Randomized Intraclass-Distance Minimizing Binary Codes for Face Recognition

Hao Zhang J. Ross Beveridge Quanyi Mo Bruce A. Draper
Colorado State University
Fort Collins, CO 80523
{zhangh, ross, qmo, draper}@cs.colostate.edu

P. Jonathon Phillips
National Institute of Standards and Technology
jonathon.phillips@nist.gov

Abstract

A new algorithm for learning binary codes is presented using randomized initial assignments of bit labels to classes followed by iterative refinement to minimize intraclass Hamming distance. This Randomized Intraclass-Distance Minimizing Binary Codes (RIDMBC) algorithm is introduced in the context of face recognition, an area of biometrics where binary codes have rarely been used (unlike iris recognition). A cross-database experiment is presented training RIDMBC on the Labeled Faces in the Wild (LFW) and testing it on the Point-and-Shoot Challenge (PaSC). The RIDMBC algorithm performs better than both PaSC baselines. RIDMBC is compared with the Predictable Discriminative Binary Codes (DBC) algorithm developed by Rastegari et al. *The DBC algorithm has an upper bound on the number of bits in a binary code; RIDMBC does not. RIDMBC outperforms DBC when using the same bit code length as DBC's upper bound and RIDMBC further improves when more bits/features are added.*

1. Introduction

Binary codes play an important role in many classification/recognition tasks. In biometrics, for example, they are the basis for most iris recognition algorithms[19]. More generally, hashing techniques for large-scale object recognition measure similarity as Hamming distances between binary codes [7, 8]. The goal is to learn hash functions such that samples from the same category have small Hamming distances between their codes, while samples from different categories have large distances. The resulting binary codes facilitate fast nearest-neighbor search using hash tables and/or Hamming ball queries, achieving query times that are independent of dataset size. Even in classic nearest-neighbor search settings, where query times scale linearly with the size of the dataset, binary codes can accelerate nearest-neighbor search by more than two orders of magnitude on modern processors [8].

There has been significant progress on learning binary codes to tackle conventional image classification tasks such as Caltech256 [9] and fine-grained image classification tasks such as Caltech-UCSD Birds-200-2011 [6]. However, these datasets have fixed numbers of classes, e.g. 256 for Caltech256. In comparison, face datasets equate class with identity: more people means more classes. People used for training must differ from those used to test an algorithm. Only a few works [14, 12] have been proposed that use binary codes to perform face recognition.

Individual binary classifiers may be associated with semantically meaningful binary attributes, e.g. gender. Kumar *et al*. [11] present such an approach, although in their case attributes are encoded as continuous values indicating the degree of presence of that attribute, e.g. maleness, rather than bits. Semantic attributes have the advantage that the individual bits are meaningful to humans. Attributes have the disadvantage that the number of known meaningful attributes is small, while the cost of obtaining semantically labeled data to train them with is large.

An alternative is to learn binary codes from data. Semi-supervised and supervised methods employ category labels of images or a match/non-match indicator of image pairs. In the context of face recognition, the goal is to learn binary codes that place images of a given person close to each other in Hamming space. The state-of-the-art binary code learning method, Predictable Discriminative Binary Codes (DBC [17]), therefore tries to optimize Fisher's criterion by both minimizing the Hamming distance between images from the same class and maximizing the Hamming distance

[1]CSU was funded in part by the Department of Defense through the Technical Support Working Group (TSWG). PJP was supported in part by the FBI.

between images from different classes.

We present a new algorithm that learns largely uncorrelated binary codes by minimizing the Hamming distances within classes. Each bit is determined by a linear classifier and a goal is a classifier that assigns all images of a given person the same bit label. A random assignment of labels, 0 or 1, to people is used for initialization. Then an iterative procedure trains a classifier, updates label assignments based on that classifier, and repeats until convergence. This learning procedure generates highly uncorrelated bits in part because of its randomized initialization. It also avoids naive solutions, such as assigning all images the same label.

The algorithm presented here is simpler than DBC, and to some it may appear less sophisticated. After all, it only minimizes intraclass (with-in class) similarity. As a result, however, it avoids two limitations that follow indirectly from maximizing between-class distances. The first is that if we view every bit of a binary code as a weak classifier, DBC can only learn classifiers that partition the data evenly, i.e. produce roughly as many 1's as 0s. This excludes many useful partitions of the data. Second, it limits the number of classifiers (i.e. the length of the code) to the number of dimensions in the input data if the features are to remain sufficiently uncorrelated. Removing the interclass distance constraint allows our new algorithm to produce longer and more discriminative binary codes.

