Feature Selection in Face Recognition: A Sparse Representation Perspective

April 13, 2007

Abstract

In this paper, we examine the role of feature selection in face recognition from the perspective of sparse representation. We cast the recognition problem as finding a sparse representation of the test image features w.r.t. the training set. Using face recognition under varying illumination as an example, we show that if this inherent sparsity is properly harnessed, the choice of features is no longer critical. What is critical, however, is whether the number of features is sufficient and whether the sparse representation is correctly found. We propose a simple algorithm that classifies a test image based on a sparse representation of its features in terms of training image features. The representation is efficiently and effectively computed by \( \ell_1 \)-minimization. Our experiments show that imposing sparsity significantly increases the recognition rate for most conventional features. For example, using 120 Eigenfaces one can achieve 96.5% recognition rate on the entire Extended Yale B database. The differences between different features become insignificant as the feature-space dimension increases. Other unconventional features such as severely down-sampled images and completely random projections perform almost equally well. For example, the holistic features given by images down-sampled to just \( 12 \times 10 \) pixels achieve 92.4% recognition rate on the same database. All source codes in MATLAB will be made available for peer evaluation.

1 Introduction

Human faces are arguably the most extensively studied object in computer-based recognition. This is partly due to the remarkable face recognition capability of the human visual system [23], and also many important applications of computer-based face recognition. In addition, problems associated with face recognition are representative for object recognition in general. A central issue in the study of object recognition is the question of which features of an object are most important or informative for recognition. Due to the special geometric shape and appearance of the face, fixed filter banks (e.g., downsampling, Fourier, Gabor, wavelets) that are effective for analyzing stationary signals such as textures are believed unsuitable for face recognition. Thus, the dominant approach discards fixed filter banks in favor of features chosen adaptively based on the given images via techniques such as Eigenfaces [24], Fisherfaces [3], Laplacianfaces [15], and a host of variants [16, 18] (see Figure 1 for examples). The
features extracted using such filters are thought to be more relevant to face recognition, allowing reasonable recognition performance with simple, scalable classifiers such as nearest-neighbor (NN).

However, with so many proposed features, there is a lack of guidelines for practitioners to decide which features to use. Furthermore, it has been noticed that face recognition methods based on low-dimensional features cannot achieve satisfactory performance against human performance [19]. The emphasis on choosing the “best” features may have obscured important factors that could clarify the role of feature selection in the overall recognition problem. In this paper, using face recognition as an example, we reexamine feature selection within a new context and try to answer the following question:

To what extent does the selection of features matter, if the sparsity inherent in the recognition problem is properly harnessed?

To help the reader better understand the motivation behind this question, let us consider a recognition problem using $12 \times 10$ down-sampled images as an example shown in Figure 1. Our experiments will show that using such severely down-sampled images as features, the computer can achieve a recognition rate as high as $92.4\%$ in the Extended Yale B database (see Figure 3). When an illumination model is applied to compensate the lighting variation in the database, the recognition rate is further boosted to $98.6\%$. These performances arguably surpass humans’ ability to recognize down-sampled images of even familiar faces (humans require about $16 \times 16$ pixels [23]).

As we will show in this paper, this sudden performance boost comes from harnessing an important piece of information that is readily available in almost all object recognition problems but has not received much attention in computer vision. That is, to a large extent, object recognition seeks a sparse representation of the test image w.r.t. the training images of many objects:

Figure 1: (a). Original face image. (b). 120-D representations in terms of four different features (from top left): Eigenfaces, Laplacianfaces, down-sampled $12 \times 10$ image, and Randomfaces (see Section 2.5 for precise description). We will demonstrate that all these features give similarly good recognition performance.
The test image should ideally be interpreted only in terms of training images of the same object, a very small portion of the entire training set.

In other words, if we express the test image as a superposition of the training set, most of the weights should be zero. Hence the nonzero weights should be sparse! We caution the reader that the sparsity described here is not to be confused with the “sparse features” proposed in [20] for object detection. Recent progress in statistical signal processing has shown that, somewhat surprisingly, if the signal to be recovered is sparse, the choice of features is no longer critical. What is critical is whether the number of features is sufficient and whether the sparse solution is correctly found [6]. Furthermore, it has been shown that if the signal is sparse enough, it can be correctly and efficiently recovered via \( \ell^1 \)-minimization [8, 11].

