Are Sparse Representation and Dictionary Learning Good for Handwritten Character Recognition?

Chi Nhan Duong, Kha Gia Quach, and Tien D. Bui
Concordia University, Montréal, Canada
Email: {c duon, k_q, bui}@encs.concordia.ca

14th International Conference on Frontiers in Handwriting Recognition (ICFHR) September 1- 4, 2014 in Crete, Greece.
Contents

• Motivation
• Contributions
• Related works
• Sparse representation based recognition
• Experimental results
• Conclusions
Motivation

Why Sparse Representation & Dictionary Learning?

Human vision is good at recognizing different objects of the same kind, for example chairs with one leg or many legs, or someone we know under occlusions. So human visual system tends to retain certain sparse information that is common among objects of the same kind. And sparse representation has become a hot topic of investigation over the last few years.

Could these theories produce good results for handwritten character recognition as in the case of other applications?

Applications:
- Image denoising and inpainting
- Medical Imaging
- Face recognition
- Image classification
- Handwritten character recognition
Contributions

- Developing a sparse representation based system for handwriting character recognition.
- Analyzing different factors that affect the SR based system such as: the choice of input data, the size of dictionary, and computation time of this method in three benchmark databases.
- Experimental results show that using this framework, the choice of feature space is less important comparing to other methods.
Investigate the representations of character images that are invariant to various writing styles.
Related works

- **Hybrid approach**
  - Combining SVM & Convolution Neural Network [10].
  - Combining different features & different classifiers [1] => can exploit the strengths of features and classifiers, but expensive to decide which architecture is good for specific data.

- **Zhang et al. [11]**: decomposed image into three parts: low-rank component, sparse component and error (i.e. noise) ➔ mainly focus on handwriting recovery.

=> Testing with 240 images/digit and achieving 91.24% for MNIST.

- **Wei et al. [12]** took into account local information for dictionary learning and then using the learned dictionary to improve the performance.
Sparse representation based recognition

Dictionary Building

Training images

Training Img.

D*

Dictionary Building

Building the dictionary

Dictionary learning

Learned dict.

Original dict.

Sparse Coding

Residual error

Label

Testing Img.

Sparse coding

\[
\min_{\alpha, e} \|\alpha\|_1 \quad \text{s.t.} \quad y = D^* \alpha + e
\]
Sparse representation based recognition

**Algorithm 1**: Sparse representation based handwritten character recognition

**Input:**
- Set of training images of \( k \) classes
- Testing sample \( y \in \mathbb{R}^N \)

1. Stack the images of each class as columns of matrix \( D_i, i = 1, \ldots, k \).

2. **Building the dictionary:**
   (a) Use the original matrix \( D \); or
   (b) Use the learned matrix \( B^* \).
   \[
   (B^*, \Gamma^*) = \arg \min_{B, \Gamma} \|D - B\Gamma\|_F^2 + \lambda \|\Gamma\|_p
   \]

3. **Sparse Coding:** Solving (5) to obtain the sparse representation \( \alpha \) of \( y \)
   \[
   \min_{\alpha, e} \|\alpha\|_1 \quad \text{s.t.} \quad y = D^* \alpha + e
   \]

4. Compute the residuals and classify \( y \)
   \[
   r_i = \|y - D^* \alpha_i\|_2, \quad i = 1..k
   \]

**Output** label of \( \text{label}_y \leftarrow \arg \min r_i \).
## Experimental results

### Databases

<table>
<thead>
<tr>
<th>Database</th>
<th># Training</th>
<th># Testing</th>
<th>Image size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>60000</td>
<td>10000</td>
<td>28 × 28</td>
</tr>
<tr>
<td>US Portal Service (USPS)</td>
<td>7291</td>
<td>2007</td>
<td>16 × 16</td>
</tr>
<tr>
<td>CEDAR – upper case</td>
<td>11454</td>
<td>1367</td>
<td>32 × 32</td>
</tr>
<tr>
<td>CEDAR – lower case</td>
<td>7691</td>
<td>816</td>
<td>32 × 32</td>
</tr>
</tbody>
</table>
Evaluations

- Effects of dimensional reduction and feature spaces.
- Dictionary learning for character recognition.
- Effect of dictionary sizes.
- Computational time.
- Comparison with other methods
Effects of dimensional reduction

Table I
Effect of Dimensional reduction

<table>
<thead>
<tr>
<th>Input Data</th>
<th>MNIST</th>
<th>USPS</th>
<th>CEDAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LWR</td>
</tr>
<tr>
<td>Raw Img.</td>
<td>97.4 (784)</td>
<td>95.67 (256)</td>
<td>92.65(1024)</td>
</tr>
<tr>
<td>PCA(t=80)</td>
<td>97.2(47)</td>
<td>95.22(20)</td>
<td>91.05(67)</td>
</tr>
<tr>
<td>PCA(t=70)</td>
<td>97.1(29)</td>
<td>94.27(15)</td>
<td>91.05(31)</td>
</tr>
<tr>
<td>PCA(t=60)</td>
<td>95.82(18)</td>
<td>90.13(10)</td>
<td>89.83(17)</td>
</tr>
</tbody>
</table>

Gabor feature [7] is mainly designed for digits rather than for character images

Performance of this feature:
- Use k-nearest neighbor (k=3)
- MNIST: 90.45%
- USPS: 89.74%
- CEDAR:
  - Upper case: 49.63 %
  - Lower case: 52.38 %