In the experiments below, RIDMBC and DBC algorithms are trained on the Labeled Faces in the Wild (LFW) dataset and evaluated on the Point-and-Shoot Challenge (PaSC). This experimental protocol addresses a possible concern about automatically trained binary codings. Namely, are they overly specific to the dataset on which they are trained? Our results show cross-dataset generalization and the resulting RIDMBC algorithm outperforms both DBC and the two PaSC baseline algorithms.

2. Related Work

We begin by discussing the Predictable Discriminative Binary Codes (DBC) already mentioned in the introduction. DBC is close enough to our own to warrant direct comparison. Additional works related to binary codes and hashing are then discussed. Finally, the related topic of attribute-based face recognition is discussed.

2.1. Predictable Discriminative Binary Codes

Rastegari *et al*. [17] propose a representation of images using binary codes called predictable discriminative binary codes (DBC) which minimizes the Hamming distances between images from the same category while maximizing the Hamming distances between images of different categories. They formulate this goal as an optimization problem as follows: (we only show the key parts)

$$\arg\min_{\{\omega\}} \sum_{c\in 1:C} \sum_{m,n\in c} d(B_m, B_n) - \lambda \sum_{\substack{c'\in 1:C \\ p\in c'}} \sum_{\substack{c''\in 1:C \\ c'\neq c'', q\in c''}} d(B_p, B_q) \tag{1}$$

where B_* denotes a binary code, C is the total number of classes and $\{\omega\}$ represents the set of bit encoders (linear support vector machines). The first and second term correspond to intraclass and interclass distances, respectively. DBC produces the state-of-the-art result on Caltech256 and outperforms state-of-the-art binary code method on ImageNet [5].

To achieve Equation 1, each bit is learned such that the majority of its values assigned to each class is larger than the same value assigned to the rest of the classes (measured in percentage). This encourages the bit encoders to partition the data evenly. However, partitioning the data this way tends to create correlated bit encoders. To circumvent this problem, Rastegari *et al*. initialize the binary codes by projecting them onto orthogonal dimensions derived from Principal Component Analysis (PCA). While this step reduces correlation, it also imposes a hard constraint on the total number of possible bits to learn. Because of PCA projection, it is limited by either the number of training samples or the dimensionality of the underlying features.

2.2. Hashing Related Methods

Weiss *et al*. propose Spectral Hashing [18] that relates the search for a good binary representation to a specific form of graph partitioning. This analysis not only shows that the problem of finding the best binary code is NP-Hard, but also suggests that a relaxed version of the problem can be solved by spectral analysis. In Spectral Hashing, binary codes are generated by thresholding a subset of eigenvectors of the Laplacian of the similarity graph. Both learning codes and generating codes for new data can be done efficiently.

In [7], Gong *et al*. propose to formulate the binary code learning as finding the best rotation of PCA projected data that minimizes the quantization error of the rotated data to the vertices of a zero-centered binary hypercube. An iterative quantization method (ITQ) is proposed to solve the problem. Both an unsupervised and a supervised version of ITQ are provided in their paper while the former uses PCA and the latter uses Canonical Correlation Analysis (CCA). Unsupervised methods, including Sepctral Hashing and ITQ with PCA, often assume the Euclidean distance as the true measure of the similarity while supervised methods may be able to learn more discriminative distances.

Lin *et al*. [12] propose to learn a hash function such that semantically similar objects are closer in Hamming space. In particular, they propose to minimize the Kullback-Leibler divergence between a distribution defined by the

ground truth affinity matrix and a distribution defined by the learned Hamming distances. Two fast linear time approximate methods are adopted with which their method outperforms Spectral Hashing on the LabelMe dataset. On a face dataset collected at Google which contains 276,436 face images of 3703 celebrities, they achieve an accuracy close to the Neven Vision face recognition system[1] [16] when using 1200 bits to represent a face image.