**Contributions of this paper.** In this paper, we propose an extremely simple algorithm for face recognition, which uses \( \ell^1 \)-minimization to compute a sparse representation of (features of) the test image in terms of (features of) the training images. Our experimental results convincingly demonstrate the key role of properly enforcing and harnessing sparsity in object recognition. The algorithm achieves high recognition rates on the Extended Yale B database, significantly boosting the performance of popular face features such as Eigenfaces and Laplacianfaces (see Table 1). In addition, as stipulated by the theory of compressive sensing, the performances with different features converge as the number of features used increases. Then it should be no surprise at all that a similarly high recognition rate can be achieved using the same number of down-sampled pixels or completely random features, which we call Randomfaces (see Figure 1 for an example).

**What we do not do.** Although our problem formulation uses the fact that images of a face under varying illumination lie in a subspace (the harmonic plane [2]), we do not utilize domain-specific information about the structure of this subspace as in [2, 14, 17, 22]. By minimizing the use of domain-specific knowledge and preprocessing, we can more fairly compare the effectiveness of different features. In addition, it makes our conclusions about feature selection also applicable to other object recognition problems where a linear feature model is valid.

Nevertheless, if desired, the proposed algorithm can easily incorporate other domain-specific information about face recognition. For instance, knowing faces are approximately Lambertian surfaces, one can preprocess the face images to reduce the effect of varying illumination, say by using the self-quotient images [25]. Our experiments show that, as expected, this can further improve the recognition rate with the same features, although after imposing sparsity there is not much room left for improvement (comparing Table 4 with Table 1).

\[ ^1 \text{In object recognition, this is typically guaranteed, since the sparsity increases as the number of object classes increases.} \]
2 Problem Formulation and Solution

2.1 Recognition as a Linear Sparse Representation

In this paper, we consider face recognition with frontal images that have been properly cropped and normalized. Suppose \( n \) training images are given to the computer for each of \( k \) subjects. Each image is of the size \( w \times h \) and can be viewed as a point in the space \( \mathbb{R}^D \) with \( D = w \times h \). It has been shown in the literature [2, 17] that under varying illumination, the images of the same face span an (approximately) nine-dimensional subspace in \( \mathbb{R}^D \), called a face subspace.

More precisely, let us stack the \( n \) images associated with subject \( i \) as \( n \) vectors \( v_{i,1}, v_{i,2}, \ldots, v_{i,n} \in \mathbb{R}^D \) and suppose these vectors are sufficient to span the face subspace. Then any new test image of the same subject, stacked as a vector \( y \in \mathbb{R}^D \), can be represented as a linear superposition of the training examples associated with subject \( i \):

\[
y = a_{i,1}v_{i,1} + a_{i,2}v_{i,2} + \cdots + a_{i,n}v_{i,n},
\]

where \( a_{i,j} \) are all scalars in \( \mathbb{R} \).

Since the identity of the test image \( y \) is assumed unknown, it should be represented in terms of all \( N = k \times n \) training images in the training set. Collect all the training images as column vectors of one matrix:

\[
A = [v_{1,1}, v_{1,2}, \ldots, v_{1,n}, v_{2,1}, \ldots, v_{k,n}] \in \mathbb{R}^{D \times N}.
\]

Then ideally the test image \( y \) of subject \( i \) can be represented as

\[
y = Ax_0 \in \mathbb{R}^D,
\]

where \( x_0 = [0, \cdots, 0, a_{i,1}, a_{i,2}, \ldots, a_{i,n}, 0, \ldots, 0]^T \in \mathbb{R}^N \), whose entries are mostly zeros except those associated with the \( i \)th subject.

As the entries of the vector \( x_0 \) encode the identity of the test image \( y \), it is then tempting to solve it from the system of linear equations (3). If the system is under-determined \((D < N)\) for \( A \in \mathbb{R}^{D \times N} \), the solution is not unique and traditionally a solution is chosen with minimum \( \ell^2 \)-norm:

\[
(P_2) \quad \min \|x\|_2 \quad \text{subject to} \quad y = Ax.
\]

Similarly, if the system is over-determined \((D > N)\), one often seeks the least-squares solution that minimizes \( \|y - Ax\|_2 \).

