Sparse Coding

Acknowledgement: The slides adapted from the ones prepared by M. Elad of the Technion - Israel Institute of Technology (general discussion) and L. Xu et al. of the Chinese University of Hong Kong (LO-smoothing)

Erkut Erdem
Hacettepe University
Computer Vision Lab (HUCVL)

Denoising By Energy Minimization

Many of the proposed image denoising algorithms are related to the minimization of an energy function of the form

\[ f(x) = \frac{1}{2} \|x - y\|^2_2 + G(x) \]

- \( y \): Given measurements
- \( x \): Unknown to be recovered

• This is in-fact a Bayesian point of view, adopting the Maximum-A-posteriori Probability (MAP) estimation.
• Clearly, the wisdom in such an approach is within the choice of the prior – modeling the images of interest.

The Evolution of G(x)

During the past several decades we have made all sort of guesses about the prior G(x) for images:

\[ G(x) = \lambda \|x\|^2_2 \quad G(x) = \lambda \|Lx\|^2_2 \quad G(x) = \lambda \|Lx\|^1_1 \quad G(x) = \lambda p \{Lx\} \]

- Energy
- Smoothness
- Adapt + Smooth
- Robust Statistics

\[ G(x) = \lambda \|\nabla x\|_1 \quad G(x) = \lambda \|Wx\|_1 \quad G(x) = \lambda \|Lx\|_1 \]

- Total-Variation
- Wavelet Sparsity
- Sparse & Redundant

- Hidden Markov Models
- Compression algorithms as priors
- …
Sparse Modeling of Signals

- Every column in \( D \) (dictionary) is a prototype signal (atom).
- The vector \( \alpha \) is generated randomly with few (say \( L \)) non-zeros at random locations and with random values.
- We shall refer to this model as Sparseland

Interesting Model:
- Simple: Every generated signal is built as a linear combination of few atoms from our dictionary \( D \)
- Rich: A general model: the obtained signals are a union of many low-dimensional Gaussians.
- Familiar: We have been using this model in other context for a while now (wavelet, JPEG, …).

Sparseland Signals are Special

Our signal model is thus: \( x = D\alpha \) where \( \alpha \) is sparse

Sparse & Redundant Rep. Modeling?

Our signal model is thus: \( x = D\alpha \) where \( \alpha \) is sparse
Sparse & Redundant Rep. Modeling?

As $p \to 0$ we get a count of the non-zeros in the vector $\| \alpha \|^0_0$

Our signal model is thus: $x = D\alpha$ where $\| \alpha \|^0_0 \leq L$

Back to Our MAP Energy Function

- $L_0$ norm effectively counts the number of non-zeros in $\alpha$.
- The vector $\alpha$ is the representation (sparse/redundant) of the desired signal $x$.

- The core idea: while few ($L$ out of $K$) atoms can be merged to form the true signal, the noise cannot be fitted well. Thus, we obtain an effective projection of the noise onto a very low-dimensional space, thus getting denoising effect.

Wait! There are Some Issues

- **Numerical Problems:** How should we solve or approximate the solution of the problem

  $$\min_{\alpha} \| D\alpha - y \|^2_0 \quad \text{s.t.} \quad \| \alpha \|^0_0 \leq L$$

  or

  $$\min_{\alpha} \| \alpha \|^0_0 \quad \text{s.t.} \quad \| D\alpha - y \|^2_0 \leq \epsilon^2$$

  or

  $$\min_{\alpha} \lambda \| \alpha \|^1_1 + \| D\alpha - y \|^2_0$$

- **Theoretical Problems:** Is there a unique sparse representation? If we are to approximate the solution somehow, how close will we get?

- **Practical Problems:** What dictionary $D$ should we use, such that all this leads to effective denoising? Will all this work in applications?

To Summarize So Far …

- Image denoising (and many other problems in image processing) requires a model for the desired image

  $D\alpha - y = 0$

  or

  $\frac{1}{2} \| x - y \|^2_2$

  What can we do?