2.3. Attributes, Simile & 'Tom-vs-Pete' Algorithms

Kumar *et al*. [11] use attributes such as gender, race, and eye color to measure similarity among faces. The face attributes are predefined manually, and classifiers for the attributes are learned from human labeled training data. Test images of new faces are input to every classifier, producing vectors of attribute scores that are used as intermediate representations of the test images. A verification classifier is then trained to determine if pairs of attribute vectors represent matching or non-matching images. In addition to attribute classifiers, Kumar *et al*. also learn a number of simile classifiers to determine how similar a face image is to a predefined model (e.g., Brad Pitt's nose).

Berg and Belhumeur [1] learn many binary "Tom-vs-Pete" classifiers that distinguish between pairs of subjects in a reference set. These classifiers analyze different regions of the face images. As a result, they tend to be decorrelated, but face alignment becomes critical; Berg and Belhumuer introduce an identity-preserving alignment process to address this issue. During testing, the vector of distances to the decision boundary of "Tom-vs-Pete" classifiers is used to represent an image, and a second-layer classifier makes yes-or-no decisions about whether a pair of images match.

These three approaches should not be confused with binary code methods. They all use intermediate representations defined in the space of R^n. Recall, attribute approaches encode 'maleness' not merely 'male yes or no'. In contrast, binary code learning methods yield representations residing in the space of B^n, where each of the n element is either 0 or 1.

3. The RIDMBC Algorithm

Our goal is to construct an algorithm that projects face images to binary strings and performs face recognition in Hamming space. Each bit is learned by a linear classifier with the goal of assigning the same binary value to all images belonging to each person.

[1]The identification of any commercial product or trade name does not imply endorsement or recommendation by NIST.

Each linear classifier optimizes the following:

$$\begin{aligned} \arg\min_{\omega,\xi_k,\hat{l}_k} \quad & \sum_{c\in 1:C}\sum_{i,j\in c}\|b_i - b_j\| + \gamma\|\omega\|^2 + \lambda\sum_{k\in 1:N}\xi_k \\ \text{subject to} \quad & \hat{l}_k(\omega^T x_k) \geq 1-\xi_k, \quad \xi_k \geq 0 \end{aligned} \tag{2}$$

where ω is the linear bit encoder, C is the total number of classes, N is the total number of training samples, $x_k (k = 1,\ldots,N)$ is a feature representation of an image, ξ_k is the slack variable for x_k and $\hat{l}_k$ is its training label. Here $b_k = sign(\omega^T x_k)$ is the predicted bit value for x_k. It is equal to 1 when $\omega^T x_k$ is positive and -1 otherwise.

Equation 2 presents a hard optimization problem. We find locally optimal solutions by iterating two steps. In one step, we fix the decision boundary ω and minimize the third term corresponding to the slack variables by changing the training labels $\hat{l}_k$ of the data. In the other step, we train a linear SVM given the new $\hat{l}_k$ to update ω. While this procedure does not guarantee descent in each iteration, we found in our experiments that descent is achieved in each iteration and convergence is obtained in only a few iterations. There is a degenerate solution assigning the same label to all samples, but in practice convergence on this degenerate solution has never been observed for the training data used in this paper.

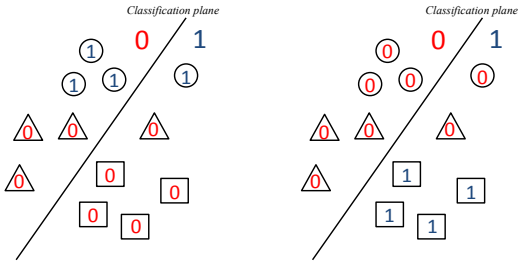
The iterative solution process begins by assigning binary values, 0 or 1, to people in a training set S. The initial assignment is random, but is performed at the level of people, not images. Thus the same label is assigned to all images of any one person. This procedure roughly balances the total number of 0s and 1s. In practice, this random initialization results in highly uncorrelated and roughly balanced bit assignments, as shown in Section 4.2.

Once initialized, the training set is divided into two subject disjoint image subsets, S_1 and S_2, to prevent overfitting. A linear SVM classifier is trained on S_1 based upon the initial randomized assignment of labels. This classifier is then used to assign labels to S_2. In general, some of the labels assigned to images in S_2 will violate the constraint that all images of a given person are assigned the same label. The assignment of labels to subjects is then adjusted. To carry out this adjustment, whatever label is initially assigned to the majority of that subject's images is then assigned to all images of that subject. This procedure is illustrated in Figure 1.