However, at least two major difficulties make both approaches impractical and ineffective for face recognition:

1. The data is very high-dimensional. For instance, for a \( 640 \times 480 \) grayscale image, the dimension \( D \) is on the order of \( 10^5 \), while we normally have only a few samples per subject. Such small sample size only exacerbates the “curse of dimensionality” that plagues high-dimensional statistics [10]. Aside from the computational cost of solving such large systems of equations, the least-squares (or minimum \( \ell^2 \)-norm) solution can exhibit severe bias if the system is not properly regularized [4].
2. The desired solution is sparse. The ratio of the nonzero entries in $x_0$ is only \( \frac{n}{N} = \frac{1}{k} \): For instance, if $k = 20$, only $5\%$ of the entries of $x_0$ should be nonzero. The more sparse the recovered $x$ is, the easier it will be to accurately determine the identity of the test image $y$. Unfortunately, the minimum \( \ell^2 \)-norm solution of the equation $y = Ax$ is generally non-sparse, and can be very far from the true sparse solution in (3) when the system is under-determined or there is a large error in $y$ [7, 9, 13].

2.2 Sparse Solution in a Reduced Dimension

To tackle the above difficulties, we seek methods that

1. reduce the data dimension $D$ to $d \ll D$ and

2. explicitly compute sparse representations of $y$ in the lower-dimensional space.

We will see that these two goals are complementary: Appropriately enforcing sparsity renders the outcome less dependent on the details of dimension reduction.

In the computer vision literature, numerous dimension reduction methods have been investigated to project high-dimensional face images to low-dimensional feature spaces. One class of methods extracts holistic face features, such as Eigenfaces [24], Fisherfaces [3], and Laplacianfaces [15]. Another class of methods tries to extract significant partial facial features (e.g., eye corners) [21, 23]. For such face features, the projection from the image space to the feature space can be represented as a matrix $R \in \mathbb{R}^{d \times D}$ with $d \ll D$. Applying $R$ to both sides of equation (3) yields:

$$\tilde{y} = Ry = RAx_0 \in \mathbb{R}^d. \quad (5)$$

After projection, the dimension $d$ of the feature space usually becomes smaller than $N$. Hence, the system of equations (5) is under-determined, and the solution $x$ is not unique. Nevertheless, the desired $x_0$ should still be sparse. Under very mild conditions on $\tilde{A} = RA$, the sparsest solution to the system of equations is indeed unique [13]. In other words, the desired $x_0$ is the unique solution to the following optimization problem:

$$(P_0) \quad \min \| x \|_0 \quad \text{subject to} \quad \tilde{y} = \tilde{A}x,$$  \quad (6)

where $\| \cdot \|_0$ denotes the $\ell^0$ norm, which simply counts the number of nonzero entries in a vector. Directly solving $(P_0)$ is an NP-hard problem [1]: In the most general case, the sparsest solution is found by exhausting all subsets of the entries for $x$.

2.3 Sparse Solution via $\ell^1$-Minimization

A recent breakthrough in statistical signal processing [7, 9, 13] reveals that if the solution $x_0$ sought is sparse enough, the combinatorial problem $(P_0)$ is equivalent to the following $\ell^1$-minimization problem:

$$(P_1) \quad \min \| x \|_1 \quad \text{subject to} \quad \tilde{y} = \tilde{A}x.$$

---

\(^2\)One may also consider features computed via a nonlinear map $\Phi$, e.g., using kernel methods [19, 26]. The proposed framework also applies to kernel-based features if the object classes are linearly separable in the kernel space: $\tilde{y} = \Phi(y) = \Phi(A)x_0$. 

This problem can be solved in polynomial time by standard linear programming or convex optimization methods.

Figure 2 gives a geometric interpretation (essentially due to [12]) of why minimizing the $\ell^1$ norm recovers sparse solutions. Let $B_\alpha$ denote the $\ell^1$-ball of radius $\alpha$:

$$B_\alpha = \{x : \|x\|_1 \leq \alpha\} \subset \mathbb{R}^N.$$ (8)

In Figure 2, the unit $\ell^1$-ball $B_1$ is mapped to the polytope $P = \tilde{A} \cdot B_1 \subset \mathbb{R}^d$ consisting of all $\tilde{y}$ that satisfy $\tilde{y} = \tilde{A}x$ for some $x$ whose $\ell^1$-norm is $\leq 1$.

