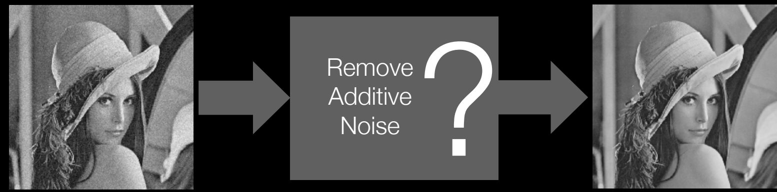
BIL 717
Image Processing
Apr. 8, 2015

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 (L0-smoothing)

Erkut Erdem
Hacettepe University
Computer Vision Lab (HUCVL)

Noise Removal?



- Important: (i) Practical application; (ii) A convenient platform (being the simplest inverse problem) for testing basic ideas in image processing, and then generalizing to more complex problems.
- Many Considered Directions: Partial differential equations, Statistical estimators, Adaptive filters, Inverse problems & regularization, Wavelets, Example-based techniques, Sparse representations, …

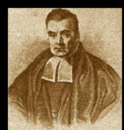
Denoising By Energy Minimization

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

$$f(\underline{x}) = \underbrace{\frac{1}{2}\|\underline{x}-\underline{y}\|_2^2}_{\text{Relation to measurements}} + \underbrace{G(\underline{x})}_{\text{Prior or regularization}}$$

$\underline{y}$: Given measurements

$\underline{x}$: Unknown to be recovered


Thomas Bayes
1702 - 1761

- 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($\underline{x}$) for images:

$G(\underline{x}) = \lambda\|\underline{x}\|_2^2$ Energy

$G(\underline{x}) = \lambda\|\mathbf{L}\underline{x}\|_2^2$ Smoothness

$G(\underline{x}) = \lambda\|\mathbf{L}\underline{x}\|_{\mathbf{W}}^2$ Adapt+ Smooth

$G(\underline{x}) = \lambda\rho\{\mathbf{L}\underline{x}\}$ Robust Statistics

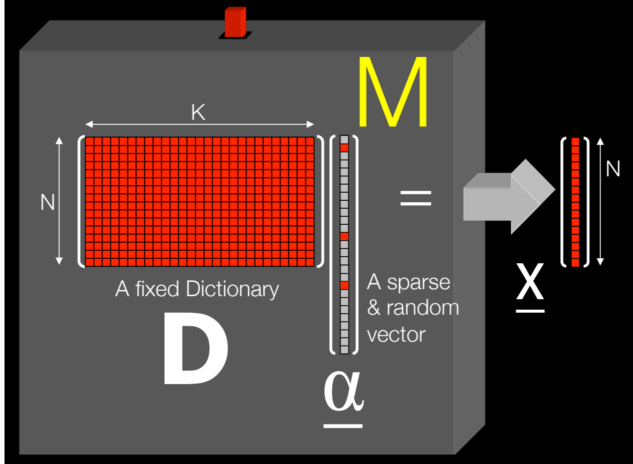
$G(\underline{x}) = \lambda\|\nabla\underline{x}\|_1$ Total-Variation

$G(\underline{x}) = \lambda\|\mathbf{W}\underline{x}\|_1$ Wavelet Sparsity

$G(\underline{x}) = \lambda\|\underline{\alpha}\|_0^0$ for $\underline{x} = \mathbf{D}\underline{\alpha}$ Sparse & Redundant

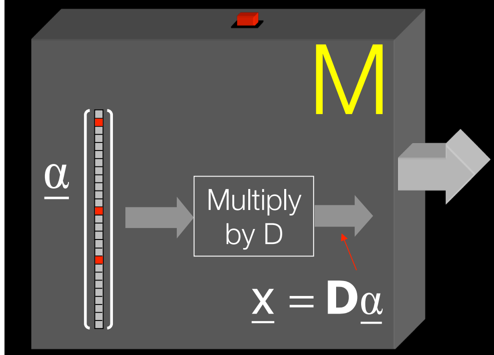
- Hidden Markov Models,
- Compression algorithms as priors,
- …

Sparse Modeling of Signals



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

Sparseland Signals are Special



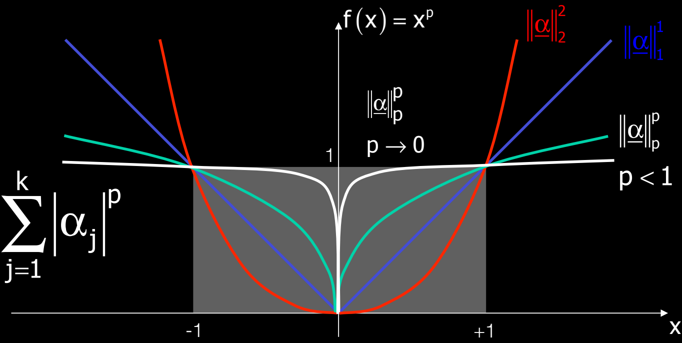
Interesting Model:

- 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, …).