When the label assignments are adjusted, a new SVM is trained on S_2. It is then evaluated on S_1 in what is now an iterative process with S_1 and S_2 switching roles on each iteration. By design, S_1 and S_2 share no people in common. The entire procedure iterates until it converges, where the test for convergence is that the labels assigned to people are no longer changing. This means that the labels assigned to either subset are consistent with the ones assigned to the

other subset. In other words, an SVM learned on either subset predicts well the labels on the other subset. After convergence, there is one final step where S_1 and S_2 are recombined and a final linear SVM is trained on both sets. The estimation of the SVM on an image returns a binary answer which composes one bit in its binary code representation.

Algorithm 1 summarizes the learning process for our binary classifiers. Any number of such classifiers can be learned by randomly initializing with different label assignments. We will later demonstrate the effectiveness of using random initialization in our experiments. Each random label assignment will result in one classifier (bit) in the binary code representation. Our experience suggests the accuracy for a single bit typically ranges between 68% and 73% when averaged over S_1 and S_2. Also, the procedure typically converges in fewer than 25 iterations. Consequently, it is both feasible and desirable to construct thousands of bits. After each image is represented as a binary code (bit string), the similarity between a pair of images is computed as their Hamming distance.



(a) Before Label Change (b) After Label Change

Figure 1. Binary labels for three subjects, denoted by shapes, are shown relative to an SVM decision boundary. Part (a) shows the labels prior to label adjustment. Two subjects, depicted as circles and rectangles, lie in majority (circle) or totally (square) on the wrong side of the boundary. Their labels are therefore adjusted to agree with the label assigned to the majority of their instances, as shown in part (b).

4. Training and Pilot Studies on LFW

We show the generalization abilities of our algorithm by training it on images from the Labeled Faces in the Wild (LFW) dataset and evaluating it on images from the Point-and-Shoot Challenge (PaSC) dataset. Section 4.1 presents details of training our algorithm on LFW. Evaluations on PaSC are reported in Section 5. Figure 2 shows example pictures from LFW and PaSC. The top row shows four aligned images of the same person in LFW and the bottom row shows four aligned images of the same person in PaSC. We can see that PaSC images are even more challenging with low-resolution images and large pose/expression/illumination variations.

Algorithm 1 Learning Bit Encoders

INPUT: Two subject disjoint image sets S_1 and S_2,
for i from 1 to k, the number of bits **do**
  Generate random $\hat{L}_1$ and $\hat{L}_2$ desired labels
  Initialize Training Data = $\{S_1, \hat{L}_1\}$
  Initialize Test Data = $\{S_2, \hat{L}_2\}$
  while labels in L_1 or L_2 changing **do**
    Train a classifier on Training Data
    Test the classifier on Test Data
    for Each subject j in Test Data **do**
      Let $L_j = \{l_{j,1}, l_{j,2}, ..., l_{j,m}\}$ be the predicted labels of all the images belonging to that subject
      Let $\hat{L}_j = \{\hat{l}_{j,1}, \hat{l}_{j,2}, ..., \hat{l}_{j,m}\}$, be the corresponding desired labels, satisfying $\hat{l}_{j,1} = \hat{l}_{j,2} = ..., = \hat{l}_{j,m}$
      if majority of L_j not equal to $\hat{L}_j$ **then**
        $\{\hat{l}_{j,1}, \ldots, \hat{l}_{j,m}\} = \{1 - \hat{l}_{j,1}, \ldots, 1 - \hat{l}_{j,m}\}$
      end if
    end for
    Swap the Training Data and Test Data.
  end while
  Combine $\{S_1, \hat{L}_1\}$ and $\{S_2, \hat{L}_2\}$ and train classifier a_i.
end for
Output: bit encoders

Figure 2. Examples from PaSC and LFW. Top row: aligned images of the same person in LFW. Bottom row: aligned images of the same person in PaSC.

4.1. Algorithm Input Features & Training

In this section, we present the features that we adopted for learning binary codes and the training data we use.

4.1.1 Input Features

For LFW, we use the provided images which are aligned by a commercial face alignment software. We align PaSC images according to the same rule as LFW. In a result, the aligned images used for both datasets are 250×250 pixels, with eye coordinates at fixed locations. Before extracting features, the images are further cropped to a 128×170 rectangle containing the face. Three different input representations are then extracted. The first is gray-scale pixel values,

resampled from the cropped window to a size of 32×42. The second representation is Local Binary Pattern (LBP) features, extracted from a 64×85 version of the face image and computed with a sampling step of 8 and a radius of 1, using the open source package "mahotas" [4]. The third representation is also based on LBP, this time with a sampling step of 8 and a radius of 2. We call these features gray, LBPr1, and LBPr2, respectively.

