Robust Face Recognition via Sparse Representation

Chuxiangbo Wang

December 10, 2023

Contents

1 Background Introduction

1.1 Building Training Set

Training samples are crucial in constructing a robust face recognition model. A good, efficient training set can greatly increase the face recognition accuracy. Here, we define the training set for the i-th object class (i-th person) as $A_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,n}] \in \mathbb{R}^{m \times n}$. where each $v_{i,j}$ represents one image of the i-th object class, the cloumn vector $v_{i,j}$ is formed by vertically stack the matrix that reprensented the j-th image of i-th object class. This approach aligns with the concept of sparse signal representation, as discussed in Wright et al.'s paper, which emphasizes the importance of a large feature set for effective face recognition.

Figure 1: Samples from one of the Object Classes

1.2 Construct the Overall Training Samples for All Classes

The overall training set for all k object classes is constructed as a matrix $A = [A_1, A_2, \ldots, A_k]$. Each of these matrices A_i contains training samples for the i-th class as we discussed above.

1.3 Test Samples and Their Representation

For a test sample $y \in \mathbb{R}^m$ belonging to the same i-th class, it lies in the linear span of A_i. This is represented as $y = \gamma_{i,1}v_{i,1} + \gamma_{i,2}v_{i,2} + \ldots + \gamma_{i,n}v_{i,n}$ for some scalars $\gamma_{i,j} \in \mathbb{R}$. And the linear representation of a test sample y is expressed as $y = Ax_0$, where $x_0 = [0, \ldots, 0, \gamma_{i,1}, \gamma_{i,2}, \ldots, \gamma_{i,n}, 0, \ldots, 0]^T$. This linear representation aligns with the sparse representation approach, where test samples are represented as sparse linear combinations of the training samples. Note that the test sample is classified under the fourth object class, indicating it originates from the same individual; however, it is essential to ensure this test image is not part of the training set, as including it would lead to a trivial and uninformative result

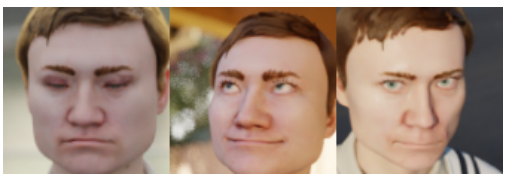


Figure 2: Test Samples from the Same Object Class

2 ℓ_2 and ℓ_0 Minimization

Recalling our discussion from Chapter 1, we understand that a test sample, denoted as y, can be mathematically expressed in the form $y = Ax_0$. In this expression, A represents the overall training set matrix, and x_0 is the coefficient vector we aim to find. It is a fundamental concept in linear algebra that if the system described by $y = Ax_0$ is over-determined, which typically means there are more equations than unknowns, a unique solution for x_0 generally exists. This scenario, however, is less common in the context of robust face recognition.

In the realm of face recognition, we often encounter the opposite situation, where the system is under-determined. This means there are fewer equations than unknowns, leading to a multitude of possible solutions for x_0, rather than a single, unique solution.

Given this problem, our current focus shifts to developing and implementing a method that can effectively solve the under-determined system $y = Ax_0$. This involves seeking a solution that not only satisfies the equation but also aligns with our specific application on face recognition. Our goal is to find solution that is most suitable and reliable for identifying or classifying the test sample y in the context of face recognition.

2.1 ℓ_2-Minimization

The most straightforward approach to solving the system $Ax = y$ is by employing ℓ_2-minimization. This method is mathematically represented as follows:

$$\hat{x}_2 = \arg\min_x \|\mathbf{x}\|_2, \quad \text{subject to} \quad A\mathbf{x} = \mathbf{y}.$$

In this formulation, the goal is to find the vector $\mathbf{x}$ that minimizes the ℓ_2-norm. The condition $A\mathbf{x} = \mathbf{y}$ ensures that the solution adheres to the constraints of our system.