Sparse & Redundant Rep. Modeling?

Our signal model is thus: $\underline{x} = \mathbf{D}\underline{\alpha}$ where $\underline{\alpha}$ is sparse

Sparse & Redundant Rep. Modeling?

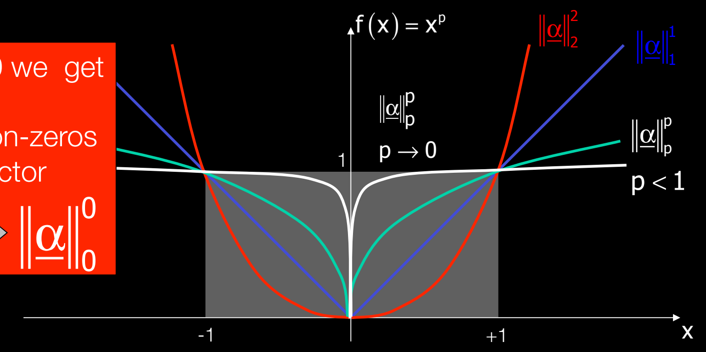


$$\|\underline{\alpha}\|_p^p = \sum_{j=1}^{k} |\alpha_j|^p$$

Our signal model is thus: $\underline{x} = \mathbf{D}\underline{\alpha}$ where $\underline{\alpha}$ is sparse

Sparse & Redundant Rep. Modeling?

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



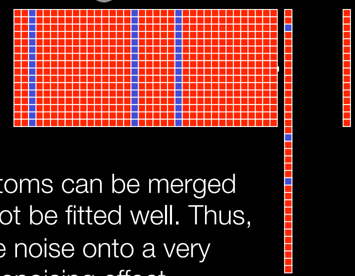
Our signal model is thus: $\underline{x} = \mathbf{D}\underline{\alpha}$ where $\|\underline{\alpha}\|_0^0 \le L$

Back to Our MAP Energy Function

- L_0 norm effectively counts the number of non-zeros in $\underline{\alpha}$.
- The vector $\underline{\alpha}$ 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.

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

$$\mathbf{D}\underline{\alpha}-\underline{y}=$$



Wait! There are Some Issues

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

$$\min_{\underline{\alpha}} \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \text{ s.t. } \|\underline{\alpha}\|_0^0 \le L \quad \text{or} \quad \min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \text{ s.t. } \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2$$

$$\text{or} \quad \min_{\underline{\alpha}} \lambda\|\underline{\alpha}\|_0^0 + \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2$$

- 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

What can we do?

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

Great! No?

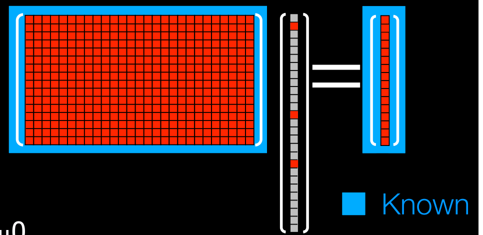
There are some issues:

1. Theoretical
2. How to approximate?
3. What about D?

Lets Start with the Noiseless Problem

Suppose we build a signal by the relation

$$\mathbf{D}\underline{\alpha} = \underline{x}$$



We aim to find the signal's representation:

$$\hat{\underline{\alpha}} = \underset{\underline{\alpha}}{\mathrm{ArgMin}} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \underline{x} = \mathbf{D}\underline{\alpha}$$

Uniqueness

Why should we necessarily get $\hat{\underline{\alpha}} = \underline{\alpha}$?

It might happen that eventually $\|\hat{\underline{\alpha}}\|_0^0 < \|\underline{\alpha}\|_0^0$.

Matrix "Spark"

Definition: Given a matrix D, σ =Spark{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.

Uniqueness Rule

Suppose this problem has been solved somehow

$$\hat{\underline{\alpha}} = \underset{\underline{\alpha}}{\mathrm{ArgMin}} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \underline{x} = \mathbf{D}\underline{\alpha}$$

Uniqueness

Donoho & E. ('02)

If we found a representation that satisfy

$$\|\hat{\underline{\alpha}}\|_0 < \frac{\sigma}{2}$$

Then necessarily it is unique (the sparsest).

