neighbors (k-NN) [2] and neural networks [3]. Nowadays, combinations of multiple classifiers are gaining attention [4], among which methods based on boosting techniques are very popular [5].

Adaptive boosting a.k.a. Adaboost has been successfully applied to binary classification problems [6]. It is a kind of boosting algorithm capable of constructing a strong classifier through a weighted combination of weak classifiers [4] [5] [6]. From many weak classifiers, the one with smallest weighted error on the training data is selected. Once a weak classifier is selected, each training sample gets a new weight value and the procedure of selecting a weak classifier is repeated T times. Finally, the selected weak classifiers are combined to make a strong classifier. In doing so, Adaboost increases the weight of erroneous samples while decreasing the weight of correctly classified samples at each stage and selects the weak classifier with the lowest weighted error. The Viola-Jones algorithm is a successful example of applying Adaboost in object detection problems where Haar-like features are used as weak classifiers [7].

In this paper, to construct a weak classifier of the Adaboost algorithm, principal component analysis (PCA) is applied. PCA [8] [9] is one of the most popular linear feature extraction methods used mainly for dimensionality reduction. It is a powerful tool for reducing the number of observed variables into a smaller number of artificial variables that account for most of the variance in the data set. In PCA, the eigenvectors corresponding to the largest eigenvalues of sample covariance matrix is searched for, which corresponds to the direction of principal components of sample data. The principal components may then be used as predictor or criterion variables in subsequent analyses. For example, in classification problems, these principal components may be used as inputs to a classifier. Like this scenario, conventionally, the roles of feature extraction and classification have been distinct, i.e., a feature extraction method and a classifier are applied sequentially to classify input variable into several categories. Others propose a PCA features representation for later stages [12], however there used PCA features in the late stages of boosting. In this paper, by using PCA in constructing weak classifiers of Adaboost, the two steps of feature extraction and classification are combined into one resulting in a better classification performance. The proposed method is named as Boosted-PCA.

The paper is organized as follows. Adaboost algorithm and PCA are briefly reviewed in section II and section III respectively. In section IV, Boosted-PCA is proposed. The experimental results on UCI dataset are shown in section V. Finally, conclusions and future works follow in Section VI.

II. ADABOOST ALGORITHM

The Adaboost algorithm introduced in 1995 by Freund and Schapire [5] uses a series of weak classifiers to constitute a strong classifier. Consider the training samples $\{(x_i, y_i)\}_{i=1\ldots N}$ where x is a data vector, y is a binary class label and N is the number of training samples. We assume $y_i \in \{0,1\}$. In addition, consider a set of candidate weak classifier or weak learner which is denoted as $\{g_j\}_{j=1\ldots k}$ where k is the total number candidate weak classifiers. Among the candidate weak classifiers, a set of weak classifiers $\{h_t\}_{t=1\ldots T}$ is selected based on the weighted error on the training samples. Here, T is the number of weak classifiers or round of iterations in Adaboost.

0. Input: training samples

$$(x_1, y_1), \dots, (x_N, y_N), x_i \in R^d, y_i \in \{0,1\}.$$

1. Weight initialization: for each sample i=1…N, initialize each weight as follows:

$$w_{1,i} = \begin{cases} \frac{1}{2m} \text{ if } y_i = 0 \\ \frac{1}{2n} \text{ if } y_i = 1 \end{cases}.$$

Here m and n are numbers of samples belonging to class 0 and class 1 respectively.

2. Repeat for $t = 1, \dots, T$

1) Computation of error: for each candidate classifier $\{g_j\}_{j=1\dots k}$, compute the weighted error as follows:

$$\varepsilon = \sum_{i=1}^{N} w_{t,i} |g_j(x_i) - y_i|.$$

2) Selection of weak classifier: choose the candidate classifier g_j with the lowest error ε_t and set it as h_t, i.e.,

$$h_t \leftarrow g_j$$

3) Weight update: update the weights as follows:

$$w_{t+1,i} = w_{t,i} \times e^{\alpha_t |h_t(x_i) - y_i|},$$

where $\alpha_t = \log(\frac{1-\varepsilon_t}{\varepsilon_t})$.