We also adopt a simple rule to compute a Hamming distance for an image pair when combining all three features:

$$d = \frac{d_{gray}}{l_{gray}} + \frac{d_{LBPr1}}{l_{LBPr1}} + \frac{d_{LBPr2}}{l_{LBPr2}} \quad (3)$$

where d_* is the Hamming distance computed from the particular feature and l_* is the maximal value of that Hamming distance, which is also the number of bits.

4.1.2 Training Data

Our algorithm is set up on the Labeled Faces in the Wild (LFW) dataset [10]. This dataset is composed of 13,233 images of 5749 people. It is partitioned into 2 views. View 1 is further divided into a training part and a test part without subject overlap. Although our emphasis is on generalizing across training sets, we conduct several pilot studies on LFW. We first examine the independence of the learned binary bits. Then a direct comparison between our technique and DBC is performed on View 1 data. Finally, pilot evaluations on View 1 are reported.

In all experiments carried out below, LFW View 1 training data are exclusively used as our training data. They contain 3443 images of 2132 people. The reason for doing this is to obtain a set of fixed binary classifiers to quickly generate a binary code for any future test image. The binary classifier associated with each bit is a linear SVM implemented using Libsvm package [3]. For computational efficiency, the SVMs we learn do not have a bias (offset) term. For each kind of feature, we learn a binary code of length 2000. Our experience suggests that using more bits will only yield a slight improvement. In all cases, the Libsvm parameter $C = 10$.

4.2. Pilot Study: Independence Between bits

Since the similarity is defined as the Hamming distance where each bit is weighted equally, it is important to know whether the learned bits are uncorrelated to each other. The more uncorrelated they are, the more powerful they will be when combined. To quantify the degree to which bit encoders make the same identity decisions over a set of image pairs, the following experiment is carried out.

View 1 test data are subject-independent to View 1 training data. They contain 1000 image pairs, 500 of which are from the same person while the rest are from two different people. For each of the 1000 image pairs, a pair of bit classifiers l_i and l_j are run. For l_i, a bit string of length 1000 is constructed where a 1 indicates the labels $l_i(q)$ and $l_i(t)$ match between the query image q and target image t in the pair. This essentially represents the decision of whether the pairs match according to bit l_i. The same procedure is then repeated for classifier l_j and a second bit string of length 1000 is generated. The degree to which the decisions from l_i and l_j correlate is captured by the Hamming distance between these two strings.

Figure 3 presents histograms of the normalized Hamming distance (ordinary Hamming distance divided by bit length) between all pairs of bits. The histograms are normalized such that the integral of each histogram is 1. We observe that these histograms are centered at 0.5, representing a strong trend toward independence. There are no examples of pairs of binary encoders with significant, say 75% or higher, agreement. These results bolster our confidence in the independence of the bit encoders found using Algorithm 1 and suggest the iterative procedure with different random initializations is converging on different local optima. Given our desire for independent bits, the existence of so many local optima resulting from random initializations is a virtue.

4.3. Pilot Study: Comparison with DBC

In Rastegari *et al.*'s work [17], they also propose to learn a binary representation of images. They learn discriminate binary codes (DBCs) by formulating their goal as an optimization problem where the distances of examples in the same class are minimized and the distances of examples in different classes are maximized after binary representation. Their method is the state-of-the-art binary code technique on classification problems. To show the effectiveness of our approach, we conduct an independent experiment to make a direct comparison between ours and theirs.

The comparison is performed on the LFW View 1 data set. For both methods, View 1 training data serve as a training set where the projection to obtain a binary representation is learned, and View 1 test data serve as the test data for evaluation. The classification is performed by first computing the Hamming distance of each pair and then thresholding it to get a match/non-match answer. The threshold is the median value of all Hamming distances of test pairs. We plot the classification rate corresponding to various lengths of binary code in Figure 4. The feature we adopt in this figure is LBPr1.