However, it's crucial to note that while ℓ_2-minimization is conceptually simple and computationally feasible (we can just apply the Pseudo-Inverse of A), it often leads to solutions that are very dense. This means that most of the elements of the solution vector $\mathbf{x}$ are non-zero, or even have very large values accross the entries. In the context of face recognition, a dense solution can be less informative. This is because sparsity in the solution is often desirable; it allows us to identify which specific features or components are most significant in representing the test sample $\mathbf{y}$. Therefore, despite its simplicity, the ℓ_2-minimization approach may not always be the most effective method, especially when a more sparse and informative solution is required for better interpretation and analysis.

2.2 ℓ_0-Minimization

Building upon our discussion of the ℓ_2-minimization method, it motivate us to explore alternative approaches that might address its limitations, which is the sparsity. The most straightforward alternative is the ℓ_0-minimization, which leads us to the question:

$$\hat{x}_0 = \arg\min_x \|\mathbf{x}\|_0, \quad \text{subject to} \quad A\mathbf{x} = \mathbf{y}.$$

In this context, the ℓ_0-norm of a vector $\mathbf{x}$ refers to the number of non-zero elements in $\mathbf{x}$. Therefore, minimizing the ℓ_0-norm means finding the most sparse solution.

A sparse solution can lead to more meaningful results, as it highlights which specific aspects of the training set contribute most significantly to representing a given test sample $\mathbf{y}$.

However, despite its advantages in terms of sparsity, ℓ_0-minimization presents significant computational challenges. The process of finding the sparsest solution is typically NP-Hard, there is

no known polynomial-time algorithm to solve it efficiently. This computational complexity makes the ℓ_0-minimization method less practical, especially for large-scale problems. Therefore, while the pursuit of sparsity is desirable, we need to consider a suitable method for solving the problem. In the later numerical experiment section, we will use the greedy method: Orthogonal Matching Pursuit (OMP) to perform the ℓ_0-minimization.

Given the computational challenges associated with the ℓ_0-minimization, we want to seek alternative methods that can approximate its effectiveness in achieving sparsity while being less computational heavy. This brings us to the concept of ℓ_1-minimization, an alternative that often serves as a practical substitute for ℓ_0-minimization under certain conditions.

3 Sparse Solution via ℓ_1 Minimization

In the study of compressive sensing, we know that the solution to ℓ_0-minimization can be equivalent to that of ℓ_1-minimization under center conditions (sparsity of x, coherence of A, etc.). Thus, the ℓ_1 minimization of our problem can be formulated as:

$$\hat{x}_1 = \arg\min_x \|\mathbf{x}\|_1, \quad \text{subject to} \quad A\mathbf{x} = \mathbf{y}.$$

Here, ℓ_1-minimization aims to minimize the ℓ_1-norm of the vector $\mathbf{x}$, which is the sum of the absolute values of its components. This approach is significantly more tractable than ℓ_0-minimization due to its convex nature, allowing for efficient optimization algorithms.

In our face recognition application, ℓ_1-minimization has found practical application due to its ability to strike a balance between computational efficiency and the desired sparsity of the solution. While it may not always achieve the extreme sparsity of an ℓ_0-minimization solution, ℓ_1-minimization provides a more realistic approach for our applications.

4 Characteristic Function and Classification

With the understanding of the training set and the sparse representations of test samples, now we need to turn our attention on how to perform the classification when we successfully find the sparse representations of test sample. Here, we introduce a Characteristic function.

4.1 Definition and Application

Given a test sample $y \in \mathbb{R}^m$ and its corresponding sparse representation $\hat{x} \in \mathbb{R}^n$, obtained from the minimization techniques discussed earlier, we introduce the characteristic function δ_i for the i-th class. This function plays a crucial role in the classification process. Specifically, for any vector $\hat{x} \in \mathbb{R}^n$, applying the characteristic function $\delta_i(\hat{x})$ yields a new vector in $\mathbb{R}^n$ which has nonzero entries only at positions corresponding to the i-th class. This selective feature of δ_i can isolate the contribution of each class to the test sample's representation.