This result implies that if $\mathcal{M}$ generates signals using "sparse enough" $\underline{\alpha}$, the solution of the above will find it exactly.

Our Goal

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2$$

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

Here is a recipe for solving this problem:

Set L=1 → Gather all the supports $\{S_i\}_i$ of cardinality L → Solve the LS problem $\min_{\underline{\alpha}} \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2$ s.t. $\mathrm{supp}(\underline{\alpha}) = S_i$ for each support → LS error ≤ ε^2 ?

No → Set L=L+1 → Gather all the supports

Yes → Done

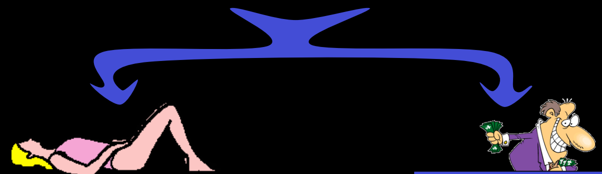
There are $\binom{K}{L}$ such supports

Assume: K=1000, L=10 (known!), 1 nano-sec per each LS

We shall need ~8e+6 years to solve this problem !!!!!

Lets Approximate

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \text{ s.t. } \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2$$



Relaxation methods

Smooth the L_0 and use continuous optimization techniques

Greedy methods

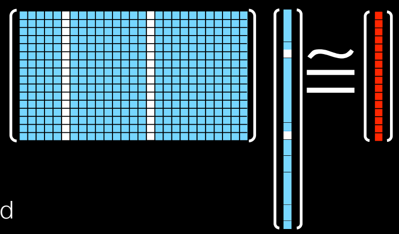
Build the solution one non-zero element at a time

Relaxation – The Basis Pursuit (BP)

Instead of solving		Solve Instead
$\text{Min}_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \text{ s.t. } \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2 \le \varepsilon$	⇒	$\text{Min}_{\underline{\alpha}} \|\underline{\alpha}\|_1 \text{ s.t. } \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2 \le \varepsilon$

- This is known as the Basis-Pursuit (BP) [Chen, Donoho & Saunders ('95)].
- The newly defined problem is convex (quad. programming).
- Very efficient solvers can be deployed:
  - Interior point methods [Chen, Donoho, & Saunders ('95)] [Kim, Koh, Lustig, Boyd, & D. Gorinevsky ('07)].
  - Sequential shrinkage for union of ortho-bases [Bruce et.al. ('98)].
  - Iterative shrinkage [Figuerido & Nowak ('03)] [Daubechies, Defrise, & De-Mole ('04)] [E. ('05)] [E., Matalon, & Zibulevsky ('06)] [Beck & Teboulle ('09)] …

Go Greedy: Matching Pursuit (MP)



- The MP is one of the greedy algorithms that finds one atom at a time [Mallat & Zhang ('93)].
- Step 1: find the one atom that best matches the signal.
- Next steps: given the previously found atoms, find the next one to best fit the residual.
- The algorithm stops when the error $\|\mathbf{D}\underline{\alpha} - \underline{y}\|_2$ is below the destination threshold.
- The Orthogonal MP (OMP) is an improved version that re-evaluates the coefficients by Least-Squares after each round.

Pursuit Algorithms

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0^0 \text{ s.t. } \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2$$

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), Dnatzig 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)

$$\underline{\hat{\alpha}} = \underset{\underline{\alpha}}{\text{ArgMin}} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \underline{x} = \mathbf{D}\underline{\alpha}$$

BP and MP Equivalence (No Noise)

Equivalence

Donoho & E. ('02)
Gribonval & Nielsen ('03)
Tropp ('03)
Temlyakov ('03)

Given a signal $\underline{x}$ with a representation $\underline{x} = \mathbf{D}\underline{\alpha}$, assuming that $\|\underline{\alpha}\|_0^0 < 0.5(1+1/\mu)$, 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 ('04)] [Candes et.al. ('04)] [Tanner et.al. ('05)] [E. ('06)] [Tropp et.al. ('06)] … [Candes et. al. ('09)].

BP Stability for the Noisy Case

$$\min_{\underline{\alpha}} \lambda \|\underline{\alpha}\|_1 + \|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2$$

BP Stability for the Noisy Case

Stability

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

$$\|\underline{\hat{\alpha}}_{BP} - \underline{\alpha}\|_2^2 < \text{Const}(\lambda) \cdot \log K \cdot \|\underline{\alpha}\|_0^0 \cdot \sigma^2$$

Ben-Haim, Eldar & E. ('09)

* With very high probability

- For σ=0 we get a weaker version of the previous result.
- This result is the oracle's error, multuiplied by C· logK.
- 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

What do we do?