The geometric relationship between $B_\alpha$ and the polytope $\tilde{A} \cdot B_\alpha$ is invariant to scaling. That is, if we scale $B_\alpha$, its image under $\tilde{A}$ is also scaled by the same amount. Geometrically, finding the minimum $\ell^1$-norm solution $x_1$ to $(P_1)$ is equivalent to expanding the $\ell^1$-ball $B_\alpha$ until the polytope $\tilde{A} \cdot B_\alpha$ first touches $\tilde{y} = \tilde{A}x_0$. The value of $\alpha$ at which this occurs is exactly $\|x_1\|_1$.

Now suppose that $y = \tilde{A}x_0$ for some sparse $x_0$. We wish to know when solving $(P_1)$ correctly recovers $x_0$. This question is easily resolved from the geometry of Figure 2: Since $x_1$ is found by expanding both $B_\alpha$ and $P = \tilde{A} \cdot B_\alpha$ until a point of $P$ touches $\tilde{y}$, the $\ell^1$-minimizer $x_1$ must generate a point $\tilde{A}x_0$ on the boundary of $P$.

Thus $x_1 = x_0$ if and only if the point $\tilde{A}(x_0/\|x_0\|_1)$ lies on the boundary of $P$. For the example shown in Figure 2, it is easy to see that $\ell^1$ minimization recovers all $x_0$ with only one nonzero entry. This equivalence holds because all of the vertices of $B_1$ map to points on the boundary of $P$.

If $\tilde{A}$ maps all $k$-dimensional faces of $B_1$ to faces of $P$, the polytope $P$ is referred to as (centrally) $k$-neighborly [12]. From the above, we see that $(P_1)$ recovers all $x_0$ with $\leq k + 1$ nonzeros iff $P$ is $k$-neighborly. This condition is surprisingly common: the results of [6] show that even random matrices (e.g., uniform, Gaussian, and partial Fourier) are highly neighborly and therefore admit sparse solution by $\ell^1$-minimization.

Unfortunately, there is no known algorithm for efficiently verifying the neighborliness of a given, fixed $\tilde{A}$. However, our experimental results verify the ability of $\ell^1$-minimization to recover sparse representations in several commonly-used feature spaces. This suggests that the data-dependent feature matrices popular in face recognition may also be highly neighborly.
Since real images are noisy, it may not be possible to express the (features of) the test image exactly as a sparse superposition of (features of) the training images. To model noise and error in the data, one can consider a stable version of (5) that includes a noise term with bounded energy $\|z\|_2 < \epsilon$:

$$\tilde{y} = \tilde{A}x + z \in \mathbb{R}^d. \tag{9}$$

It has been shown in [11] that in this case the sparse solution can be approximately found via the following program:

$$\begin{align*}
(P_1') \quad \min \|x\|_1 \quad \text{subject to} \quad \|\tilde{y} - \tilde{A}x\|_2 \leq \epsilon.
\end{align*} \tag{10}$$

This program can be efficiently solved via convex optimization [5] (see Section 3 for our algorithm of choice).

### 2.4 Classification from Sparse Coefficients

Ideally, the nonzero entries in the estimate $x$ will all be associated with the columns in $\tilde{A}$ from a single subject, and we can easily assign the test image $y$ to that subject. However, due to noise, the nonzero entries may be associated with multiple subjects (see Figure 3). Many classifiers can resolve this problem. For instance, we can assign $y$ to the subject with the most non-zero entries in $x$ (majority vote); or we can assign $y$ to the subject with the single largest entry of $x$. However, these heuristic classifiers do not harness the subspace structure associated with face images. To better harness this structure, we instead classify $y$ based on how well the coefficients associated with training images of each subject reproduce $y$.

For each subject $i$, define a function $\rho_i : \mathbb{R}^N \to \mathbb{R}^N$ which selects the coefficients associated with the $i$-th subject. For $x = (x_j) \in \mathbb{R}^N$, $\rho_i(x) \in \mathbb{R}^N$ is a new vector whose only nonzero entries are the entries in $x$ that are associated with subject $i$, and whose entries associated with all other subjects are zero. We then set

$$\text{identity}(y) = \arg \min_i \|\tilde{y} - \tilde{A}\rho_i(x)\|_2, \tag{11}$$

that is, we assign $y$ to the subject whose associated coefficients, $\rho_i(x)$, give the best approximation to $y$. This simple classifier produces all the results in Section 3.