Once we apply δ_i to $\hat{x}$, the test sample y can be approximated as $\hat{y}_i = A(\delta_i(\hat{x}))$. This approximation essentially reconstructs y using only the features related to the i-th class. The final step in the classification process involves a residual-based approach, where we determine the class of y by finding the minimum residual The problem is formed as:

$$\min_i r_i(y) = \|y - A(\delta_i(\hat{x}))\|_2 \tag{1}$$

Here, $r_i(y)$ denotes the residual for class i, calculated as the ℓ_2-norm of the difference between the test sample y and its approximation $\hat{y}_i$. The class that yields the smallest residual is selected as the class where the test sample y belongs to.

This method of classification is in line with the principles of the sparse representation framework. It is also a very efficient and simple method to work with.

5 Numerical Experiments and Results

5.1 Experiment Setup and Methods

In the numerical experiment conducted, a diverse dataset was utilized to evaluate the performance of different sparse representation methods. The training set comprised 11 object classes, with each class contributing 54 unique samples, resulting in a total of 594 images. This training set ensures a comprehensive representation of each object class, providing a robust basis for the training phase. For the testing phase, a more concise set was chosen, consisting of 3 samples per object class, culminating in a total of 33 images. This selection was made to effectively test the generalization capability of the methods under study.

The experiment employed three distinct methods for sparse representation, each with its unique approach:

- l_2 **Method:** This method utilized the Pseudo Inverse of matrix A for computations. It is designed to provide a solution by minimizing the least squares error.
- l_0 **Method:** Implemented using Orthogonal Matching Pursuit (OMP), this method follows a Greedy algorithmic approach. It aims to find the sparsest solution by iteratively selecting the dictionary elements that best correlate with the data.
- l_1 **Method:** Executed using the Alternating Direction Method of Multipliers (ADMM), this method is well-suited for solving convex optimization problems. It focuses on finding solutions by breaking them into smaller pieces, which are easier to solve.

Figure 3 presents the first object class of the training set, exemplifying the nature of the data used in the experiment. From the figure, it is evident that the dataset consists of images with varying facial expressions and angles, such as left side and right side face views. This variety introduces a level of complexity to the dataset. It is noted that using a more consistent dataset, for instance, images with very centered facial orientations, could potentially yield better results by reducing the variability the algorithms need to account for.

Figure 3: First object class of the training set.

5.2 Sparse Representation and Residual

The results of the numerical experiment are analyzed here, focusing on the comparison of the sparse representation x and residuals from the three different sparse representation methods. To provide a clear context, we begin with a presentation of the test sample followed by the outcomes from each method.

Figure 4: Random test sample 1

The figure 4 illustrates a random test sample used in the experiment (Belongs to same object class as Figure 3). This sample serves as the basis for the comparison of the three methods.

