Kullback-Leibler Boosting

Ce Liu   Hueng-Yeung Shum

Microsoft Research Asia

{i-celiu, hshum}@microsoft.com

Abstract

In this paper, we develop a general classification framework called Kullback-Leibler Boosting, or KLBoosting. KLBoosting has following properties. First, classification is based on the sum of histogram divergences along corresponding global and discriminating linear features. Second, these linear features, called KL features, are iteratively learnt by maximizing the projected Kullback-Leibler divergence in a boosting manner. Third, the coefficients to combine the histogram divergences are learnt by minimizing the recognition error once a new feature is added to the classifier. This contrasts conventional AdaBoost where the coefficients are empirically set. Because of these properties, KLBoosting classifier generalizes very well. Moreover, to apply KLBoosting to high-dimensional image space, we propose a data-driven Kullback-Leibler Analysis (KLA) approach to find KL features for image objects (e.g., face patches). Promising experimental results on face detection demonstrate the effectiveness of KLBoosting.

1. Introduction

Robust and reliable classifiers are widely used in pattern recognition problems. A good classifier should generalize well the properties learnt from a small amount of data, with a reasonable speed for both training and classification. Many theories and algorithms have been developed for learning a general classifier. For example, Neural Networks [1] was devised two decades ago to learn a classification function from training data and applied to solving many visual recognition/detection problems [8, 14]. However, it is nontrivial to decide the number of the neurons and the type of the nonlinear function, and to train the parameters. Support Vector Machines (SVM) [15] are then developed to maximize the margin of the labeled data and explain the insight of Neural Network. Although there are many successful applications [6, 7], choosing proper kernel functions for a specific real problem remains challenging. Moreover, the number of the support vectors that compose the decision function would dramatically increase when the decision manifold becomes complicated.

Recently, Boosting (e.g. AdaBoost) [11] theory has been proposed to combine simple weak learners to a strong classifier. Boosting is simple and easy to implement. It has been proven that Boosting minimizes an exponential function of the margin over the training set [3]. However, a strong classifier learnt by AdaBoost is suboptimal for the applications in terms of the error rate [2]. Similar to SVM, typically a large number of weak learners should be integrated, e.g., over 6000 are used in [16] for a face detector. Two problems remain unsolved in AdaBoost: how to design the weak learners, and how to optimally combine them.

Our objective in this paper is to design an optimal classifier using a small number of robust features. We project high-dimensional data to linear features and get 1D histograms along these features. These histograms are robust statistics, and have been successfully used in computer vision and machine learning [20, 4, 12]. We use the histogram divergences of two classes on these linear features as the evidence for classification.

Specifically, we seek for the most discriminating feature by maximizing the Kullback-Leibler (KL) divergence of the two-class histograms, which corresponds to the margin of the data. This feature is called KL feature. By incrementally learning KL features that can best discriminate the data, we construct an optimal feature set with a small number of KL features. Then, the coefficients that combine the weak learners are learnt to best combine the histogram divergences with the KL features so that the training error is minimized. Since the feature selection and parameter optimization are driven by a boosting framework, we call our technique Kullback-Leibler Boosting (KLBoosting).

It is, however, nontrivial to find the KL features in a high-dimensional space. We propose a data-driven Kullback-Leibler Analysis (KLA) to pursue KL features for image objects. The most discriminating wavelets are selected as the promising features. Then a sequential 1D optimization is proceeded along these promising features to find the optimal KL feature. This method is efficient particularly for high-dimensional image space.

By integrating KLBoosting framework and KLA, we learn a classifier for face detection. A cascade classifier is gradually learnt that covers $\pm 20^\circ$ in-plane and $\pm 20^\circ$ out-of-plane rotations. In a test with CMU+MIT data, the classifier can detect 95% faces with 10^{-6} false alarm rate, outperforming previous face detectors.

The rest of this paper is organized as follows. We introduce the main theory of KLBoosting in Section 2. KL Analysis for image objects is discussed in Section 3. In Section 4, the experiment of face detection is presented. After some discussion in Section 5, we conclude the paper in Section 6.

2. Kullback-Leibler Boosting Classifier

2.1 Linear features

Suppose that we are given a set of labeled samples $\{x_i, y_i\}_{i=1}^N$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, and are asked to give a decision y to any $x \in \mathbb{R}^d$. It is convenient for us to get some 1D statistics from the data, using a mapping function $\phi() : \mathbb{R}^d \rightarrow \mathbb{R}^1$. There are two kinds of mappings $\phi(x) = \phi^T x$ and $\phi(x) = \|\phi - x\|$, $\phi \in \mathbb{R}^d$. The former mapping is linear and $\phi(x) = \phi^T x$ is used in this paper as linear feature[1]. We restrict the linear feature to be a unite vector: $\|\phi\| = 1$.