Algorithm 1 below summarizes the complete recognition procedure.

**Example 1 (Down-sampled Images)** To illustrate how Algorithm 1 works, we randomly select half of the 2,414 images in the Extended Yale B database as the training set, and the rest for testing. In this example, we choose $R$ to simply be the down-sampling filter that sub-samples the images to the size of $12 \times 10$. The pixel values of the down-sampled image are used as features, and hence the feature space dimension is $d = 120$.

Figure 3 illustrates the sparse coefficients recovered by Algorithm 1 for a test image from Subject 1. The figure also shows the features and original images that correspond to the two largest coefficients. As we see, the two largest coefficients are all associated with the training samples from Subject 1. In Section 3, we will show that the
Algorithm 1 (Recognition via $\ell^1$-Minimization)

1: **Input:** a matrix of training images $A \in \mathbb{R}^{D \times N}$ for $k$ subjects, a linear feature transform $R \in \mathbb{R}^{d \times D}$, a test image $y \in \mathbb{R}^D$, and an error tolerance $\epsilon$. 
2: Compute the features $\tilde{y} = R y$ and $\tilde{A} = RA$. 
3: Solve the convex optimization problem ($P'_1$):
   \[
   \min \|x\|_1 \text{ subject to } \|\tilde{y} - \tilde{A}x\|_2 \leq \epsilon.
   \]
4: Compute identity $(y) = \arg \min_i \|\tilde{y} - \tilde{A}_i(x)\|_2$. 
5: **Output:** identity $(y)$.

**Figure 3:** Recognition with down-sampled images as features. The test image belongs to Subject 1. The values of the sparse coefficients recovered from Algorithm 1 are plotted in the right together with the two training examples that correspond to the two largest coefficients.

The total recognition rate on the Extended Yale B database is 92.4\% using the $12 \times 10$ downsample images.

For comparison, we compute the coefficients given by the conventional $\ell^2$-minimization (4) and the $\ell^2$ distances between the test image and the training images that are often used for the NN algorithm. The coefficients and the distances are shown in Figure 4. Neither the largest coefficients nor the smallest distances are associated with Subject 1. As we will see in Section 3, this inevitably leads to inferior recognition performance (only 61.81\%).

### 2.5 Feature Selection for $\ell^1$-Minimization

With this algorithm in place, the remaining question is how the choice of the feature transform $R$ affects its recognition performance.

Obviously $R$ affects the performance through the matrix $\tilde{A} = RA$’s ability to recover sparse solutions via $\ell^1$-minimization. When the number of non-zero entries in

---

3 Other commonly used distance metrics in NN such as $\ell^1$ distance give similar results as Figure 4 bottom.
Figure 4: Coefficients from \(\ell^2\)-minimization (top) and \(\ell^2\) distances between the test image and the training images (bottom).

As \(x\) increases past a critical value, \(\ell^1\)-minimization no longer always finds the correct, sparse \(x_0\). This value is called the breakdown point of \(\tilde{A}\) [13]. Different \(\tilde{A}\) have different breakdown points. However, there is no known closed-form formula nor polynomial-time algorithm to determine the breakdown point of a given matrix. Thus, in Section 3, we choose to measure and compare experimentally the improvement in recognition rate due to \(\ell^1\)-minimization for popular features such as Eigenfaces, Fisherfaces, and Laplacianfaces.

For extant face recognition methods, it is known that increasing the dimension of the feature space generally improves the recognition rate, as long as the feature distribution does not become degenerate [19]. However, degeneracy is no longer an issue for our algorithm since \(\ell^1\)-minimization properly regularizes linear regression [4]. We can use very high-dimensional features without any concern about degeneracy. In addition, it is easy to show that if \(d\) increases, so does the breakdown point of \(\tilde{A}\) [6, 9]. Therefore, as we will demonstrate in Section 3, the performance of our algorithm improves gracefully when \(d\) increases. As the optimization problem \((P_1)\) or \((P'_1)\) can be efficiently solved by linear programming or convex optimization, we are able to experiment with features of dimension over \(d = 16,000\).4