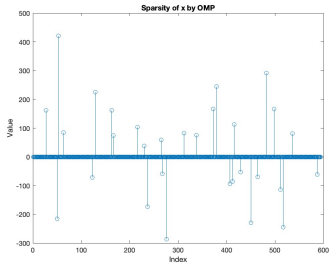


Figure 5: X by OMP

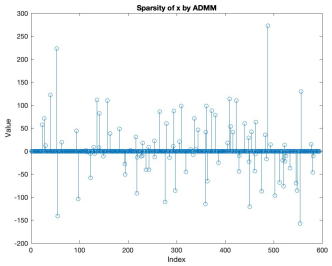


Figure 6: X by ADMM

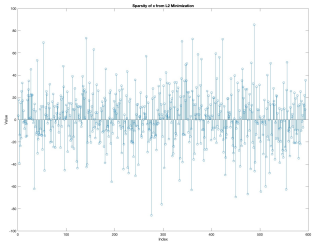


Figure 7: X by using Pseudo-inverse of A

The figures 5, 6, and 7 showcase the vectors X obtained from the l_0 (OMP), l_1 (ADMM), and l_2 (Using Pseudo-inverse of A) methods, respectively. From these images, it is evident that the vectors X obtained from the OMP and ADMM methods are relatively more sparse compared to the Pseudo-inverse of A method. The OMP method, in particular, yields the sparsest vector among the three. Conversely, the vector produced using the Pseudo-inverse of A is very dense, with large values distributed across its entire entries, demonstrating a less sparse representation.

An essential aspect of the numerical experiment was the comparison of the residuals obtained from the l_0 (OMP), l_1 (ADMM), and l_2 (Pseudo-inverse of A) methods. This comparison is crucial in assessing the accuracy and effectiveness of each method in classifying the test samples.

From the figures 8, 9, and 10, it is observable that both OMP and ADMM methods resulted in the smallest residuals at class 1. This indicates that the test sample belongs to the first class in the training set, aligning with the correct classification. Interestingly, the residual from the ADMM method is smaller than that from the OMP, suggesting a slightly more accurate classification. However, the L_2 minimization method presents a different scenario, showing the smallest residuals at class 7. This implies that the test sample is identified as belonging to class 7, which, in this case, is an incorrect classification. This variance in the results highlights the differences in the accuracy and reliability of these methods in sparse signal representation and classification tasks.

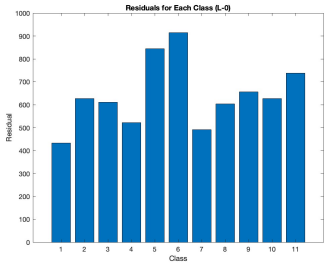


Figure 8: Residuals from OMP

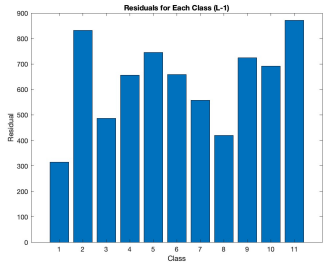


Figure 9: Residuals from ADMM

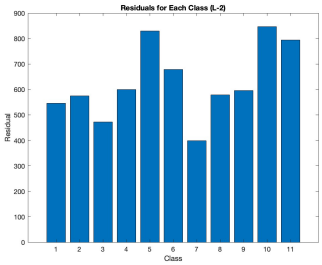


Figure 10: Residuals from L_2 Minimization

5.3 Overall Performance of l_1 and l_0 Minimization

Having established that the l_2 minimization method is not as informative or accurate for our purposes, we shifted our focus to evaluate the overall performance of the l_1 (ADMM) and l_0 (OMP) minimization methods.

For the l_1 minimization using ADMM, a grid size of 12×10 was employed. When this method was applied to all 33 images in the test set, it achieved an impressive accuracy rate of 96.6%. The average time taken for each image to be recognized was approximately 40 seconds. This result underscores the effectiveness of the ADMM method in terms of both accuracy and computational efficiency.

On the other hand, the performance of the l_0 minimization using the OMP method, with the same grid size of 12×10, did not meet expectations. However, considering that OMP is a greedy method known for its speed, we explored the possibility of increasing the grid size to enhance accuracy. Consequently, upon expanding the grid to 360×300, the OMP method exhibited a notable improvement, achieving an accuracy rate of 87.8%. Remarkably, the average processing time was only 1.2 seconds per image, which is substantially faster compared to the ADMM method. This finding highlights the trade-off between accuracy and speed in the choice of sparse representation methods, with OMP offering a rapid, albeit slightly less accurate, alternative.

6 Ability to Detect and Reject Invalid Test Samples

A critical aspect of the recognition system is its ability to distinguish between valid and invalid test samples. This ability is particularly important in real-world scenarios, where a recognition system might encounter images that do not belong to any class in the training set or are not relevant to the classification task (e.g., a face recognition system receiving an image of a subject not in the database or an image that is not a face at all).