4) Weight normalization: Normalize each weight as follows:

$$w_{t+1,i} \leftarrow \frac{w_{t+1,i}}{\sum_{i=1}^{u} w_{t+1,i}}$$

4. Output: the final strong classifier is:

$$H(x) = \begin{cases} 1 \text{ if } \sum_{t=1}^{T} \alpha_t h_t(x) > 0.5 \\ 0 \text{ otherwise} \end{cases}.$$

Figure 1. Adaboost algorithm

The final strong classifier is denoted as H. Given an input vector x, the role of any cl assifier (candidate weak, weak, and strong) is to classify x into one of the classes (0 or 1).

The Adaboost algorithm is outlined in Fig. 1. Given training samples, in step 1, each training sample is allocated with a corresponding weight value. Here, the sum of weights belonging to each class is set to be equal to one half.

Step 2 is the main step of Adaboost. In this step, firstly, for each candidate weak classifier, the weighted error is computed and the one with the smallest weighted error is selected as the t^{th} weak classifier h_t. Determination of each weak classifier involves evaluation of each candidate on all the training samples in other to find the best performing candidate. The best performing candidate is chosen based on the weighted error it produces. This weighted error is a function of the weights belonging to the training examples.

Once the weak classifier is selected, weight of each sample is updated in substeps 3) and 4). In this step, weights of correctly classified samples are decreased while those of erroneous samples are increased. This makes the next weak classifier to have a good chance of correctly classifying the incorrectly classified samples by the current weak classifier. In the weight update step, the parameter α_t plays an important role. Note that $\alpha_t \geq 0$ if $\varepsilon_t \leq 0.5$ (which we can assume without loss of generality), and α_t increases if ε_t decreases. For large α_t, the weights of the incorrectly classified samples increase more rapidly than the case of small α_t.

After selecting T consecutive weak classifiers, finally, in Step 4, the strong classifier is achieved. Note that the output of strong classifier H is the weighted sum of those of the weak classifiers. In this step also, α_t is important. Intuitively, α_t can be interpreted as measuring the importance that is assigned to h_t. The weak classifier with the smallest weighted error contributes the most to the combined strong classifier.

III. PCA : Principal Component Analysis

In data analysis problems with a large number of input variables, dimensionality reduction methods are typically used to reduce the number of input variables to simplify the problems without degrading performances. Among them, the principal component analysis [9] is one of the most popular methods. PCA seeks to find the vectors that best describe the data in terms of reproducibility. Also it is proven that the vectors with best reproducibility correspond to the ones that maximize the variance of given data. These projections constitute a low-dimensional linear subspace by which the data structure in the original input space can effectively be captured.

In Fig.2, the PCA algorithm is briefly overviewed. Consider n samples in a d-dimensional space. The objective of PCA is to find a d'-dimensional subspace that best describes the data. Here, it is assumed that $d' < d$. In the first step of PCA, the sample data are centered by subtracting the mean vector μ from each sample vector. Then, the scatter matrix S is constructed as shown in the figure. Finally, eigenvalues and eigenvectors of S are obtained by taking eigenvalue decomposition of S. Once eigenvectors are found from the scatter matrix, the next step is to order them by eigenvalue, highest to lowest. This gives the components in order of

0. Input : training samples $x_1, \dots, x_N \in R^d$.
Objective: find a d' dimensional subspace where the reconstruction error of x_i in this subspace is minimized.

1. Construction of scatter matrix:

$$S = \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T, \ \mu = \frac{1}{N} \sum_{i=1}^{N} x_i.$$

2. Eigenvalue decomposion:

$$Sv_i = \lambda_i v_i, \ i = 1 \dots d', \ \lambda_1 > \lambda_2 > \dots > \lambda_{d'}.$$

$$V = [v_1, v_2, \dots v_{d'}] \in R^{d \times d'}$$

3. Output: a projection matrix V.

4. Projection: When a new data $x \in R^d$ is presented,

$$y = V^T(x - \mu) \in R^{d'}$$

becomes the principal component of x.

Figure 2. PCA algorithm

significance. Now, you can decide to ignore the components of lesser significance by selecting only the d' important components. Finally, the projection matrix $V = [v_1, v_2, \ldots v_{d'}]$ is the result of PCA where v_i is the eigenvector of S corresponding to the i^{th} largest eigenvalue. When a new data $x \in R^d$ is presented, $y = V^T(x - \mu)$ becomes the principal component of x.

IV. PROPOSED ALGORITHM: BOOSTED-PCA

In this section, a candidate weak classifier based on PCA is used in the Adaboost algorithm. To adapt PCA in Adaboost algorithm, the conventional PCA should be modified to incorporate the concept of sample weight.