  - Use a model for signals/images based on sparse and redundant representations

    There are some issues:
    1. Theoretical
    2. How to approximate?
    3. What about $D$?
Let's Start with the Noiseless Problem

Suppose we build a signal by the relation \( \mathbf{D}\alpha = \mathbf{x} \)

We aim to find the signal's representation:

\[ \hat{\alpha} = \text{ArgMin}_{\alpha} \|\alpha\|_0^0 \quad \text{s.t.} \quad \mathbf{x} = \mathbf{D}\alpha \]

Uniqueness

Why should we necessarily get \( \hat{\alpha} = \alpha \) ?

It might happen that eventually \( \|\hat{\alpha}\|_0 < \|\alpha\|_0 \).

Uniqueness Rule

Suppose this problem has been solved somehow

\[ \hat{\alpha} = \text{ArgMin}_{\alpha} \|\alpha\|_0^0 \quad \text{s.t.} \quad \mathbf{x} = \mathbf{D}\alpha \]

If we found a representation that satisfy

\[ \|\hat{\alpha}\|_0 < \frac{\sigma}{2} \]

Then necessarily it is unique (the sparsest).

This result implies that if \( \mathbf{M} \) generates signals using “sparse enough” \( \alpha \), the solution of the above will find it exactly.

Matrix “Spark”

Definition: Given a matrix \( \mathbf{D} \), \( \sigma = \text{Spark}(\mathbf{D}) \) is the smallest number of columns that are linearly dependent.

Donoho & E. (’02)

Example:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Rank = 4
Spark = 3

\* In tensor decomposition, Kruskal defined something similar already in 1989.

Our Goal

\[
\min_{\alpha} \|\alpha\|_0^0 \quad \text{s.t.} \quad \|\mathbf{D}\alpha - \mathbf{y}\|_2^2 \leq \varepsilon^2
\]

Here is a recipe for solving this problem:

This is a combinatorial problem, proven to be NP-Hard!

1. Set \( L = 1 \)
2. Gather all the supports \( |S| \) of cardinality \( L \)
3. Solve the LS problem \( \min_{\alpha} \|\mathbf{D}\alpha - \mathbf{y}\|_2^2 \quad \text{s.t.} \text{ supp}(\alpha) = S \) for each support
4. Set \( L = L + 1 \)
5. There are \( L \) such supports
6. No?
7. Yes?
8. \( \text{LS error} \leq \varepsilon^2 ? \)
9. Assume: \( K = 1000, L = 10 \) (known!), 1 nano-sec per each LS

We shall need \(~8e+6\) years to solve this problem!!!!!
Pursuit Algorithms

\[ \min_{\alpha} \| \alpha \|_0 \quad \text{s.t.} \quad \| D \alpha - y \|_2^2 \leq \varepsilon^2 \]

There are various algorithms designed for approximating the solution of this problem:

- **Greedy Algorithms:** Matching Pursuit, Orthogonal Matching Pursuit (OMP), Least-Squares-OMP, Weak Matching Pursuit, Block Matching Pursuit [1993-today].
- **Relaxation Algorithms:** Basis Pursuit (a.k.a. LASSO), Dantzig Selector & numerical ways to handle them [1995-today].
- **Hybrid Algorithms:** StOMP, CoSaMP, Subspace Pursuit, Iterative Hard-Thresholding [2007-today].
- ...
**BP and MP Equivalence (No Noise)**

\[ \hat{\alpha} = \text{ArgMin}_{\alpha} \|\alpha\|_0 \quad \text{s.t.} \quad x = D\alpha \]

**Equivalence**

Given a signal \( x \) with a representation \( x = D\alpha \), assuming that \( \|\alpha\|_0 < 0.5(1+1/\mu) \), BP and MP are guaranteed to find the sparsest solution.