Once we have a set linear features $\{\phi_i\}_{i=1}^k$, the form of the classification function is

$$F(x) = \mathrm{sign}[\sum_{i=1}^{k} \lambda_i(\phi_i^T x)], \tag{1}$$

where $\lambda_i()$ is a $\mathbb{R} \rightarrow \mathbb{R}$ discriminating function with respect to each feature ϕ_i. Eqn.1 can be viewed as a two-level forward neural network if $\lambda_i()$ is a truncated Sigmoid, or SVM if ϕ_i are support vectors and $\lambda_i()$ are kernel functions, or AdaBoost if $\lambda_i(\phi_i^T x)$ are base classifiers. Our goal is to obtain a robust and compact classifier using reliable functions $\{\lambda_i\}$ and a smallest feature set $\{\phi_i\}$.

2.2 Classification function

Along each feature ϕ_i, we may obtain the histograms of positive and negative samples $h_i^+(\phi_i^T x)$ and $h_i^-(\phi_i^T x)$, with certain weights. At a specific point $z = \phi_i^T x$, if $h_i^+(z) > h_i^-(z)$, it will be more likely from this evidence that x is a positive sample. Therefore we may assume $\lambda_i() = \alpha_i \log \frac{h_i^+()}{h_i^-()}$, which is proved in [3, 10] to maximize the margin. The classification function becomes

$$F(x) = \mathrm{sign}[\sum_{i=1}^{k} \alpha_i \log \frac{h_i^+(\phi_i^T x)}{h_i^-(\phi_i^T x)}], \tag{2}$$

with parameters $\{\alpha_i\}$ to balance the evidence from each feature. $\mathrm{sign}() \in \{-1, 1\}$ is an indicator function. We introduce a soft identity function that maps x to $[-1, 1]$

$$f(x) = \tanh[\sum_{i=1}^{k} \alpha_i \log \frac{h_i^+(\phi_i^T x)}{h_i^-(\phi_i^T x)}], \tag{3}$$

which implies $F(x) = \mathrm{sign}[f(x)]$.

[1] $\phi(x) = \|\phi - x\|$ usually corresponds to RBF function and can also be handled in KLBoosting. However, it is more difficult to find such kind of ϕ since it is not linearly additive – in general $\|\phi + \Delta\phi - x\| \neq \|\phi - x\| + \|\Delta\phi\|$. Therefore we select linear projection $\phi(x) = \phi^T x$.

2.3 KL feature pursuit

There are two terms to learn in the classification function (Eqn.2), the feature set $\{\phi_i\}$ and combining coefficients $\{\alpha_i\}$. To achieve a minimum set of features, we take a greedy strategy to gradually add the most discriminating feature to the feature set. Here we adopt the maximizing information gain criterion [20, 4], i.e., to maximize the Kullback-Leibler (KL) divergence of the positive and negative histograms projected on the feature. At iteration k, we may compute a symmetric measure of KL divergence

$$\mathrm{KL}(\phi) = \int [h_k^+(\phi^T x) - h_k^-(\phi^T x)] \log \frac{h_k^+(\phi^T x)}{h_k^-(\phi^T x)} d\phi^T x. \tag{4}$$

where $h_k^+(\phi^T x)$ and $h_k^-(\phi^T x)$ are the histograms of the positive and negative samples with weights $W_k(x_i^+)$ and $W_k(x_i^-)$, respectively. The most discriminating feature that maximizes the KL divergence

$$\phi_k^* = \arg\max_{\phi} \mathrm{KL}(\phi), \tag{5}$$

is called KL feature.

It is very difficult to optimize the KL divergence (Eqn.5). For low-dimensional data, we can simply use stochastic ascent such as Markov chain Monte Carlo (MCMC), to sample the feature sapce. But it does not work well for high-dimensional data such as images. We shall show how to solve this problem in Section 3.

2.4 Parameter learning

Once a set of features has been obtained, we should tune the coefficients $\{\alpha_i^*\}$[2] that best combine the histogram divergences from the features. In KLBoosting, we optimize the parameters $\{\alpha_i^*\}$ by minimizing the recognition error rate at the kth step

$$\{\alpha_i^*\}_{i=1}^k = \arg\min_{\{\alpha_i\}} \varepsilon_k, \tag{6}$$

where the recognition error is

$$\varepsilon_k = \frac{1}{N} \sum_{j=1}^{N} \delta(y_j \neq F(x_i)). \tag{7}$$

To find the optimal $\{\alpha_i^*\}_{i=1}^k$ is a nontrivial task especially when there are a large number of features. But it can help us a lot if we know the last optimal parameters $\{\alpha_i^*\}_{i=1}^{k-1}$. In initialization we fix the first $(k-1)$ α_i's as the last optimal one, and set $\alpha_k = 0$. In this manner, the initialization guarantees the recognition error to be no more than that of the last optimal parameters, i.e. $\varepsilon_k \leqslant \varepsilon_{k-1}$. Then we adopt a greedy algorithm to find better parameters for smaller recognition error.

```
Given: $(x_1, y_1), \cdots, (x_N, y_N)$, $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$
Initialize: $W_1(x_i^+) = \frac{1}{N^+}$, $W_1(x_i^-) = \frac{1}{N^-}$, $k=1$, $K_{max}$
Do
    • Feature pursuit: learn KL feature $\phi_k$ via Eqn.4
    • Parameter learning: learn $\{\alpha_i\}_{i=1}^k$ via Eqn.6
    • Compute recognition error $\varepsilon_k$ via Eqn.7
    • Update $W_{k+1}(x_i^+)$ and $W_{k+1}(x_i^-)$ via Eqn.8
    • $k = k + 1$
While $k < K_{max}$ and $\varepsilon_k \neq 0$
Output classifier:
    $F(x) = \mathrm{sign}[\sum_{i=1}^{k} \alpha_i \log \frac{h_i^+(\phi_i^T x)}{h_i^-(\phi_i^T x)}]$
```

Figure 1. The flowchart of KLBoosting learning.