Algorithm 1’s ability to handle high-dimensional features allows us to observe an important phenomenon in feature selection, that is unique to the \(\ell^1\) framework. Theoretical results of [6, 9] have shown that if the signal \(x\) is sparse, then with overwhelming probability, it can be correctly recovered via \(\ell^1\)-minimization from any sufficiently large dimension \(d\) of linear measurements \(\tilde{y}\). This surprising phenomenon has been dubbed as the “blessing of dimensionality” [6, 10]. Thus, one should expect to see similar recognition performance from Algorithm 1 even with randomly selected facial features:

**Definition 1 (Randomfaces)** Consider a transform matrix \(R \in \mathbb{R}^{d \times D}\) whose entries are independently sampled from a zero-mean normal distribution. The row vectors of \(R\) can be viewed as \(d\) random faces in \(\mathbb{R}^D\).

---

4This is still constrained by the memory limit of MATLAB, but not the complexity of the algorithm.
The somewhat extreme and controversial assertion that Randomfaces can be as effective as conventional features will be verified by our experiments in the next section.

3 Experiments

In this section, we verify the performance of the proposed face recognition algorithm using the Extended Yale B database [14]. The database consists of 2,414 valid frontal-face images of 38 individuals. The cropped and normalized $192 \times 168$ face images were captured under various laboratory-controlled lighting conditions. To fairly compare with the performance of other recognition algorithms, we randomly select half of the images of each subject for training, and the other half for testing. The reason for randomly choosing the training set is to make sure that our results and conclusions will not depend on any special choice of the training data. The $\ell^1$ optimization in our algorithm is based on the “$\ell^1$-magic” MATLAB toolbox at: http://www.acm.caltech.edu/l1magic/. In all the experiments, the error distortion $\epsilon = 0.18$, which is also estimated using the toolbox. We will release the code of our face recognition algorithm on our website after the review.

3.1 Boosting Performance of X-face Features

We first test our algorithm using several conventional holistic face features, namely, Eigenfaces, Laplacianfaces, and Fisherfaces. We compare their performances with two other features that seem to contradict conventional wisdom: Randomfaces and down-sampled images. Whenever possible, we compute the recognition rate with the feature space dimension being 16, 30, 56, 120, and 504, respectively. Those numbers correspond to the dimensions of the down-sampled image with the ratios 1/40, 1/32, 1/24, 1/16, and 1/8, respectively. The MATLAB implementation of our algorithm only takes a few seconds to classify one test image on a typical 3G Hz PC.

Table 1 shows the recognition rates of all the X-face features using Algorithm 1. In comparison, Table 2 shows the recognition rates using the nearest neighbor (NN) classifier. Notice that Fisherfaces are different from the others features since the maximal number of valid Fisherfaces is one less than the number of classes $k$ [3], which is 38 in our case.

Comparing results in Table 1 and Table 2, we draw the following conclusions:

1. By imposing sparsity via $\ell^1$-minimization, Algorithm 1 significantly boosts the performances of all the features except for Fisherfaces. When the dimensions of the feature spaces are higher than 56, all features achieve recognition rates near or above 90%. The recognition rates of the conventional features using NN shown in Table 2 are similar to what have been reported in the literature, although some reported on different databases or with different training subsets.

---

5 We cut off the dimension at 504 as the implementation of Eigenfaces and Laplacianfaces reaches the memory limit of MATLAB. Nevertheless, 504 is already sufficient to verify our conclusions.

6 One may further improve the speed of the recognition algorithm by reformulating the optimization using linear program (P1) described in Section 2.3, which does not explicitly model the noise. The implementation of (P1) is significantly faster than (P1’), yet their performances are quite close.
Table 1: Performance of Algorithm 1 with X-face features.

<table>
<thead>
<tr>
<th>Dimension (d)</th>
<th>16</th>
<th>30</th>
<th>56</th>
<th>120</th>
<th>504</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen [%]</td>
<td>61.97</td>
<td>84.67</td>
<td>92.87</td>
<td>96.5</td>
<td>98.8</td>
</tr>
<tr>
<td>Laplacian [%]</td>
<td>79.96</td>
<td>82.93</td>
<td>91.72</td>
<td>96.2</td>
<td>98</td>
</tr>
<tr>
<td>Random [%]</td>
<td>58.74</td>
<td>81.94</td>
<td>90.89</td>
<td>94.4</td>
<td>96.6</td>
</tr>
<tr>
<td>Downsample [%]</td>
<td>54.35</td>
<td>76.22</td>
<td>87.49</td>
<td>92.4</td>
<td>96.9</td>
</tr>
<tr>
<td>Fisher [%]</td>
<td>51.53</td>
<td>85.92</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 2: Performance of NN with X-face features.