6.1 Sparsity Concentration Index

The distribution of the estimated sparse coefficients $\hat{x}$ provides vital information about the validity of a test image. A valid test image is expected to have a sparse representation with non-zero entries predominantly concentrated on one subject. In contrast, an invalid image is characterized by sparse coefficients that are spread across multiple subjects.

To quantify the degree of concentration of these coefficients on a single class, we employ the Sparsity Concentration Index (SCI). The SCI for a coefficient vector $x \in \mathbb{R}^n$ is defined as:

$$SCI(x) = \frac{k \cdot \max_i \|\delta_i(x)\|_1 / \|x\|_1 - 1}{k - 1} \in [0, 1].$$

where k is the number of classes we have in our training set. The SCI value ranges from 0 to 1, with higher values indicating that the coefficients are more concentrated on a single class, thereby suggesting the validity of the test image. If the SCI of the solution $\hat{x}$ is close to 1, it implies that the test image closely aligns with one of the known classes, affirming its validity. Conversely, lower SCI values suggest that the test image does not strongly align with any single class, indicating its invalidity.

6.2 Application of Sparsity Concentration Index

The Sparsity Concentration Index (SCI) serves as a powerful metric for validating the authenticity of test samples in sparse representation-based classification systems. Integrating SCI into our algorithm enhances its robustness by enabling the rejection of outliers or invalid samples.

Incorporation of SCI into the existing framework is straightforward. After obtaining the sparse representation and calculating the residuals, the algorithm proceeds to compute the SCI. A predetermined threshold is set for validation. The process is as follows:

1. Calculate the SCI for the coefficient vector resulting from the sparse representation.
2. Compare the computed SCI against a predefined threshold.
3. If the SCI exceeds the threshold, the algorithm concludes that the test sample is valid and assigns it to the identified class.
4. If the SCI is below the threshold, the algorithm rejects the image, classifying it as an invalid sample.

The most challenging aspect of incorporating SCI is determining an appropriate threshold. If set too high, there is a risk of rejecting valid test samples.

6.2.1 Numerical Experiment on SCI

In our numerical experiment, we computed the SCI for all 33 valid test samples in our dataset. The smallest SCI value recorded from these samples was 0.0528. Given that all 33 test samples were known to be valid, we established this minimum SCI value as our threshold for the test.

Figure 11: Invalid test samples (1).

The above images are 3 invalid test samples we use. However, with the 0.0528 setup as the SCI threshold, these 3 images could not be detected and rejected. The reason for the very small SCI obtained from our valid test samples is likely due to the samples having weird angles or facial expressions, or they might be the ones which are not accurately detected. Therefore, we need to

consider resetting the SCI threshold to be larger. Although doing this might cause some valid test samples to be rejected, it may also improve the overall recognition accuracy, since some test samples, although they are valid (belong to the class in the training set), but might not be good test samples, will be rejected as well.

Resetting the SCI to be 0.085, all three invalid test images from Figure 11 are successfully detected and rejected. Now we attempt to validate another set of 3 invalid test samples that pose a greater challenge, as shown in Figure 12.

Figure 12: Invalid test samples (2).

With the threshold set to 0.085, all 3 test images in Figure 12 are also rejected successfully. Based on the current stage of our numerical experiment and the configuration of the training set, this value of the SCI threshold seems to be a reasonable choice.

7 Feature Extraction in Face Recognition

7.1 Introduction

In addition to the ability to detect and reject invalid test images, a crucial function for face recognition models in real-world applications is feature extraction. Effective feature extraction enables the model to recognize images even under occlusion, such as when the subject is wearing a face mask or sunglasses.