2.5 Boosting by sample re-weighting

By means of KL feature pursuit and parameter learning, we can achieve the maximum drop of the recognition error by adding a feature. But how can we continue to drive these two steps to gradually decrease the training error? The answer lies in the boosting strategy. From the previous learnt classifier, we could increase the weight of misclassified samples while reducing the weight of recognized samples, and then learn a best feature to discriminate them. Assume that at step $(k-1)$ the weights of the positive and negative samples are $W_{k-1}(x_i^+)$ and $W_{k-1}(x_i^-)$, respectively. Then at step k we may re-weight the samples by

$$\begin{aligned} W_k(x_i^+) &= \frac{1}{Z^+} W_{k-1}(x_i^+) \exp\{-\beta_k y_i^+ f_{k-1}(x_i^+)\} \\ W_k(x_i^-) &= \frac{1}{Z^-} W_{k-1}(x_i^-) \exp\{-\beta_k y_i^- f_{k-1}(x_i^-)\} \end{aligned} \quad (8)$$

where Z^+ and Z^- are normalization factors for $W_k(x_i^+)$ and $W_k(x_i^-)$, respectively. Note the sequence β_k controls how fast to adapt the weight. We choose

$$\beta_k = (1 + c \cdot a^k) \log \frac{1 - \varepsilon_k}{\varepsilon_k} \quad (9)$$

where $c > 0$, $0 < a < 1$, and ε_k is the training error of current classifier. Typically we choose $c = 2$ and $a = 0.95$. In this manner, at the beginning of the training when the training error ϵ_k is a little big, the adapting factor β_k is big enough to significantly change the weights of the samples. Otherwise the sample re-weighting or boosting procedure would be very slow.

We use the soft identify function $f(x)$ instead of the classification function $F(x)$ in weight updating (Eqn.8), similar to what has been used in AdaBoost [11], to avoid frequent switching of the weight of the samples around the border. By means of this re-weighting algorithm, KLBoosting learns the features and classifiers efficiently. Figure 1 shows the flowchart of KLBoosting learning.

[2]These coefficients are empirically set in AdaBoost to be incrementally optimal.

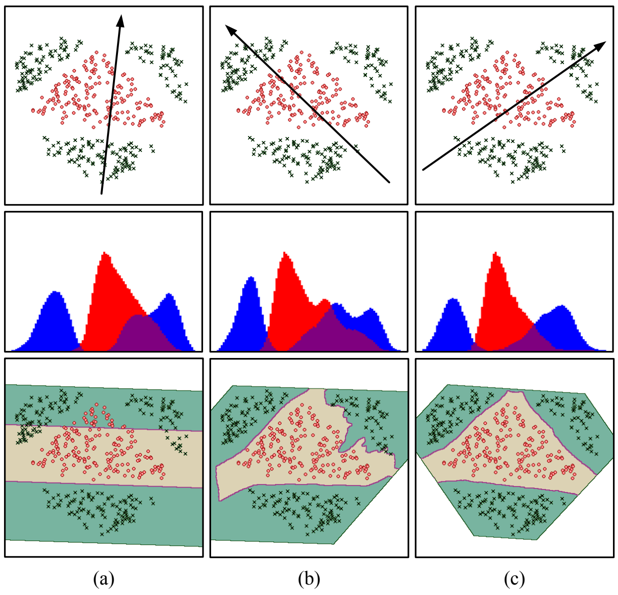

Figure 2. Illustration of KLBoosting learning. ○ and × denote positive and negative samples, respectively. (a) with one feature; (b) with two features; (c) with three features. Top: the original samples and the features (displayed as arrowed lines) in sequence. Middle: the weighted histograms projected on the corresponding features. Bottom: the decision manifold $F(x) = 0$ (displayed as solid curves). The features are learnt by stochastic ascent.

2.6 Examples

We now demonstrate the effectiveness of KLBoosting on two artificial non-linear separation problems. Figure 2 shows the simpler of the two problems. The positive samples lie within a triangle while the negative samples are distributed along with the three directions perpendicular to each side of the triangle. With KLBoosting, three features (with their corresponding histograms and classification manifolds) are found in succession by stochastic ascent, as shown in Figure 2(a)-(c). The final classification manifold, shown at the bottom of Figure 2(c), demonstrates that KLBoosting is capable of a reasonable classifier that not only gets the training error to zero, but also has good margins. Other boosting algorithms such as AdaBoost, can hardly get such a compact classifier with only three features.