Consider the i^{th} training sample has a weight $w_{t,i}$ in the t^{th} stage of Adaboost. Then the first step of modified PCA is to calculate the weighted sample mean as follows:

$$\mu_t = \sum_{i=1}^{N} w_{t,i} x_i. \quad (1)$$

Then the weighted scatter matrix is obtained based on this weighted mean as follows:

$$S_t = \sum_{i=1}^{N} w_{t,i}(x_i - \mu_t)(x_i - \mu_t)^T. \quad (2)$$

Once the weighted scatter matrix S_t is obtained, by eigenvalue decomposition, we can obtain k eigenvectors $\{v_j\}_{j=1\ldots k}$, where k is the rank of S.

For each principal component v_j, one can easily construct a candidate weak classifier that classifies a vector x into one of two classes as follows:

$$g_j(x) = \begin{cases} 0 \text{ if } v_j^T x \le p_j \tau_j \\ 1 \text{ if } v_j^T x > p_j \tau_j \end{cases}. \quad (3)$$

Here, $p_j \in \{1, -1\}$ and $\tau_j \in R$ are the polarity and threshold of j^{th} candidate classifier respectively which are set to be the optimal values that minimizes the weighted training error of the classifier, i.e.,

$$(p_j, \tau_j) = \operatorname{argmin}_{(p_j,\tau_j)} \sum_{i=1}^{N} w_{t,i} |g_j(x_i) - y_i|. \quad (4)$$

After the construction of candidate classifiers $\{g_j\}_{j=1\ldots k}$, the t^{th} weak classifier is selected and then the remaining weight update procedures are identical to those of conventional Adaboost in Fig. 1.

The Boosted-PCA algorithm is summarized in Fig. 3. At each repeat of weak classifier selection, different PCAs are performed to minimize corresponding weighted reconstruction error. The projection vector v_j, threshold τ_j, weighted mean μ_t and polarity p_j of each stage is saved to make a strong classifier H. Once, a new test data is presented, it is subtracted by the mean vector and then projected using the projection vector. Finally, the strong classifier uses the saved thresholds and polarities as well as α_t to classify the projected test data into one of the two classes.

0. Input: training samples
$$(x_1, y_1), \ldots, (x_N, y_N), x_i \in R^d, y_i \in \{0,1\}.$$

1. Weight initialization: for each sample i=1…N, initialize each weight as follows:
$$w_{1,i} = \begin{cases} \frac{1}{2m} \text{ if } y_i = 0 \\ \frac{1}{2n} \text{ if } y_i = 1 \end{cases}.$$
Here m and n are numbers of samples belonging to class 0 and class 1 respectively.

2. Repeat for $t = 1, \ldots, T$
 1) Weighted PCA: Let
 $$S_t = \sum_{i=1}^{N} w_{t,i}(x_i - \mu)(x_i - \mu)^T, \qquad \mu_t = \sum_{i=1}^{N} w_{t,i} x_i$$
 Then, apply eigenvalue decomposition to S to obtain a set of eigenvectors $\{v_j\}_{j=1\ldots k}$.
 2) Construction of candidate weak classifiers
 $$g_j(x) = \begin{cases} 0 \text{ if } v_j^T x \le p_j \tau_j \\ 1 \text{ if } v_j^T x > p_j \tau_j \end{cases}.$$
 3) Computation of error: for each candidate classifier $\{g_j\}_{j=1\ldots k}$, compute the weighted error as follows:
 $$\varepsilon = \sum_{i=1}^{N} w_{t,i} |g_j(x_i) - y_i|.$$
 4) Selection of weak classifier: choose the candidate classifier g_j with the lowest error ε_t and set it as h_t, i.e.,
 $$h_t \leftarrow g_j$$
 5) Weight update: update the weights as follows:
 $$w_{t+1,i} = w_{t,i} \times e^{\alpha_t |h_t(x_i) - y_i|},$$
 where $\alpha_t = \log(\frac{1-\varepsilon_t}{\varepsilon_t})$.
 6) Weight normalization: Normalize each weight as follows:
 $$w_{t+1,i} \leftarrow \frac{w_{t+1,i}}{\sum_{i=1}^{u} w_{t+1,i}}$$

4. Output: the final strong classifier is:
$$H(x) = \begin{cases} 1 \text{ if } \sum_{t=1}^{T} \alpha_t h_t(x) > 0.5 \\ 0 \text{ otherwise} \end{cases}.$$

Figure 3. Boosted-PCA algorithm

V. EXPERIMENTAL RESULTS

In this section, the performance of the Boosted-PCA is compared with those of conventional sequential application of PCA and a classifier such as PCA+kNN and PCA+SVM. Matlab was used to implement all the algorithms. For SVM, linear SVM in MATLAB was used.