From Figure 4 we can see that although DBC performs better at very short code lengths, our method achieves better performance when the code length becomes large. The superior performance at longer code lengths indicates that our technique can learn many discriminative bits, while DBC seems to saturate after learning relatively fewer bits. Also,

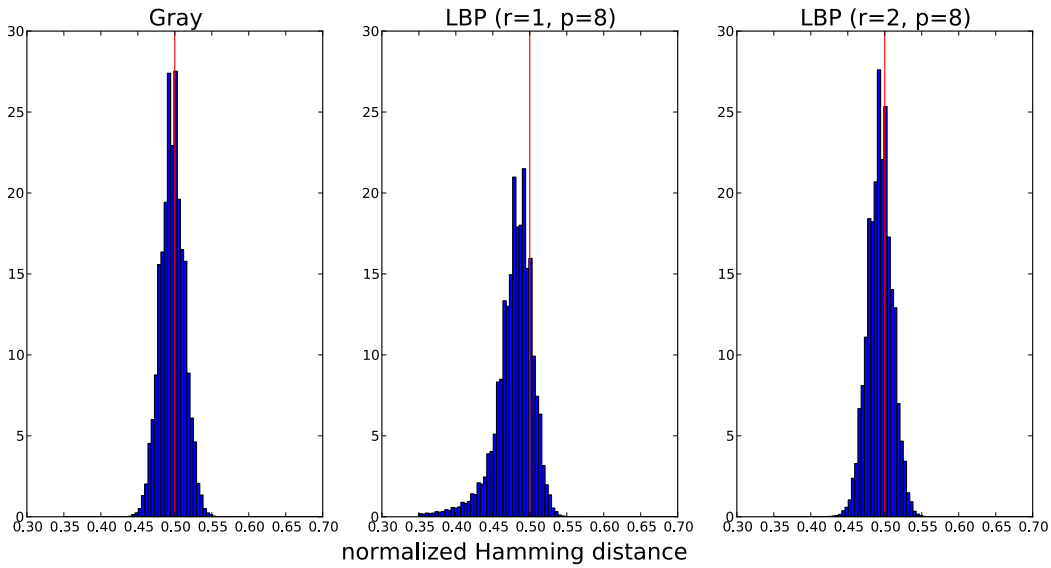


Figure 3. Normalized histogram for normalized Hamming Distance of classification result of all possible pairs of each feature.

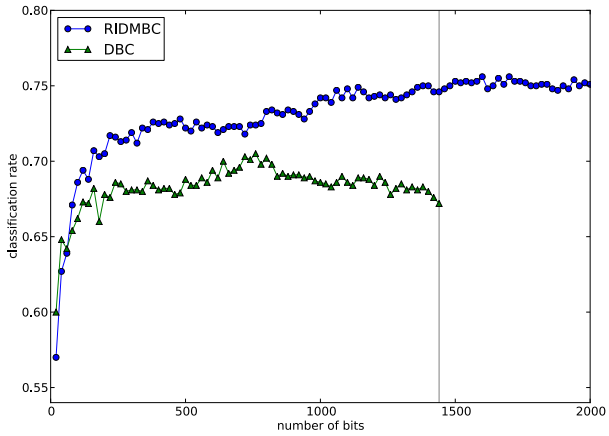


Figure 4. A direct comparison to DBC: Classification rate on View 1 test set with various code lengths.

note that the code length of DBC is bounded at 1440 bits for this data set (as shown by the gray vertical line in Figure 4). Our method, however, is able to learn many more bits. In general, performance is improved when the number of bits becomes larger.

4.4. Pilot Evaluation on LFW View 1 Data

We present the classification rate on LFW View 1 test data for each of the three kinds of features and for all three features combined. Details are as follows. For View 1 test data, Hamming distance between each image pair is computed. Since there is no other validation set to determine a threshold, we set the threshold to be the median value of all the Hamming distances given that a half of the test data are match pairs and the other half are non-match pairs. The performance of each of the three features and a combination of them is reported.

	Gray	LBPr1	LBPr2	Combined
RIDMBC	70.60	75.10	72.90	79.90
DBC	70.60	67.20	69.70	73.40

Table 1. Classification rate on LFW View 1 test data for RIDMBC and DBC

Table 1 shows the classification rate of RIDMBC and DBC on View 1 test data for each feature and the combined feature. Note that the number of bits for each feature is 2000 for RIDMBC. For DBC, the number of bits cannot exceed the dimension of the original feature space. So we set the dimension of the feature (the maximal number) as the number of bits DBC learns. In more detail, for DBC the number of bits for the gray, LBPr1 and LBPr2 features are 1344, 1440 and 1440 respectively. We can observe from the table that our approach (RIDMBC) outperforms DBC in most cases and that combining the three features performs better than any single feature.