The second separation example is a much more challenging one, in the form of interleaving spirals seen in Figure 3(a). Despite its complexity, KLBoosting is able to produce a very clean separation (after learning 32 features), shown in Figure 3(b). By examining the effect of the number of features on the fit error (Figure 3(c)), we see that the fit error for the training data drops to zero only after learn-

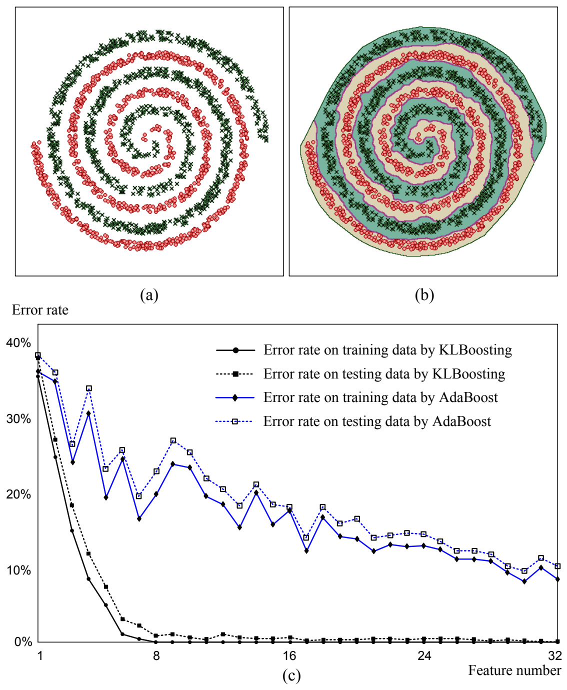

Figure 3. A complicatedly distributed example for testing the generalization ability of KLBoosting. (a) Synthetic data. $\circ$ and $\times$ denote positive and negative samples, respectively. (b) The decision manifold learnt by KLBoosting with 32 features, which are learnt by stochastic ascent. (c) The training and testing error rate vs. the number of features for KLBoosting and AdaBoost.

ing 8 features. The corresponding fit error for the test data is very small at 8 features, but eventually drops to zero at 30 learned features.

By comparison, the performance for AdaBoost is significantly inferior. For the AdaBoost algorithm, there are 1024 candidate features uniformly distributed in the 2D plane. In each iteration we select the best feature with minimum recognition error from the 1024 candidates, and generate a weak classifier by comparing the histogram. The weak classifiers are empirically combined to output a "strong" classifier. Clearly the error rate of AdaBoost converges more slowly and the testing error rate remains at a high value. This example demonstrates that our algorithm can use only a small number of most efficient features to classify even twisted data, and minimize the generalization error as more features collected.

3. Kullback-Leibler Analysis for Images

It is a nontrivial task to find the optimal linear feature that maximizes the KL divergence of the projected histograms, i.e. Eqn.4. For low-dimensional problems, e.g., Figure 2 and Figure 3, we simply use the stochastic ascent algorithm because the feature space has low dimensionality. But when we deal with practical problems, e.g., 20×20 image patches, the stochastic ascent algorithm becomes inefficient because sampling in a 400 dimensional space may be required.

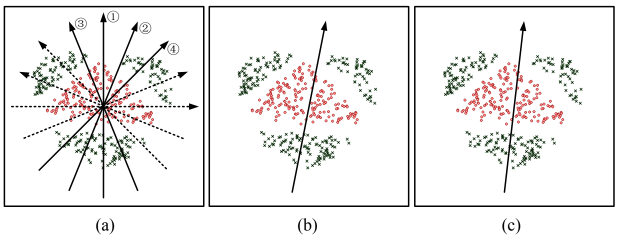

Figure 4. Using KLA to find the first feature for the example in Figure 2. (a) We select top 4 features from 8 candidates by ranking the KL divergence. (b) Initialization by combining the 4 features with KL divergence as weights. (c) KL feature pursued by sequential 1D optimization along the 4 features.

3.1 Sequential 1D optimization

Instead of stochastic ascent in high dimensional space, we propose a 1D optimization to find optimal features. Since the KL features ϕ's pursued in KLBoosting are unit vectors, there exists a boundary $[a_w, b_w]$ for any linear feature $w \in \mathbb{R}^d$ such that $a_w \leqslant w^T\phi \leqslant b_w$, for $\phi \in \mathbb{R}^d$,$\phi = 1$. Thus, given ϕ_0 we can do 1D optimization along the feature w by

$$\phi^*_{(w,\phi_0)} = \arg\max \mathrm{KL}(\beta w + \phi_0),\ \ \beta \in [a_w - w^T\phi_0, b_w - w^T\phi_0]. \tag{10}$$

In practice we can quantize the value along feature w and find the 1D optimum. Ideally, if we could have an infinite number of linear features $\{w_i\}$, we might find a satisfactory solution by sequentially doing 1D optimization along $\{w_i\}$. Practically, we only use a few most promising features. There are three steps for optimization

(1) Construct a feature bank $\{w_i\}$;

(2) Select the most promising features from the feature bank to form the feature set $\{w_i^*\} \subset \{w_i\}$;

(3) Sequentially do 1D optimization along the features in the feature set $\{w_i^*\}$.