The UCI machine learning repository which contains many real-world data sets that have been used by numerous researchers [10] [11] is used in this experiment. The binary classification problems in UCI data set were selected for performance evaluation, which include Sonar, Pima, Heart disease, Liver, Breast cancer and Australian.

Table 1 shows the brief information of the data sets used in this paper. The Adaboost was trained for 30 iterations. For all the experiment, 10-fold cross validation was used, i.e., 90% of the data set was used as training samples while the remaining 10% was used as test samples and then this was repeated 10 times. The recognition rates of Table 2 are the average recognition rate of 10 experiments on the test data. In the table, for all the dataset, Boosted-PCA outperforms the other method such as PCA+1NN and PCA+SVM. Especially, for Breast Cancer data, around 20% improvement of recognition rates is seen by the proposed method.

TABLE I. SUMMARY OF UCI DATA SETS

	# of variables	**# of classes**	**# of instances**
Sonar	60	2	208
Pima	8	2	768
Heart disease	13	2	297
Liver	6	2	345
Breast cancer	9	2	683
Australian	14	2	690

TABLE II. RECOGNITION RATES (%) OF EACH ALGORITHM

	PCA+1NN	PCA+SVM	**Boosted-PCA**
Sonar	59.52	52.38	**72.38**
Pima	64.93	64.93	**69.22**
Heart disease	53.33	73.66	**74.33**
Liver	58.23	58.82	**61.76**
Breast cancer	76.17	77.64	**97.35**
Australian	52.60	66.08	**69.56**

VI. CONCLUSIONS

In this paper, we proposed a new method 'Boosted-PCA' for binary classification problems. In this method, PCA is incorporated in the structure of Adaboost. More specifically, at each stage of Adaboost, different PCAs are performed and the resultant principal components were used as candidate weak classifiers to constitute a strong classifier. The performance of the proposed algorithm on UCI dataset was better than the conventional sequential application of PCA and a classifier such as PCA+1NN and PCA+SVM.

As a future work, firstly, the algorithm can be extended to multi class problems. This can be done by substituting the Adaboost with multi-class Adaboost. Secondly, not only PCA but also other feature extraction methods such as LDA and ICA can be incorporated in the structure of Adaboost to achieve better classification performances.

ACKNOWLEDGEMENT

This work was supported by Korea Research Foundation Grant funded by Korean Government (KRF-2011-0005324).

REFERENCES

[1] Christopher J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery, vol.2, pp.121-167, 1998.*

[2] Cover T. M. and Hart P. E, "Nearest Neighbor Pattern Classification", *IEEE Transactions on Information Theory, vol. IT-13(1), pp.21-27, 1967.*

[3] Simon Haykin, "Neural networks", *2nd Edition, Prentice Hall, 1999.*

[4] Robert. E. Schapire and Yoram Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, *37(3):297-336, 1999.*

[5] Yoav Freund , Robert E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting," *In European Conference on Computational Learning Theory, pp. 23-37, 1995.*

[6] Y. Freund, R. E. Schapire, "A Short Introduction to Boosting", *Journal of Japanese Society for Artificial Intelligence, 14(5), pp.771-780, 1999.*

[7] P. viola and M. J. Jones, "Robust Real-time Face Detection", *International Journal of Computer Vision, Vol.57, No.2, pp. 137-154, 2004.*

[8] Matthew Turk and Alex Pentland, "Eigenface for Recognition," *Journal of Cognitive Neuroscience 3(1), pp.70-86, 1991.*

[9] I.T.Joliffe, "Principal Component Analysis, " *Springer-Verlag*, 1986.

[10] UCI Data Sets, Availabe: http://archive.ics.uci.edu/ml/datasets.html.

[11] P. M. Murphy and D. W. Aha, "UCI repository of machine learning databases," 1994, For more information contact ml-repository@ics.uci.edu or http://www.cs.toronto.edu/_delve/.

[12] Dong Zhang, S. Z. Li, D. Gatica-Perez, "Real-time face detection using boosting in hierarchical feature spaces", *ICPR 2004. Proceedings of the 17th International Conference on, Vol. 2, pp. 411-414, 2004.*