- MP and BP are different in general (hard to say which is better).
- The above result corresponds to the worst-case, and as such, it is too pessimistic.
- Average performance results are available too, showing much better bounds [Donoho ('02)] [Candes et al. ('04)] [Tanner et al. ('05)] [E. ('06)] [Tropp et al. ('06)] … [Candes et al. ('09)].

---

**BP Stability for the Noisy Case**

Given a signal \( y = D\alpha + \nu \) with a representation satisfying \( \|\alpha\|_0 < 1/3\mu \) and a white Gaussian noise \( \nu \sim N(0, \sigma^2 I) \), BP will show stability, i.e.,

\[ \|\hat{\alpha}_{BP} - \alpha\|_2^2 < \text{Const}(\lambda) \cdot \log K \cdot \|\alpha\|_0 \cdot \sigma^2 \]

- For \( \sigma=0 \) we get a weaker version of the previous result.
- This result is the oracle's error, multiplied by \( C \cdot \log K \).
- Similar results exist for other pursuit algorithms (Dantzig Selector, Orthogonal Matching Pursuit, CoSaMP, Subspace Pursuit, …)
To Summarize So Far …

- Image denoising (and many other problems in image processing) requires a model for the desired image.
- Use a model for signals/images based on sparse and redundant representations.
- What do we do?
- Problems?
- The Dictionary D should be found somehow!!!
- We have seen that there are approximation methods to find the sparsest solution, and there are theoretical results that guarantee their success.

What Should D Be?

- Our Assumption: Good-behaved Images have a sparse representation.
- D should be chosen such that it sparsifies the representations.
- One approach to choose D is from a known set of transforms (Steerable wavelet, Curvelet, Contourlets, Bandlets, Shearlets …).

Measure of Quality for D

- Min \( D, A \)
- Each example is a linear combination of atoms from D
- Each example has a sparse representation with no more than L atoms

\[
\begin{align*}
\text{Min}_{D, A} & \sum_{j=1}^{P} \left| \langle D_{\alpha_j}, x_j \rangle \right|^2 \\
\text{s.t.} \quad & \forall j, \| \alpha_j \|_0 \leq L
\end{align*}
\]

K–Means For Clustering

- Clustering: An extreme sparse representation

\[
\hat{x} = \text{arg min}_{\alpha} \| y \|_0 \quad \text{s.t.} \quad \frac{1}{2} \| D\alpha - y \|_2^2 \leq \varepsilon^2
\]

K = \| x - y \|_2^2

K–Means For Clustering

- Initialize D
- Sparse Coding
- Nearest Neighbor
- Dictionary Update
- Column-by-Column by Mean computation over the relevant examples

[Field & Olshausen (’96)]
[Engan et al. (’99)]
[Lewicki & Sejnowski (’00)]
[Cotter et al. (’05)]
[Gribonval et al. (’04)]
[Aharon, E. & Bruckstein (’04)]
[Aharon, E. & Bruckstein (’05)]
The K–SVD Algorithm – General

[Aharon, E. & Bruckstein (’04, ’05)]

Initialize \( D \)

Sparse Coding
Use Matching Pursuit

Dictionary Update
Column-by-Column by SVD computation over the relevant examples

K–SVD: Sparse Coding Stage

\[
\min_{\alpha} \sum_{j=1}^{p} \| D \alpha_j - x_j \|_2^2 \quad \text{s.t.} \quad \forall j, \| \alpha_j \|_0 \leq L
\]

\( D \) is known!
For the \( j \)th item we solve

\[
\min_{\alpha} \| D \alpha - x \|_2^2 \quad \text{s.t.} \quad \| \alpha \|_0 \leq L
\]

Solved by A Pursuit Algorithm

K–SVD: Dictionary Update Stage

Refer only to the examples that use the column \( d_k \)

We should solve:

\[
\min_{d_k, \alpha_k} \| \alpha_k d_k^T - E \|_F^2
\]

Fixing all \( A \) and \( D \) apart from the \( k \)th column, and seek both \( d_k \) and the \( k \)th column in \( A \) to better fit the residual!