7.2 Robustness to Occlusion or Corruption

In practical scenarios, when test images are partially occluded or corrupted. The linear model is modified to $y = y_0 + e_0 = Ax_0 + e_0$, where e_0 represents a vector of errors corresponding to occluded or corrupted pixels.

Regular ADMM for $y = Ax$ does not incorporate an error term for occlusions, focusing solely on finding a sparse solution x. However, with this $y = y_0 + e_0 = Ax_0 + e_0$ setup, we adopt it into our ADMM method to explicitly handles occlusions or corruptions in the images:

1. Normalizes A to have unit L_2 norm.
2. Sets ADMM parameters: ρ, λ, ϵ, *maxIter*.
3. Initializes x, e, z, and u.
4. Iteratively updates x, e, z, and u until convergence based on $\|Ax + e - y\|_2 < \epsilon$.

The error term e is updated as follows:

$$e = \text{shrinkage}(y - A \times x, \frac{\lambda}{\rho}),$$

where the shrinkage function applies soft thresholding, effectively isolating the occluded or corrupted components of the image.

7.2.1 Numerical Experiment on Occluded Image

We use the same test sample (class 1) as in Section 5.2, but with a modification where sunglasses are added to the image. This simulates an occlusion scenario in facial recognition.

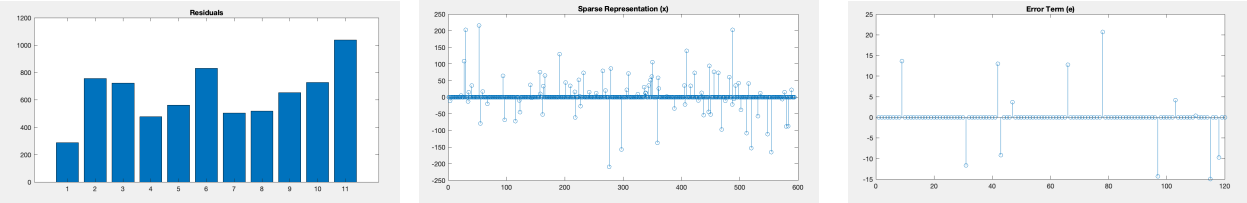
Figure 13: Test sample with sunglasses added

We apply the modified ADMM algorithm described earlier, computing the residuals as

$$\|y - e - A(\delta_i(\hat{x}))\|_2,$$

and find the smallest residuals to classify the image.

From the analysis, three key plots are presented horizontally:



(a) Residuals for each class (b) Sparse representation x (c) Error term e

Figure 14: Analysis of the modified ADMM algorithm on the occluded test sample

As shown in Figure 14a, the smallest residual occurs at class 1, indicating that the algorithm correctly recognizes the image of the class, despite the occlusion caused by the sunglasses. Furthermore, as seen in Figures 14b and 14c, both the x and e terms are sparse vectors, underscoring the effectiveness of the algorithm in handling occluded images.

Conclusion

This study has demonstrated the effectiveness of ℓ_1 and ℓ_0 minimization methods in robust face recognition. The Alternating Direction Method of Multipliers (ADMM) applied for ℓ_1 minimization emerged as a balanced approach in terms of sparsity and accuracy. On the other hand, ℓ_0 minimization using Orthogonal Matching Pursuit (OMP) offered a faster but slightly less accurate alternative. The integration of the Sparsity Concentration Index (SCI) for invalid sample rejection and the modified ADMM algorithm for occluded images significantly enhanced the system's robustness and practical applicability.

In essence, this project illustrates the effectiveness of sparse representation in face recognition, balancing sparsity, computational efficiency, and accuracy.

All the face data and code utilized in this study are accessible at this link.

References

[1] G. Bae, M. de La Gorce, T. Baltrušaitis, C. Hewitt, D. Chen, J. Valentin, R. Cipolla, and J. Shen, "DigiFace-1M: 1 Million Digital Face Images for Face Recognition," in *2023 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2023.

[2] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, Feb. 2009.