<table>
<thead>
<tr>
<th>Dimension (d)</th>
<th>16</th>
<th>30</th>
<th>56</th>
<th>120</th>
<th>504</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen [%]</td>
<td>58.08</td>
<td>72.78</td>
<td>79.78</td>
<td>83.93</td>
<td>85.75</td>
</tr>
<tr>
<td>Laplacian [%]</td>
<td>62.88</td>
<td>75.64</td>
<td>81.28</td>
<td>85.17</td>
<td>87.74</td>
</tr>
<tr>
<td>Random [%]</td>
<td>52.36</td>
<td>61.06</td>
<td>66.53</td>
<td>67.85</td>
<td>66.36</td>
</tr>
<tr>
<td>Downsample [%]</td>
<td>45.9</td>
<td>46.73</td>
<td>54.68</td>
<td>61.81</td>
<td>65.37</td>
</tr>
<tr>
<td>Fisher [%]</td>
<td>78.54</td>
<td>87.74</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

For example, [15] reported the best recognition rate of 75% using Eigenfaces at 33 dimension, and 89% using Laplacianfaces at 28 dimension on the Yale face database.

2. Increasing the dimension of the feature space allows all features to achieve almost equally good performance in our framework. In particular, at dimension 504, Randomfaces and down-sampled images both achieve recognition rates exceeding 96%, only 2% worse than conventional features. On the other hand, they significantly underperform in Table 2 using NN. The results corroborate predictions from the theory of sparse representation and ℓ1-minimization discussed in Section 2.5.

Notice that above conclusions do not diminish the significance of previous studies on the conventional face features, since these studies are mostly limited to the scenario where the feature space dimension is low (usually less than 50). In that case, Table 2 confirms that the choice of feature does make some difference, as Fisherfaces clearly outperform the other features considered. However, Table 1 shows that sparsity is arguably an equally, if not more, important source of information for achieving high-performance recognition – it allows us to harness additional information encoded in any high-dimensional features and to achieve rates far beyond the capability of Fisherfaces. In contrast, classical classifiers (e.g., NN) are unlikely to produce good results in very high-dimensional feature spaces with only limited training samples per class. In the next subsection, we will further demonstrate the performance of our algorithm using other even higher-dimensional features.
3.2 Partial Face Features

As a second set of experiments, we test the proposed algorithm using the following two partial face features:

1. **Half faces**: We use the left half of a face image as a partial face feature (shown in Table 3), which corresponds to the right half face of the subject.

2. **Right eyes**: It is known in the study of human vision that the region around the eyes is one of the most informative features for face recognition [21, 23]. We extract $60 \times 84$ right-eye regions from the face images as another type of partial face feature.

![Example of right-eye (RE) feature and half-face (HF) feature.]

<table>
<thead>
<tr>
<th>Features</th>
<th>RE</th>
<th>HF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension ($d$)</td>
<td>5,040</td>
<td>16,128</td>
</tr>
<tr>
<td>Algorithm 1 [%]</td>
<td>93.5</td>
<td>98.4</td>
</tr>
<tr>
<td>NN [%]</td>
<td>60.73</td>
<td>65.12</td>
</tr>
</tbody>
</table>

Notice that the dimension $d$ of either feature is larger than the number of training samples ($N = 1, 207$), and the linear system (5) to be solved becomes over-determined. Nevertheless, we simply apply the same Algorithm 1. The results in Table 3 again show that $\ell^1$-minimization achieves significantly better recognition rates than NN.

3.3 Illumination Effects

In all the above experiments, we did not use the domain-specific knowledge that faces are approximately Lambertian surfaces and the variability in the training/test images is caused by varying illumination. It is well known in the vision literature that if this information is allowed to be used in the training stage and the illumination effect can be reduced, the problem becomes much simpler and recognition rates can be significantly improved for the conventional features. For instance, we choose to preprocess all the images in the Extended Yale B database using the *self-quotient image* (SQI) technique [25] (see Figure 5 for an example), which to some extent reduces the illumination effect.\(^7\) Table 4 shows the recognition rate when Algorithm 1 is applied to features computed using the SQIs. Table 5 shows the recognition rate with NN. Notice that preprocessing with SQI significantly improves the recognition performance of Algorithm 1 and NN comparing to Table 1 and 2.