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

Problems?

We have seen that there are approximation methods to find the sparsest solution, and there are theoretical results that guarantee their success.

What next?

The Dictionary D should be found somehow !!!

What Should D Be?

$$\hat{\underline{\alpha}} = \underset{\underline{\alpha}}{\arg\min} \|\underline{\alpha}\|_0^0 \quad \text{s.t.} \quad \frac{1}{2}\|\mathbf{D}\underline{\alpha} - \underline{y}\|_2^2 \le \varepsilon^2 \;\Rightarrow\; \hat{\underline{x}} = \mathbf{D}\hat{\underline{\alpha}}$$

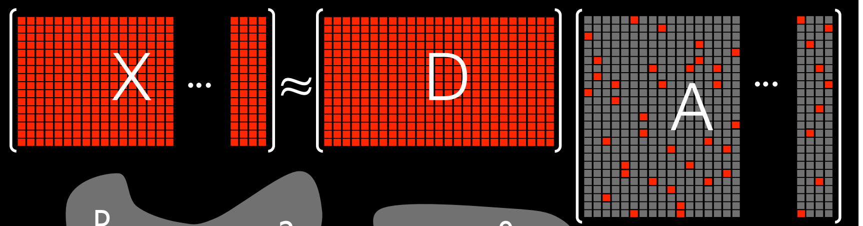
Our Assumption: Good-behaved Images have a sparse representation

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 …)

The approach we will take for building D is training it, based on Learning from Image Examples

Measure of Quality for D



$$\underset{\mathbf{D},\mathbf{A}}{\text{Min}} \sum_{j=1}^{P} \|\mathbf{D}\underline{\alpha}_j - \underline{x}_j\|_2^2 \quad \text{s.t.} \; \forall j, \|\underline{\alpha}_j\|_0^0 \le L$$

Each example is a linear combination of atoms from D

Each example has a sparse representation with no more than L atoms

[Field & Olshausen ('96)]
[Engan et. al. ('99)]
[Lewicki & Sejnowski ('00)]
[Cotter et. al. ('03)]
[Gribonval et. al. ('04)]
[Aharon, E. & Bruckstein ('04)]
[Aharon, E. & Bruckstein ('05)]

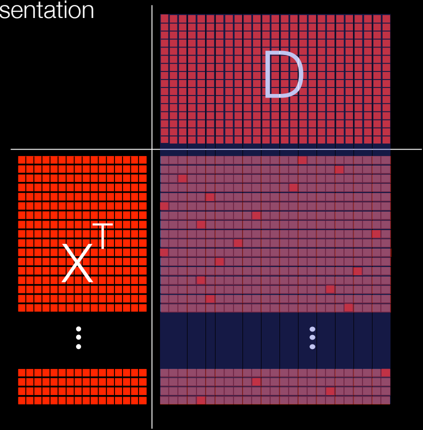
K–Means For Clustering

Clustering: An extreme sparse representation

Initialize D

Sparse Coding
Nearest Neighbor

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



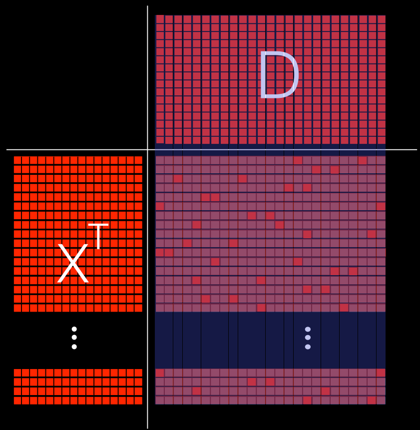
The K–SVD Algorithm – General

[Aharon, E. & Bruckstein ('04,'05)]

Initialize D

Sparse Coding
Use Matching Pursuit

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



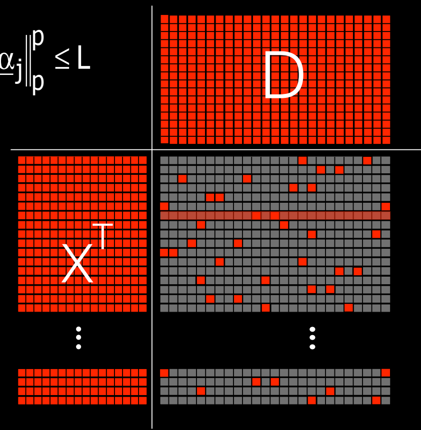
K–SVD: Sparse Coding Stage

$$\underset{\mathbf{A}}{\text{Min}} \sum_{j=1}^{P} \left\| \mathbf{D}\underline{\alpha}_j - \underline{x}_j \right\|_2^2 \quad \text{s.t.} \quad \forall j, \left\| \underline{\alpha}_j \right\|_p^p \le L$$