We use this optimization strategy to revisit the KL feature pursuit problem for the example in Figure 2, focusing on the first feature. In Figure 4(a), we choose 8 directions uniformly distributed on the 2D plane, as the feature bank. We select top 4 by ranking the KL divergences along them. Then initialization is obtained by combining the 4 features with the KL divergence as the weights, as shown in Figure 4(b). Finally 1D optimization is sequentially proceeded along the 4 features and we get the KL feature in Figure 4(c). Note this one is exactly the same as the feature pursued by stochastic ascent, as shown in Figure 2(a). More interestingly, the sequential 1D optimization can be extended to high-dimensional space while stochastic ascent cannot.

```
Given: labeled samples $\{x_i, y_i\}_{i=1}^N$, weights $W(x_i^+)$,
    $W(x_i^-)$, wavelet bank $\{w_i\}_{i=1}^M$
• Compute $\mathrm{KL}(w_i)$ for each wavelet $w_i$ via Eqn.4
• Rank top $L$ wavelets by $\mathrm{KL}(w_i)$ to compose $\{w_i^*\}_{i=1}^L$
• Initialize: $\phi^{(0)} = \frac{1}{Z}\sum_{i=1}^L \gamma_i^{(0)} w_i^*$, $\gamma_i^{(0)} = \mathrm{KL}(w_i^*)$, $\eta = 0.1$,
    $H = 10$, $\mathrm{KL}^* = \mathrm{KL}(\phi^{(0)})$
For $k = 1$ to $K$
    For $i = 1$ to $L$
        • Neighborhood size: $s = \eta \mathrm{KL}(w_i^*)$
        • $\gamma_i^* = \gamma_i^{(k-1)}$
        // 1D optimization on feature $w_i^*$
        For $j = -H$ to $H$
            • $\gamma_i^{(k)} = \gamma_i^{(k-1)} + \frac{j}{H}s$
            • Compose to $\phi$ and compute $\mathrm{KL}(\phi)$
            If $\mathrm{KL}(\phi) > \mathrm{KL}^*$
                • $\mathrm{KL}^* = \mathrm{KL}(\phi)$, $\gamma_i^* = \gamma_i^{(k)}$
        • $\gamma_i^{(k)} = \gamma_i^*$
    • $\eta = 0.9\eta$
Output    $\phi^* = \frac{1}{Z}\sum_{i=1}^L \gamma_i^{(K)} w_i^*$
```

Figure 5. The pseudo code of KLA for images.

3.2 A data-driven approach

It is possible to find a feature bank for simple 2D problems, but difficult for high-dimensional space such as images. For example, in a d-dimensional space, to obtain a feature bank as dense as the 8 features in a 2D plane (Figure 4), 8^{d-1} features are needed! For image objects, since wavelets have been proven to be effective local features for images, we use an over-complete bank of wavelets plus variations of type, position, scale and orientation, to devise the feature bank $\{w_i\}$. Each wavelet is expanded to the image space such that $w_i \in \mathbb{R}^d$. We rank the wavelets with respect to the KL divergence $\mathrm{KL}(w_i)$ computed by Eqn.4. The top L wavelets are selected to form the most efficient feature set $\{w_i^*\}$. $L >> d$ to ensure satisfactory solution, which implies $\mathrm{span}\{w_i^*\} = \mathbb{R}^d$. In this manner, ϕ could be decomposed to and reconstructed by the feature set $\{w_i^*\}$

$$\phi = \frac{1}{Z}\sum_{i=1}^{L} \gamma_i w_i^*, \tag{11}$$

where γ_i is the decomposition coefficient, and Z is a normalization factor that makes ϕ be a unit vector. The initialization of the KL feature is given by combining the wavelets with their KL divergence as weights. Then a sequential 1D optimization is carried out along each wavelet. Since it is difficult to find the exact boundary $[a_w, b_w]$ in Eqn.10 for each feature, the 1D optimization is done within a neighborhood. As the size of the neighborhood gradually diminishes, the optimization procedure converges to a satisfactory solution. The pseudo code of KL Analysis is shown in Figure 7.

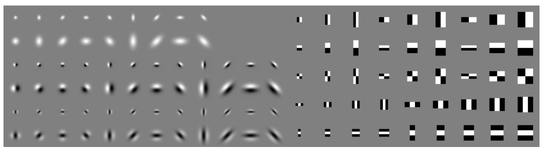

Figure 6. Wavelet bank. Left: Gaussian. Right: Harr.

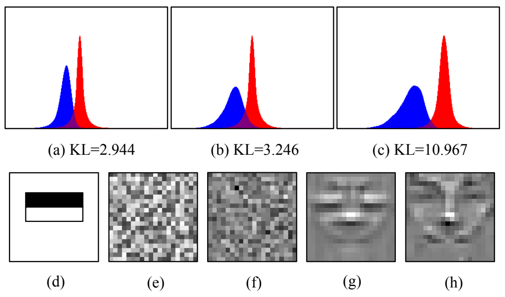

Figure 7. Comparison of the best wavelet, MCMC feature and KL feature. From (a) to (c) are the histograms of face and non-face patterns projected on the best Harr wavelet, MCMC feature and KL feature. Blue: face, red: non-face. From (d) to (h) show the best Harr wavelet, initialized and the final result of MCMC feature, initialized and the final result of KL feature, respectively.