K–SVD: Dictionary Update Stage

Refer only to the examples that use the column \( d_k \)

We should solve:

\[
\min_{d_k, \alpha_k} \| \alpha_k d_k^T - E \|_F^2
\]

Fixing all \( A \) and \( D \) apart from the \( k \)th column, and seek both \( d_k \) and the \( k \)th column in \( A \) to better fit the residual!
K-SVD: Algorithm

Train: Find the best dictionary to represent the data samples \( \{y_i\}_i \), as sparse compositions, by solving

\[
\min_{D} \| Y - DX \|_F \quad \text{subject to} \quad \|x_i\|_0 \leq T_0
\]

Initialization: Set the dictionary matrix \( D^{(0)} \in \mathbb{R}^{N \times K} \) with \( K \) normalized columns. Set \( j = 1 \). Repeat until convergence (stopping rule):

- Sparse Coding Stage: Use any pursuit algorithm to compute the representation vectors \( x_i \) for each example \( y_i \), by approximating the solution of

\[
\min_{x_i} \| y_i - D x_i \|_2 \quad \text{subject to} \quad \|x_i\|_0 \leq T_0
\]

- Coefficients Update Stage: For each column \( k = 1, 2, \ldots, K \) in \( D^{(j-1)} \), update it by

  - Define the group of atoms that use this atom, \( \omega_k = \{ i \mid 1 \leq i \leq N, x_i(k) \neq 0 \} \).
  - Compute the overall representation error matrix, \( E_k \), by

\[
E_k = Y - \sum_{i \in \omega_k} x_i(k) \alpha_i
\]

  - Restrict \( E_k \) by choosing only the column corresponding to \( x_k \), and obtain \( K^k \).
  - Apply SVD decomposition \( K^k = U \Sigma V^T \). Choose the updated dictionary column \( \hat{D}_k \) to be the first column of \( U \). Update the coefficient vector \( x_k \) to be the first column of \( Y \) multiplied by \( \hat{D}_k \).

- Set \( j = j + 1 \).

From Local to Global Treatment

- The K-SVD algorithm is reasonable for low-dimension signals (\( N \) in the range 10-400). As \( N \) grows, the complexity and the memory requirements of the K-SVD becomes prohibitive.
- So, how should large images be handled?
- The solution: Force shift-invariant sparsity - on each patch of size \( N \times N \) in the image, including overlaps.

\[
\hat{x} = \text{ArgMin}_{x_i} \left\{ \frac{1}{2} \| x - Y \|_2^2 + \sum_{ij} R_{ij}^2 \| x - D \alpha_{ij} \|_2^2 \right\} \\
\text{s.t.} \quad \| \alpha_{ij} \|_0 \leq L
\]

Extracts a patch in the \( ij \) location

Our prior

What Data to Train On?

Option 1:
- Use a database of images,
- We tried that, and it works fine (~0.5-1dB below the state-of-the-art).

Option 2:
- Use the corrupted image itself!!
- Simply sweep through all patches of size \( N \times N \) (overlapping blocks),
- Image of size 1000 pixels \( \Rightarrow \approx 10^6 \) examples to use – more than enough.
- This works much better!

To Summarize So Far …

Image denoising (and many other problems in image processing) requires a model for the desired image

What do we do?

Use a model for signals/images based on sparse and redundant representations

Problems?

Will it all work in applications?

We have seen that there are approximation methods to find the sparsest solution, and there are theoretical results that guarantee their success.
Complexity of this algorithm: $O(N^2 \times K \times L \times \text{Iterations})$ per pixel. For $N=8$, $L=1$, $K=256$, and 10 iterations, we need 160,000 (!!) operations per pixel.

The results of the K-SVD algorithm compete favorably with the state-of-the-art. In a recent work that extended this algorithm to use joint sparse representation on the patches, the best published denoising performance are obtained [Mairal, Bach, Ponce, Sapiro & Zisserman (09)].