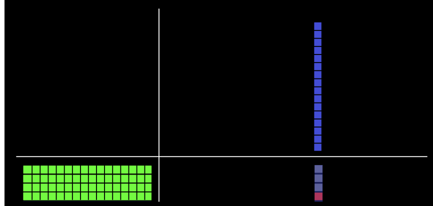
D is known!
For the j^{th} item
we solve

$$\underset{\underline{\alpha}}{\text{Min}} \left\| \mathbf{D}\underline{\alpha} - \underline{x}_j \right\|_2^2 \quad \text{s.t.} \quad \left\| \underline{\alpha} \right\|_p^p \le L$$

Solved by
A Pursuit Algorithm



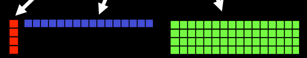
K–SVD: Dictionary Update Stage



Refer only to the examples that use the column $\underline{d}_k$

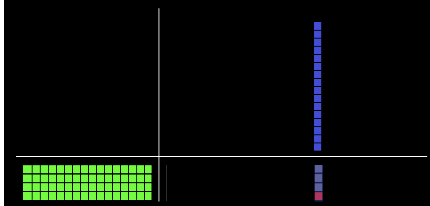
We should solve:

$$\underset{\underline{d}_k, \underline{\alpha}_k}{\text{Min}} \left\| \underline{\alpha}_k \underline{d}_k^T - \mathbf{E} \right\|_F^2$$



Fixing all A and D apart from the k^{th} column, and seek both $\underline{d}_k$ and the k^{th} column in A to better fit the residual!

K–SVD: Dictionary Update Stage

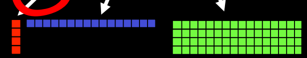


Refer only to the examples that use the column $\underline{d}_k$

We should solve:

$$\underset{\underline{d}_k, \underline{\alpha}_k}{\text{Min}} \left\| \underline{\alpha}_k \underline{d}_k^T - \mathbf{E} \right\|_F^2$$

SVD



Fixing all A and D apart from the k^{th} column, and seek both $\underline{d}_k$ and the k^{th} column in A to better fit the residual!

K–SVD: Algorithm

Task: Find the best dictionary to represent the data samples $\{\mathbf{y}_i\}_{i=1}^N$ as sparse compositions, by solving

$$\min_{\mathbf{D},\mathbf{X}} \left\{\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2\right\} \quad \text{subject to} \quad \forall i,\ \|\mathbf{x}_i\|_0 \le T_0.$$

Initialization : Set the dictionary matrix $\mathbf{D}^{(0)} \in \mathbb{R}^{n\times K}$ with ℓ^2 normalized columns. Set $J = 1$.
Repeat until convergence (stopping rule):

- *Sparse Coding Stage*: Use any pursuit algorithm to compute the representation vectors $\mathbf{x}_i$ for each example $\mathbf{y}_i$, by approximating the solution of

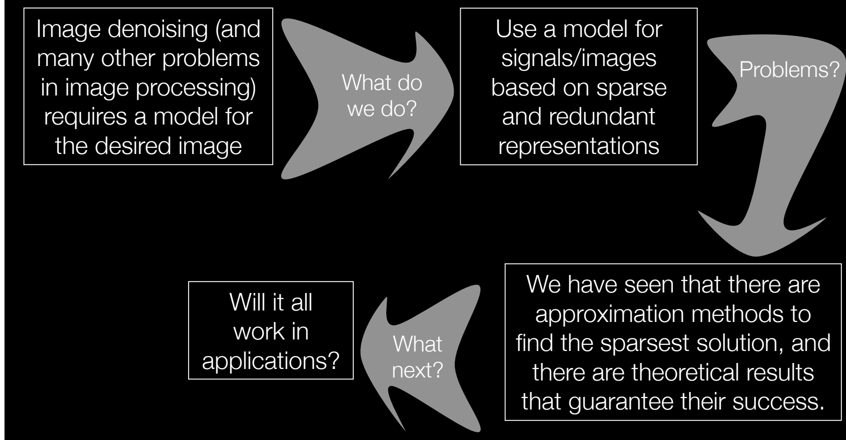
$$i = 1,2,\dots,\ N, \quad \min_{\mathbf{x}_i} \left\{\|\mathbf{y}_i-\mathbf{D}\mathbf{x}_i\|_2^2\right\} \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 \le T_0.$$