3.3 KL feature for face and non-face patterns

We use face and non-face patterns to test the KLA algorithm. The data collection will be mentioned in next section. Both face and non-face patterns are represented by 20×20 patches. We choose an over-complete wavelet bank comprising Gaussian family (Gaussian itself, the 1st and 2nd order derivatives of Gaussian) and Harr wavelets with 3 and 5 prototypes respectively, as shown in Figure 6. Using scale, orientation and tilt transforms, we generate 111 wavelets for each position, and there are in total $111 \times 400 = 44400$ local wavelets. We also choose some large scale wavelets located around the center of the pattern to capture global features, 40 Gaussian families and 40 Harr wavelets for each position within an inner 10×10 rectangle and the total number is $80 \times 100 = 8000$. From these 52400 wavelets we select top 2800 to compose the global KL feature. Since there is significant redundancy in wavelet bank, for each position, we limit the number of either local small wavelet or large-scale wavelet to no more than 6, so that we have nearly identical number of wavelets at each position.

We compare the best wavelet feature, the best MCMC feature by stochastic ascent, with the best KL feature pursued by the proposed data-driven KLA algorithm, in Figure 7. The histograms of face and non-face patterns projected on the best Harr wavelet, MCMC feature and KL feature are plotted from (a) to (c), with KL divergence 2.944, 3.246 and 10.967, respectively. The corresponding features are displayed as images from (d) to (h). The Harr wavelet feature merely describes the difference around the eye area. MCMC feature is noisy and not meaningful. On the other hand, the KL feature corresponds to face semantics very well, emphasizing the appearance of facial components and ignoring the backgrounds. Therefore, the KL feature is optimal not only because it maximizes the KL divergence, but also because it makes sense to human perception.

4. Applications to Face Detection

By integrating the KLA algorithm for images, we can apply KLBoosting to detect faces in images. This is known to be a very challenging problem due to possible clutter and varying head pose and lighting conditions.

4.1 A brief review on face detection

Most face detection approaches that use a two-class classifier are appearance-based [18]: face and nonface samples are represented as small square patches (typically 20×20), then a classifier is trained from the samples. There are basically two approaches to learning the face vs. non-face boundary. One is distribution based, using Gaussians [5] or mixtures of subspace models [19] to learn the face distribution. The classifier is the result of thresholding the pdf. This type of approach tends to produce more false positives. The other face detection approach is to directly learn the classifier, using Neural Networks [8], Support Vector Machines [6], naive Bayesian [12], SNoW network [17] and AdaBoost [16]. There is also a combination of these two approaches [14]: the pdf measures of each cluster of face and non-face are input to a neural network for classification. Most of the classification-based methods deal well with upright and frontal-view faces. The detection rate can reach 94% when the false positive rate is limited to 10^{-6}. Nevertheless, a single classifier can hardly deal with large in-plane or out-of-plane rotations. Pose estimation/correction [9] and multiple classifiers for specified poses [13] have been proposed to solve this problem.

4.2 Experimental setup

Our experiments are conducted with many face and non-face images. Our face data were obtained from some standard face database such as FERET and AR, and from the web. There are a total of 8760 faces, and by mirroring, we obtained 17520 face samples covering different lighting, orientation, pose, imaging quality, age, gender, with(out) occlusion and with(out) facial additions. Both the in-plane and out-of-plane rotations were restricted to $\pm 20^{\circ}$. We use five points, the center of the eyes, the tip of the nose and the corners of the mouth, to align all the samples. In our experiments, we cropped a 20×20 face patch from each image to

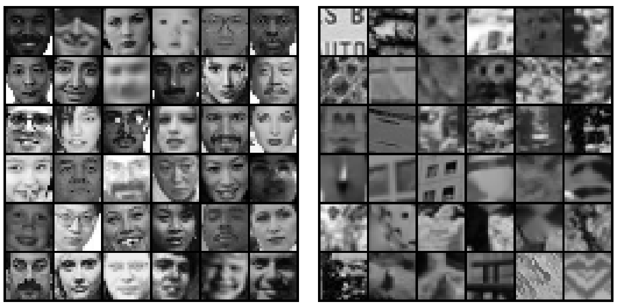

Figure 8. Some samples in the training data, (a) face patterns (b) non-face patterns. Some of the non-face patterns are similar to a face.

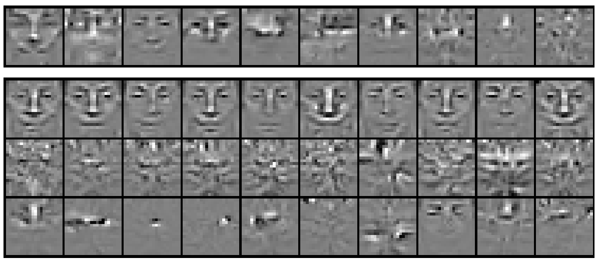

Figure 9. Face features pursued by KLA. The first row lists the first ten features sequentially found in KLBoosting. The others are typical and representative ones in the 400 learnt features. The second, third and last rows are “global semantic”, “global but not semantic” and “local” features, respectively.

form the face training data. We gathered 2484 images not containing faces, e.g. natural scenes, buildings and textures. Using a Gaussian pyramid, we extract totally 1,339,856,947 non-face 20×20 patches to form the non-face training data. Some of the face and non-face patterns are displayed in Figure 8. The number of histogram bins is set to be 300, as a trade-off between accuracy and sample number.