When turning to handle color images, the main difficulty is in defining the relation between the color layers – R, G, and B.

The solution with the above algorithm is simple – consider 3D patches or 8-by-8 with the 3 color layers, and the dictionary will detect the proper relations.
The K-SVD algorithm leads to state-of-the-art denoising results, giving ~1dB better results compared to [Mcauley et. al. ('06)] which implements a learned MRF model (Field-of-Experts).

Image Inpainting – The Basics

- Assume: the signal $x$ has been created by $x = D\alpha_0$ with very sparse $\alpha_0$.
- Missing values in $x$ imply missing rows in this linear system.
- By removing these rows, we get $D\tilde{\alpha} = \tilde{x}$.
- Now solve $\min_{\alpha} \|\alpha\|_0$ s.t. $\tilde{x} = D\tilde{\alpha}$.
- If $\alpha_0$ was sparse enough, it will be the solution of the above problem! Thus, computing $D\alpha_0$ recovers $x$ perfectly.

Side Note: Compressed-Sensing

- **Compressed Sensing** is leaning on the very same principal, leading to alternative sampling theorems.
- Assume: the signal $x$ has been created by $x = D\alpha_0$ with very sparse $\alpha_0$.
- Multiply this set of equations by the matrix $Q$ which reduces the number of rows.
- The new, smaller, system of equations is $QD\alpha = Qx \Rightarrow \tilde{D}\alpha = \tilde{x}$.
- If $\alpha_0$ was sparse enough, it will be the sparsest solution of the new system, thus, computing $D\alpha_0$ recovers $x$ perfectly.
- Compressed sensing focuses on conditions for this to happen, guaranteeing such recovery.
Experiments lead to state-of-the-art inpainting results.

Original 80% missing Result

Experiments lead to state-of-the-art inpainting results.

Original 80% missing Result

Experiments lead to state-of-the-art inpainting results.

Experiments lead to state-of-the-art inpainting results.

Image Compression [Bryt and E. ('08)]

- The problem: Compressing photo-ID images.
- General purpose methods (JPEG, JPEG2000) do not take into account the specific family.
- By adapting to the image-content (PCA/K-SVD), better results could be obtained.
- For these techniques to operate well, train dictionaries locally (per patch) using a training set of images is required.
- In PCA, only the (quantized) coefficients are stored, whereas the K-SVD requires storage of the indices as well.
- Geometric alignment of the image is very helpful and should be done [Goldenberg, Kimmel, & E. ('08)].
Image Compression

Detect main features and warp the images to a common reference (20 parameters)

Divide the image into disjoint 15-by-15 patches. For each compute mean and dictionary

Per each patch find the operating parameters (number of atoms L, quantization Q)

Warp, remove the mean from each patch, sparse code using L atoms, apply Q, and dewarp

Training set (2500 images)

On the training set

On the test image

Image Compression Results

Original
JPEG
JPEG-2000
Local-PCA
K-SVD

Results for 820 Bytes per each file

Results for 550 Bytes per each file

Image Compression Results

Original
JPEG
JPEG-2000
Local-PCA
K-SVD

Results for 400 Bytes per each file
Deblocking the Results [Bryt and E. (09)]

K-SVD results with and without deblocking

Deblock (6.24)  Deblock (5.27)  Deblock (6.03)  Deblock (11.32)

Super-Resolution [Zeyde, Procter, & E. (11)]

- Given a low-resolution image, we desire to enlarge it while producing a sharp looking result. This problem is referred to as “Single-Image Super-Resolution”.
- Image scale-up using bicubic interpolation is far from being satisfactory for this task.
- Recently, a sparse and redundant representation technique was proposed [Yang, Wright, Huang, and Ma (08)] for solving this problem, by training a coupled-dictionaries for the low- and high res. images.
- We extended and improved their algorithms and results.

Super-Resolution – Results (1)

The training image: \(717 \times 717\) pixels, providing a set of 54,289 training patch-pairs.