- *Codebook Update Stage*: For each column $k = 1,2,\dots,K$ in $\mathbf{D}^{(J-1)}$, update it by
  - Define the group of examples that use this atom, $\omega_k = \{i \mid 1 \le i \le N,\ \mathbf{x}_T^k(i) \ne 0\}$.
  - Compute the overall representation error matrix, $\mathbf{E}_k$, by

$$\mathbf{E}_k = \mathbf{Y} - \sum_{j\ne k} \mathbf{d}_j \mathbf{x}_T^j.$$

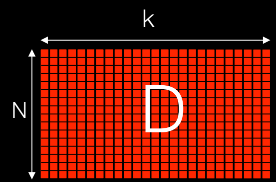
  - Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^R$.
  - Apply SVD decomposition $\mathbf{E}_k^R = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$. Choose the updated dictionary column $\tilde{\mathbf{d}}_k$ to be the first column of $\mathbf{U}$. Update the coefficient vector $\mathbf{x}_R^k$ to be the first column of $\mathbf{V}$ multiplied by $\mathbf{\Delta}(1,1)$.
- Set $J = J + 1$.

To Summarize So Far …



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 become prohibitive.
- So, how should large images be handled?
- The solution: Force shift-invariant sparsity - on each patch of size N-by-N (N=8) in the image, including overlaps.



$$\hat{\underline{x}} = \underset{\underline{x},\{\underline{\alpha}_{ij}\}_{ij}}{\text{ArgMin}} \frac{1}{2}\left\|\underline{x}-\underline{y}\right\|_2^2 + \mu\sum_{ij}\left\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}_{ij}\right\|_2^2 \quad \text{s.t.} \quad \left\|\underline{\alpha}_{ij}\right\|_0^0 \le L$$

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-by-N (overlapping blocks),
- Image of size 1000^2 pixels ➡ ~10^6 examples to use – more than enough.
- This works much better!

K-SVD Image Denoising

$$\hat{\underline{x}} = \underset{\underline{x},\{\underline{\alpha}_{ij}\}_{ij},\ \mathbf{D}?}{\text{ArgMin}} \ \frac{1}{2}\|\underline{x}-\underline{y}\|_2^2 + \mu\sum_{ij}\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}_{ij}\|_2^2 \ \text{ s.t. } \ \|\underline{\alpha}_{ij}\|_0^0 \le L$$

$\underline{x}=\underline{y}$ and $\mathbf{D}$ known | $\underline{x}$ and α_{ij} known | $\mathbf{D}$ and α_{ij} known

Compute α_{ij} per patch

$$\underline{\alpha}_{ij} = \underset{\underline{\alpha}}{\text{Min}}\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}\|_2^2$$

s.t. $\|\underline{\alpha}\|_0^0 \le L$

using the matching pursuit

Compute $\mathbf{D}$ to minimize

$$\underset{\underline{\alpha}}{\text{Min}}\sum_{ij}\|\mathbf{R}_{ij}\underline{x} - \mathbf{D}\underline{\alpha}\|_2^2$$

using SVD, updating one column at a time

K-SVD

Compute $\underline{x}$ by

$$\underline{x} = \left[\mathbf{I} + \mu\sum_{ij}\mathbf{R}_{ij}^T\mathbf{R}_{ij}\right]^{-1}\left[\underline{y} + \mu\sum_{ij}\mathbf{R}_{ij}^T\mathbf{D}\underline{\alpha}_{ij}\right]$$

which is a simple averaging of shifted patches

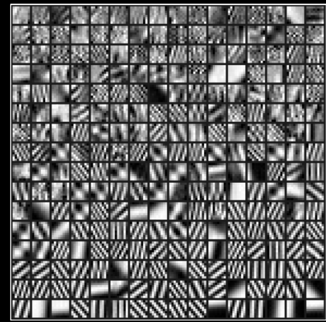
Image Denoising (Gray) [E. & Aharon ('06)]

Source

Result 30.829dB

Noisy image

$\sigma = 20$



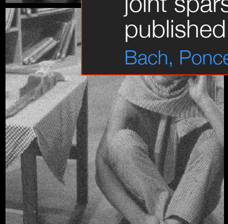
The obtained dictionary after 10 iterations

Image Denoising (Gray) [E. & Aharon ('06)]

Source

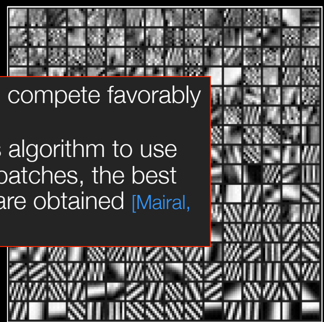
- The results of the K-SVD algorithm compete favorably with the state-of-the-art.
- In a recent work that extended this algorithm to use joint sparse representation on the patches, the best published denoising performance are obtained [Mairal, Bach, Ponce, Sapiro & Zisserman ('09)].