Table II
Effect of feature spaces

<table>
<thead>
<tr>
<th>Input Data</th>
<th>MNIST</th>
<th>USPS</th>
<th>CEDAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LWR</td>
</tr>
<tr>
<td>Raw Img.</td>
<td>97.4</td>
<td>95.67</td>
<td>92.65</td>
</tr>
<tr>
<td>Gradient</td>
<td>97.35</td>
<td>95.96</td>
<td>88.36</td>
</tr>
<tr>
<td>Gabor</td>
<td>91.22</td>
<td>91.48</td>
<td>70.35</td>
</tr>
</tbody>
</table>
Effects of dictionary learning

Table III

Dictionary learning for SR

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>USPS</th>
<th>CEDAR LWR</th>
<th>CEDAR UPPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR-RAW</td>
<td>97.4</td>
<td>95.67</td>
<td>92.65</td>
<td>93.56</td>
</tr>
<tr>
<td>SR-RAW + Dict. Learning</td>
<td>97.66</td>
<td>96.26</td>
<td>89.09</td>
<td>89.61</td>
</tr>
<tr>
<td>SR-PCA</td>
<td>97.2</td>
<td>95.07</td>
<td>91.05</td>
<td>92.17</td>
</tr>
<tr>
<td>SR-PCA + Dict. Learning</td>
<td>97.25</td>
<td>95.52</td>
<td>87.26</td>
<td>87.2</td>
</tr>
</tbody>
</table>

- Dictionary learning ➔ boosting the accuracy of SR based system
- UPPR & LWR: reduce bout 3%
  - ➔ increasing the number of classes (26 instead of 10)
  - ➔ insufficient training data for some characters (only ~ 5 images/characters) ➔ reduce the quality of atoms comparing with original full images.
Effect of dictionary sizes

Evaluation system:
(1) Choose \( n \) images (per class) randomly
(2) Classification based on dictionary conducted from these images.
  Ex:
  For USPS (10 classes)
  \( n = 100 \Rightarrow \text{total images used for learning dictionary is } N = 100 \times 10 = 1000 \)
Overall performance

Table IV
COMPUTATIONAL TIME

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>CEDAR-LWR</th>
<th>CEDAR-UPPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR-RAW</td>
<td>0.25</td>
<td>0.17</td>
<td>0.28</td>
</tr>
<tr>
<td>SR-PCA</td>
<td>0.045</td>
<td>0.023</td>
<td>0.04</td>
</tr>
<tr>
<td>VAM [15]</td>
<td>0.069</td>
<td>0.018</td>
<td>0.026</td>
</tr>
<tr>
<td>(no feature extraction time)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table V
COMPARING WITH OTHER METHODS

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>LWR (26 classes)</th>
<th>UPPR (26 classes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR-RAW</td>
<td>98.21</td>
<td>92.65</td>
<td>93.56</td>
</tr>
<tr>
<td>VAM [15]</td>
<td>99.03</td>
<td>93.5</td>
<td>95.9</td>
</tr>
<tr>
<td>SAB [16]</td>
<td>NA</td>
<td>84.93</td>
<td>79.52</td>
</tr>
<tr>
<td>BLU [17]</td>
<td>NA</td>
<td>71.52</td>
<td>81.58</td>
</tr>
</tbody>
</table>

Conclusions

- In this paper, we have developed a sparse representation based system for handwritten character recognition.
- Representing the testing image as a combination of atoms in a dictionary makes the system more robust to the changes in feature spaces and the dimension of input data.
- Different factors that affect the performance of the system are also examined in our experiments.
- Although the best performance of SR based system cannot beat the state-of-the-art methods, its ability to remove the effect of feature space can help to improve its flexibility and efficiency.


References


References


Algorithm 2 Algorithm for dictionary learning [14]

**Input:** Image data matrix $X$ and parameter $\lambda$

**Step 1:** Initialize $\mathcal{D}$ randomly with unit $l_2$-norm for each column of $\mathcal{D}$

**Step 2:** Fix $\mathcal{D}$ and solve $\Lambda$ 
Solve the following minimization problem using convex optimization technique described in [18]

$$J_{\Lambda} = \arg\min_{\Lambda} \{\|X - \mathcal{D}\Lambda\|_F^2 + \lambda\|\Lambda\|_1\}$$

**Step 3:** Fix $\Lambda$ and update $\mathcal{D}$
We update $d_j$ one by one while fixing all the other columns of $\mathcal{D}$, i.e. $d_l, l \neq j$. We can find the update by optimizing the following problem.

$$J_{\mathcal{D}} = \arg\min_{\mathcal{D}} \|X - \mathcal{D}\Lambda\|_F^2 \text{ s.t. } d_j^T d_j = 1, \forall j$$

We use Lagrange multiplier $Y$ to convert the objective function. After that differentiating $J_{d_j}$ w.r.t. $d_j$, and set it to 0. We have

$$d_j = Y \alpha_j^T (\alpha_j \alpha_j^T - \lambda)^{-1}$$

$$d_j = Y \alpha_j^T / \|Y \alpha_j\|_2$$

**Step 4:** Go back to step 2 until the values of $J_{\mathcal{D}}$ and $J_{\Lambda}$ are converged or the maximum number of iterations is reached. Finally, output $\mathcal{D}$.

**Output:** $\mathcal{D}$