The SR Result:

<table>
<thead>
<tr>
<th>Technique</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal Image</td>
<td>16.95</td>
</tr>
<tr>
<td>Bicubic</td>
<td>14.68</td>
</tr>
</tbody>
</table>

Super-Resolution – Results (1)

An amazing variety of practical problems (design, analysis, and operation) can be formulated as an optimization problem, or some variation thereof, mathematical optimization has it is widely used in engineering, in control systems, and optimal design problems and aerospace engineering. Optimization design and operation, finance, supply chain and other areas. The list of applications is endless. For most of these applications, such as a human decision maker, system design process, check the results, and modify when necessary. This human decision maker the optimization problem, e.g., buy or sell portfolio.

SR Result

Ideal Image

Given Image

Bicubic interpolation

PSNR=16.95dB

Ideal Image

SR Result

PSNR=14.68dB
Given image

Scaled-Up (factor 2:1) using the proposed algorithm, PSNR=29.32dB (3.32dB improvement over bicubic)

Super-Resolution – Results (2)

The Original  Bicubic Interpolation  SR result

L0-Image Smoothing

General goals:
• Suppress insignificant details
• Maintain major edges
L0-Smoothing Method

A general and effective global smoothing strategy based on a sparsity measure

\[ c(f) := \#\{p | |\nabla f_p| \neq 0\} \]

which corresponds to the L0-norm of gradient

Two Features

1. Flattening insignificant details
   By removing small non-zero gradients

Two Features

2. Enhancing prominent edges
   Because large gradients receive the same penalty as small ones

Our Framework in 1D

- Constrain # of non-zero gradients
  \[ c(f) = \#\{p | |f_p - f_{p+1}| \neq 0\} = k \]

- Make the result similar to the input
  \[ \min_p \sum (f_p - g_p)^2 \]

Objective function

\[ \min_p \sum (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = k \]
Our Framework in 1D

- Input 1D signal $g$

\[
\min_f \sum_p (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = 1
\]

\[
\min_f \sum_p (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = 2
\]

\[
\min_f \sum_p (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = 5
\]

\[
\min_f \sum_p (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = 200
\]
**Transformation**

\[
\min_{f} \sum_{p} (f_p - g_p)^2 \quad \text{s.t.} \quad c(f) = k
\]

\[
\min_{f} \sum_{p} (f_p - g_p)^2 + \lambda \cdot c(f)
\]

**2D Image**

\[
\min_{f} \sum_{p} (f_p - g_p)^2 + \lambda \cdot c(\partial_x f, \partial_y f)
\]

\[
c(\partial_x f, \partial_y f) = \# \{ p \mid |\partial_x f_p| + |\partial_y f_p| \neq 0 \}
\]

Finding the global optimum is NP hard

**Approximation**

\[
\min_{f} \sum_{p} (f_p - g_p)^2 + \lambda \cdot c(h, v)
\]

\[
+ \beta \sum_{p} ((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2)
\]

Separately estimate \( f \) and \((h, v)\)

**Iterative Optimization**

- Compute \( f \) given \( h, v \)

\[
E(f) = \sum_{p} (f_p - g_p)^2 + \beta \cdot \left( (\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right)
\]

Both the sub-problems are with closed-form solutions

\[
E(h, v) = \sum_{p} \left( (\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right) + \frac{\lambda}{\beta} c(h, v)
\]

- Gradually approximate the original problem

\[
\beta \leftarrow 2\beta
\]
One Example

Converge in 15 iterations

Iteration #1

Iteration #2

Iteration #3

Iteration #4

Iteration #5

Iteration #6

Iteration #7

Iteration #8

Iteration #9

Iteration #10

Iteration #11

Iteration #12

Iteration #13

Iteration #14

Iteration #15

Smoothing Strength

Input

Smoothing Strength

$\lambda = 0.01$

Smoothing Strength

$\lambda = 0.02$
Smoothing Strength

\[ \lambda = 0.03 \]

Comparison

Total Variation

L0 Smoothing