Comparing the rates with Algorithm 1 in Table 4 to the rates for NN in Table 5, we see that the conclusions given in Section 3.1 still hold: Sparsity improves performance when the dimension of the feature space is high ($\geq 56$); and the performances of all features converge as the feature dimension increases. However, comparing to the

\(^7\)SQI is an unsupervised algorithm that is known to be sensitive to image noise. We expect better recognition results can be achieved by using other more accurate illumination models [2, 14, 17, 22].
Figure 5: Left: Original. Right: Self-quotient image (SQI).

Table 4: Performance of Algorithm 1 with SQI X-face features.

<table>
<thead>
<tr>
<th>Dimension (d)</th>
<th>16</th>
<th>30</th>
<th>56</th>
<th>120</th>
<th>504</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen [%]</td>
<td>95.77</td>
<td>99.09</td>
<td>99.34</td>
<td>99.42</td>
<td>99.42</td>
</tr>
<tr>
<td>Laplacian [%]</td>
<td>95.78</td>
<td>99.09</td>
<td>99.34</td>
<td>99.42</td>
<td>99.42</td>
</tr>
<tr>
<td>Random [%]</td>
<td>52.44</td>
<td>75.39</td>
<td>89.64</td>
<td>98.6</td>
<td>99.26</td>
</tr>
<tr>
<td>Downsampling [%]</td>
<td>55.92</td>
<td>73.82</td>
<td>89.64</td>
<td>98.6</td>
<td>99.17</td>
</tr>
<tr>
<td>Fisher [%]</td>
<td>95.03</td>
<td>98.76</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

rates in Table 1, when the feature dimension is high enough, the room left for further improvement by reducing the illumination effect is not large.

4 Conclusion and Discussion

In this paper, we have argued both theoretically and experimentally that exploiting sparsity is critical for high-performance object recognition. With sparsity properly harnessed, the choice of features becomes less important than the number of features used (in our face recognition example, approximately 50 are sufficient). Furthermore, when the number of features are large enough (in our example, approximately 500), even randomly generated features or severely down-sampled images are just as good as conventional features.

Complementary to this paper, in a parallel submission to ICCV’07 (attached as supplementary), we show that sparsity also plays a crucial role in face recognition when there is severe corruption in the test images. With up to 30% occlusion, highly accurate recognition can still be achieved without any dimension reduction, feature selection, or other preprocessing.

The conclusions drawn in this paper apply to any object recognition problems where the linear feature model (5) is valid. For face recognition, this model applies to varying illumination and expression. However, for recognition with pose variation, the linear model may no longer be accurate. Most successful solutions to this problem have been focused on kernel methods that convert nonlinear face structures to linearly separable subspaces [19,21,26]. Yet practitioners are faced with similar overfitting and singularity problems in even higher-dimensional kernel spaces with only limited training samples. We believe the proposed classification framework via $\ell^1$ minimization may also provide new solutions for kernel-based face features.
Table 5: Performance of NN with SQI X-face features.

<table>
<thead>
<tr>
<th>Dimension (d)</th>
<th>16</th>
<th>30</th>
<th>56</th>
<th>120</th>
<th>504</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen [%]</td>
<td>91.63</td>
<td>97.93</td>
<td>98.92</td>
<td>99.17</td>
<td>98.84</td>
</tr>
<tr>
<td>Laplacian [%]</td>
<td>96.13</td>
<td>98.43</td>
<td>99.09</td>
<td>99.42</td>
<td>99.34</td>
</tr>
<tr>
<td>Random [%]</td>
<td>54.87</td>
<td>76.14</td>
<td>91.38</td>
<td>98.18</td>
<td>99.17</td>
</tr>
<tr>
<td>Downsample [%]</td>
<td>59.9</td>
<td>73.99</td>
<td>88.73</td>
<td>96.27</td>
<td>99.17</td>
</tr>
<tr>
<td>Fisher [%]</td>
<td>98.92</td>
<td>98.84</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

References