In face detection, the most important is to find the non-face (negative) samples that stand near the classification boundary. In face detection literature, booststrapping algorithm is always chosen to gradually find the proper non-face samples [14, 8, 6]. A cascade of classifiers is proposed in [16] to speed up detection procedure by gradually rejecting non-face patterns. We follow this architecture and train the cascade of face detectors. For each classifier in the cascade, we set the detection rate to be no less than 99.99%, namely the false positive rate is no bigger than 0.01%, while minimizing the false alarm rate. If the false alarm rate is under 35% or the number of the features exceeds a preset number like 30, the learning of this classifier stops. The learnt classifier searches in the non-face images to generate new non-face samples for training the next classifier. At detection line, we detect all possible faces in a Gaussian pyramid down-sampled by 1.11. Then the multiple detections are

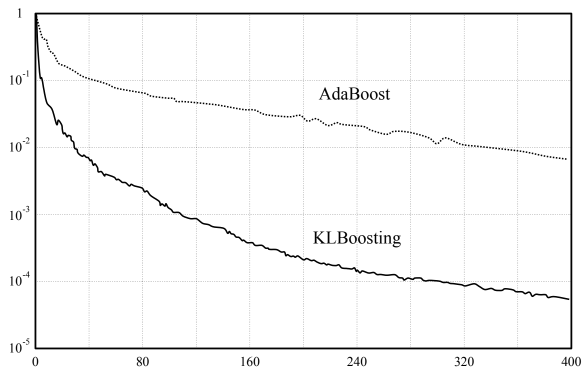

Figure 10. The curves of the false alarm rate ($\geqslant 99.99\%$ face recognition rate) vs. the number of features added, for both KLBoosting and AdaBoost.

merged in the same fashion as [16].

4.3 Experimental results

We list some of the learnt features in Figure 9. The first row shows the first ten features sequentially learnt in KLBoosting. There are in general three categories of features, shown from the second to the last row, respectively. The first is "global semantic", similar to a face pattern, to describe facial characteristics such as facial components, shape, expressions and lighting. The second is "global but not semantic", like a non-face pattern, to distinguish the frequency distributions of face and non-face. The last is "local" features like wavelets, to emphasize detailed local difference.

We use in total 22 classifiers in the cascade to learn the face detector, including 450 features. By adding one feature, the false alarm rate will decrease, which indicates that some of the non-face patterns can be cleanly rejected nearly without loss of any faces. The curve of false alarm rate vs. the number of the features is shown in Figure 10. We also implemented the AdaBoost algorithm in [16] for comparison. The fist classifier in KLBoosting uses two features to retain 100% face detecting rate with false alarm rate 10.8%. While in [16] two wavelets (local features) are used in the first cascade with about 40% false alarm rate at the same detecting rate. After learning 400 features, KLBoosting achieves false alarm rate about 5.3×10^{-5}, while the value for AdaBoost is 6.7×10^{-3}. This demonstrates that KLA efficiently learns the most discriminating features and KLBoosting combines these features very well.

We tested the KLBoosting face detector on the MIT+CMU frontal face test set [8], and used an approach similar to [16] to produce the ROC curve shown in Figure 11. We simply adjust the threshold of the final classifier from -1 to 1 with step 0.05 to count the detection rate vs. false alarm rate. We also plot the ROC curves of AdaBoost and Neural Network that have been reported in [16]. Despite the inevitable differences in the training sets, param-

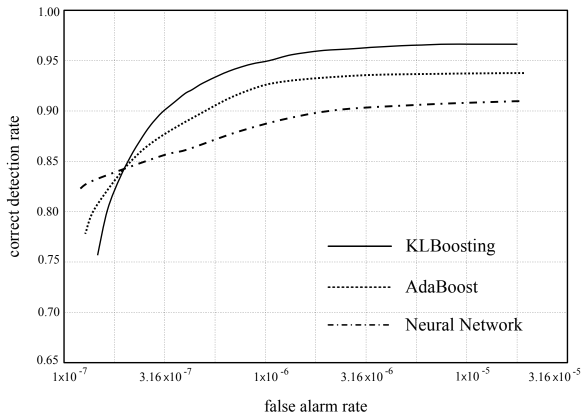

Figure 11. ROC curves for KLBoosting, AdaBoost [16] and Neural Network [8] on the MIT+CMU test set. There are int total 85,360,210 patches scanned.

eter tuning, and multi-detection merging strategies, our algorithm appears to outperform others, e.g., 95.0% detection rate at 10^{-6} false alarm rate. Note when the false alarm rate is close to 10^{-7}, the performance of AdaBoost and Neural Network is better than our approach because there are three independent classifiers fused for decision. These independent classifiers can also be fused to improve the performance of KLBoosting for future work. Some detection results are displayed in Figure 12.

Since KLBoosting classification only requires linear projection and table lookup, it is computationally efficient. It takes on average 0.4 second on a Pentium 4 1.8GHz PC to detect a few faces from a 320×240 image.