5. Performance on PaSC

One possible concern about automatically trained binary codes is that they may become overly specific to a data set. After all, it is difficult to intuit what a single binary classifier is reacting to, yet alone what the encoding as a whole has learned. To evaluate generalization ability, we therefore evaluate the performance of RIDMBC on the chal-

PaSC, all images				
	Gray	LBPr1	LBPr2	Combined
RIDMBC	11.46	10.79	10.60	14.87
DBC	9.11	8.94	9.33	10.97

Table 2. Verification rate at FAR=0.01 on PaSC for RIDMBC and DBC: all images

PaSC, frontal images				
	Gray	LBPr1	LBPr2	Combined
RIDMBC	19.73	15.67	15.80	23.48
DBC	18.42	12.60	14.07	18.19

Table 3. Verification rate at FAR=0.01 on PaSC for RIDMBC and DBC: frontal images

lenging Point-and-Shoot Challenge dataset (PaSC) [2] after it has been trained on LFW. PaSC is a new, unrelated dataset of comparable or even greater difficulty than LFW. The learned linear SVM classifiers associated with each bit remain unchanged from LFW View 1 training set.

5.1. Point-and-Shoot Face Recognition Challenge

PaSC provides researchers with 3 different face recognition tasks based on comparing different data sources, namely, still images vs. still images, still images vs. videos and videos vs videos. We focus on the still image comparisons. PaSC contains 9376 images of 293 people for testing. The images are taken at nine locations (including indoors and outdoors) using five point-and-shoot still cameras. The pose of the subjects and their distance to the camera are largely varied. The still images are balanced with respect to distance to the camera, alternative sensors, frontal versus not-frontal views, and varying locations.

The protocol of PaSC differs from LFW. The test set has 4688 query images and 4688 target images. Image pairs are formed by taking one image from the query set and the other from the target set. Algorithms generate similarity scores for all image pairs in the test set. By looking at image pairs of interest, two ROCs can be plotted, one for frontal image pairs and one that combines both frontal and non-frontal images.

5.2. Performance on PaSC

Figure 5 presents the ROCs for RIDMBC along with those for two baseline algorithms provided by Colorado State University as part of the PaSC [2]. ROCs are shown for all still images and just the frontal images. Table 2 and Table 3 provide the associated verification rates at FAR=0.01 for all still images and just the frontal images, respectively. Note that verification rate $= 1 - FRR$. Our algorithm outperforms DBC in all cases. Table 4 compares our algorithm with two baseline methods supplied with PaSC, LRPCA [15] and CohortLDA [13]. Although trained on LFW View 1 training set, our method performs the best over all still images. It also performs well when only frontal images are included. The performance of CohortLDA is less robust: it is the worst when all images are considered, but the best when the evaluation is limited to only frontal images.

Method	all	frontal
RIDMBC	14.9	23.5
DBC	11.0	18.2
LRPCA	11.5	19.6
CohortLDA	9.0	31.7