Result 30.829dB



Noisy image

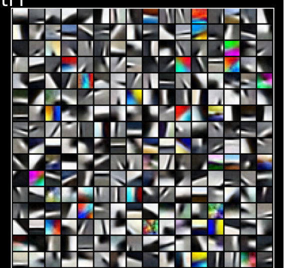
$\sigma = 20$



The obtained dictionary after 10 iterations

Denoising (Color) [Mairal, E. & Sapiro ('08)]

- 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.

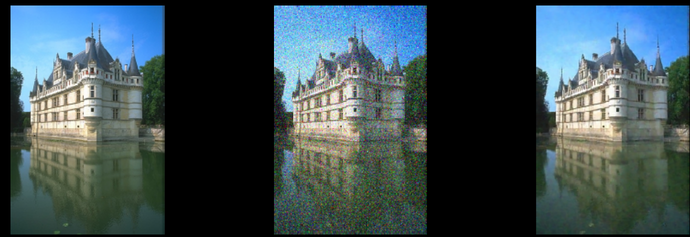


Denoising (Color) [Mairal, E. & Sapiro ('08)]

Original Noisy (20.43dB) Result (30.75dB)

Denoising (Color) [Mairal, E. & Sapiro ('08)]

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)



Original Noisy (12.77dB) Result (29.87dB)

Image Inpainting – The Basics

- Assume: the signal $\underline{x}$ has been created by $\underline{x}=\mathbf{D}\underline{\alpha}_0$ with very sparse $\underline{\alpha}_0$.
- Missing values in $\underline{x}$ imply missing rows in this linear system.
- By removing these rows, we get

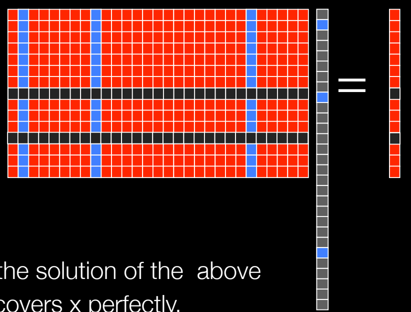
$$\tilde{\mathbf{D}}\underline{\alpha} = \tilde{\underline{x}}$$

- Now solve

$$\min_{\underline{\alpha}} \|\underline{\alpha}\|_0 \quad \text{s.t.} \quad \tilde{\underline{x}} = \tilde{\mathbf{D}}\underline{\alpha}$$

- If $\underline{\alpha}_0$ was sparse enough, it will be the solution of the above problem! Thus, computing $\mathbf{D}\underline{\alpha}_0$ recovers $\underline{x}$ perfectly.

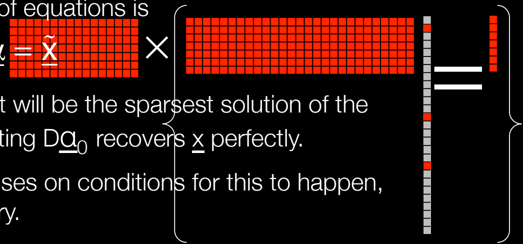
$$\mathbf{D}\ \underline{\alpha}_0 = \underline{x}$$



Side Note: Compressed-Sensing

- Compressed Sensing is leaning on the very same principal, leading to alternative sampling theorems.
- Assume: the signal $\underline{x}$ has been created by $\underline{x}=\mathbf{D}\underline{\alpha}_0$ with very sparse $\underline{\alpha}_0$.
- Multiply this set of equations by the matrix Q which reduces the number of rows.
- The new, smaller, system of equations is

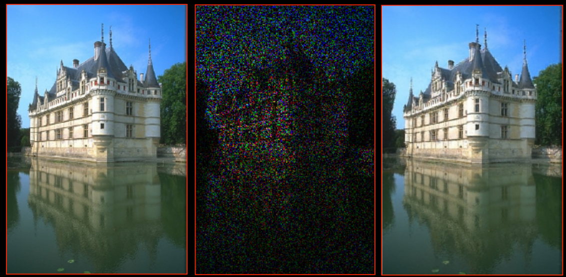
$$\mathbf{QD}\underline{\alpha} = \mathbf{Q}\underline{x} \ \Rightarrow \ \tilde{\mathbf{D}}\underline{\alpha} = \tilde{\underline{x}}$$



- If $\underline{\alpha}_0$ was sparse enough, it will be the sparsest solution of the new system, thus, computing $\mathbf{D}\underline{\alpha}_0$ recovers $\underline{x}$ perfectly.
- Compressed sensing focuses on conditions for this to happen, guaranteeing such recovery.