5. Discussion

(a) KLBoosting vs. AdaBoost

AdaBoost combines several weak classifiers (features) to compose a "strong" classifier. It left two questions unsolved. First, how to best combine the weak classifiers in terms of the coefficients; second, how to choose the best weak classifiers or features. KLBoosting solves the first problem by iteratively tuning the coefficients to minimize the recognition error. It ensures the recognition error would not increase as more features involved. To solve the second problem, KLBoosting finds KL features that maximize the symmetric KL divergence between two classes, which corresponds to the projected margin. In other words, KLBoosting uses optimal features to constitute an optimal classifier.

(b) KLA vs. FLD

Both KLA and Fisher Linear Discriminant (FLD) are projection pursuit methods, i.e. to find a feature that can best discriminate two-class data. FLD assumes the two class are both Gaussian distributed, and maximizes the distance between two class over the variation within each class. It does not work very well for non-Gaussian problems, such

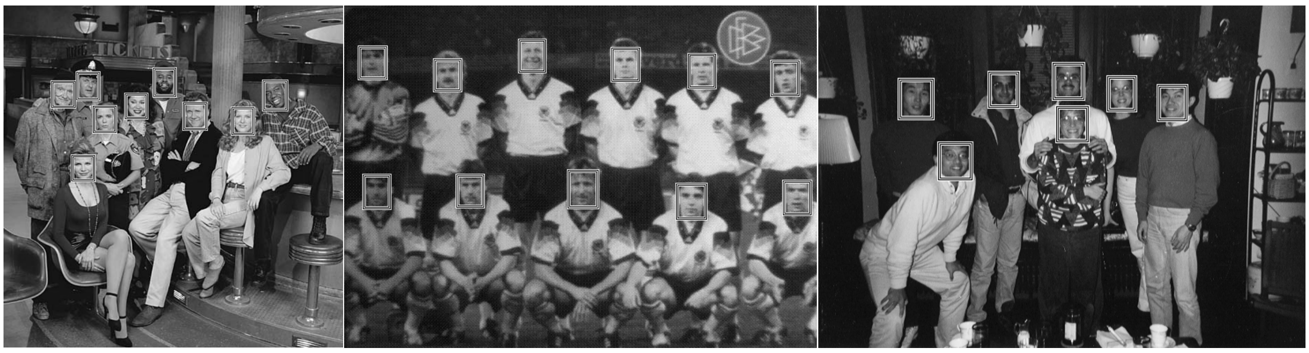

Figure 12. Output of KLBoosting face detector on some test images from CMU face test set.

as the toy problems in Figures 2 and 3. KLA is a more unified method and can be equal to FLD if the two classes are Gaussian distributed.

(c) KL feature vs. wavelet

The statistics on wavelets, such as Gaussian derivative, Gabor and Harr, are often used as the observations of images, and the evidence to construct classifiers or distributions [7, 16, 20]. However, wavelets are not designed for the specific class, e.g. human face, and as local features, not capable to capture the global statistics, e.g. the correlation of facial components. KL feature is a global feature, to capture not only local statistics, but also global properties distinguishing the data.

6. Conclusion

In this paper we have proposed KLBoosting as a classification framework and applied it to face detection. In KLBoosting, we have a *compact* classifier with *robust* features. We learn the coefficients with the weak classifier and find the KL feature that best discriminates the data. Moreover, we propose a data-driven KLA to learn the KL feature in image space. The toy problems and the application of face detection demonstrate the feasibility of KLBoosting.

References

[1] C.M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press Oxford, 1995.

[2] P. Buhlmann and B. Yu. Invited discussion on ‘Additive logistic regressions: a statistical view of boosting (Friedman, Hastie and Tibshirani)’. *The Annual of Statistics*, 28(2):377–386, April 2000.

[3] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annual of Statistics*, 28(2):337–374, April 2000.

[4] C. Liu, S.C. Zhu, and H.Y. Shum. Learning inhomogeneous gibbs model of faces by minimax entropy. *ICCV*, pages 281–287, 2001.

[5] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *TPAMI*, 19(7):696–710, 1997.

[6] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an applications to face detection. *CVPR*, pages 130–136, 1997.

[7] C. Papageorgriou, M. Oren, and T. Poggio. A general framework for object detection. *ICCV*, pages 555–562, 1998.

[8] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *TPAMI*, 20(1):23–38, 1998.

[9] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. *CVPR*, pages 38–44, 1998.

[10] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Proc. of Annual Conf. on Computational Learning Theory*, pages 80–91, 1998.

[11] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Proc. Int’l Conf. on Machine Learning*, 1997.

[12] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. *CVPR*, pages 45–51, 1998.

[13] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. *CVPR*, 1:746–751, 2000.

[14] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *TPAMI*, 20(1):39–51, 1998.

[15] V.N. Vapnik. *The nature of statistical learning*. Springer, 1995.

[16] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[17] M.-H. Yang, N. Ahuja, and D. Kriegman. Mixtures of linear subspaces for face detection. *AFGR*, pages 70–76, 2000.

[18] M.-H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *TPAMI*, 24(1):34–58, 2002.

[19] M.-H. Yang, D. Roth, and N. Ahuja. A snow-based face detector. *NIPS*, pages 855–861, 2000.

[20] S.C. Zhu, Y.N. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Nueral Computation*, 9(8), 1997.