Table 4. Verification rate at FAR=0.01 on PaSC

6. Conclusion

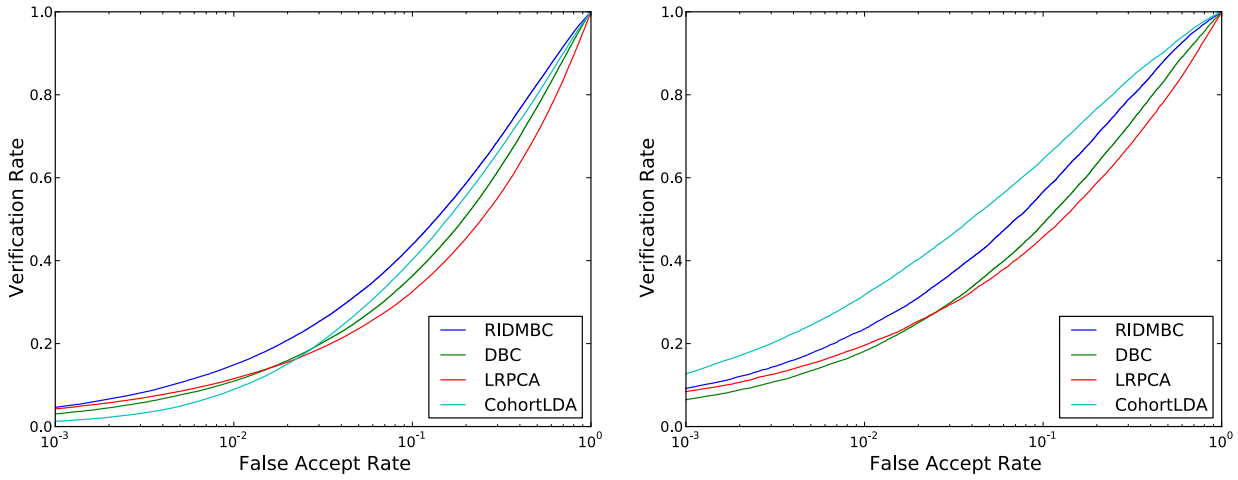
We propose RIDMBC, a new algorithm for learning binary codes. RIDMBC is similar to the prior state-of-the-art Predictable Discriminative Binary Codes algorithm in that both seek to minimize intraclass distance in Hamming space. However, the algorithms differ markedly in their treatment of interclass distances. The DBC algorithm explicitly seeks to maximize interclass distances, essentially optimizing Fisher's criterion. RIDMBC, in contrast, does not impose constraints on interclass distances. Instead, it relies upon randomized initial assignment of labels to arrive at largely uncorrelated binary features. Moreover, DBC has an upper bound on the number of bits it is able to learn while ours does not.

For face recognition, the RIDMBC algorithm is trained on the LFW dataset and performance results are presented on the PaSC dataset. As a point of comparison, the DBC algorithm is likewise trained on LFW and tested on PaSC. This experimental protocol aims to address the concern that the learned binary codes are overly specific to the dataset on which they are trained. In our experiments, the RIDMBC algorithm performs better than DBC for face recognition. The RIDMBC algorithm also outperforms two baseline algorithms provided as part of PaSC.

References

[1] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers and identity-preserving alignment for face verification. In *BMVC*, volume 1, page 5, 2012.

[2] J. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Given, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, et al. The challenge of face recognition from digital point-and-shoot cameras. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE, 2013.



(a) Entire challenge problem

(b) Frontal face images only

Figure 5. ROC curves of our method (RIDMBC) and two baselines (LRPCA and CohortLDA) on PaSC for the still challenge (a) Entire challenge problem (b) Frontal face images only

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] L. P. Coelho. Mahotas: Open source software for scriptable computer vision. *Journal of Open Research Software*, 1, July 2013.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3474–3481. IEEE, 2012.

[7] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011.

[8] K. Grauman and R. Fergus. Learning binary hash codes for large-scale image search. In *Machine Learning for Computer Vision*, pages 49–87. Springer, 2013.

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[10] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[11] N. Kumar, A. Berg, P. N. Belhumeur, and S. Nayar. Describable visual attributes for face verification and image search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):1962–1977, 2011.

[12] R.-S. Lin, D. A. Ross, and J. Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 848–854. IEEE, 2010.

[13] Y. M. Lui, D. Bolme, P. J. Phillips, J. R. Beveridge, and B. A. Draper. Preliminary studies on the good, the bad, and the ugly face recognition challenge problem. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 9–16. IEEE, 2012.

[14] D. C. Ngo, A. B. Teoh, and A. Goh. Biometric hash: high-confidence face recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(6):771–775, 2006.

[15] P. J. Phillips, J. R. Beveridge, B. A. Draper, G. Givens, A. J. O'Toole, D. S. Bolme, J. Dunlop, Y. M. Lui, H. Sahibzada, and S. Weimer. An introduction to the good, the bad, & the ugly face recognition challenge problem. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 346–353. IEEE, 2011.

[16] P. J. Phillips, W. T. Scruggs, A. J. O'Toole, P. J. Flynn, K. W. Bowyer, C. L. Schott, and M. Sharpe. Frvt 2006 and ice 2006 large-scale experimental results. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):831–846, 2010.

[17] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *Computer Vision–ECCV 2012*, pages 876–889. Springer, 2012.

[18] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, volume 9, page 6, 2008.

[19] R. P. Wildes. Iris recognition: an emerging biometric technology. *Proceedings of the IEEE*, 85(9):1348–1363, 1997.