Inpainting [Mairal, E. & Sapiro ('08)]

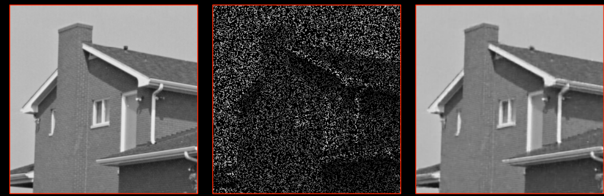
Experiments lead to state-of-the-art inpainting results.



Original 80% missing Result

Inpainting [Mairal, E. & Sapiro ('08)]

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



Original 80% missing Result

Inpainting [Mairal, E. & Sapiro ('08)]

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. ('05)].

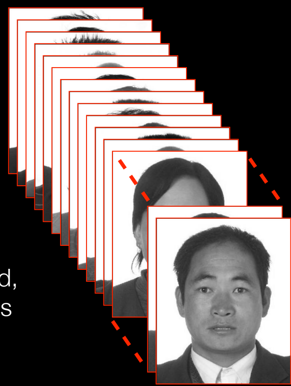


Image Compression

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

On the training set

On the test image

Training set (2500 images)

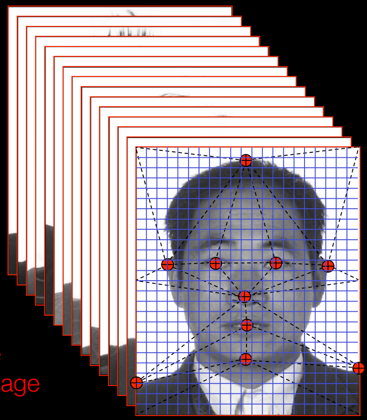


Image Compression Results

Original

JPEG

JPEG-2000

Local-PCA

K-SVD

Results for 820 Bytes per each file

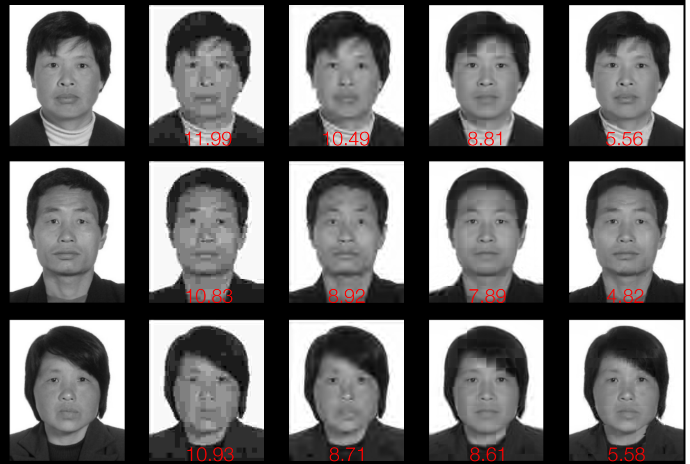


Image Compression Results

Original

JPEG

JPEG-2000

Local-PCA

K-SVD

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)]

550 bytes K-SVD results with and without deblocking

K-SVD (6.60) K-SVD (5.49) K-SVD (6.45) K-SVD (11.67)

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

Super-Resolution [Zeyde, Protter, & 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)

This book is about *convex optimization*, a special class of mathematical optimiza-tion problems, which includes least-squares and linear programming problems. It is well known that least-squares and linear programming problems have a fairly complete theory, arise in a variety of applications, and can be solved numerically very efficiently. The basic point of this book is that the same can be said for the larger class of convex optimization problems.

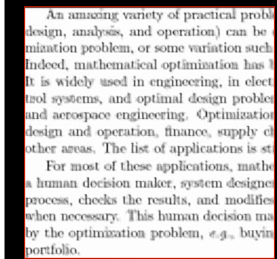
While the mathematics of convex optimization has been studied for about a century, several related recent developments have stimulated new interest in the topic. The first is the recognition that interior-point methods, developed in the 1980s to solve linear programming problems, can be used to solve convex optimiza-tion problems as well. These new methods allow us to solve certain new classes of convex optimization problems, such as semidefinite programs and second-order cone programs, almost as easily as linear programs.

The second development is the discovery that convex optimization problems (beyond least-squares and linear programs) are more prevalent in practice than was previously thought. Since 1990 many applications have been discovered in areas such as automatic control systems, estimation and signal processing, com-munications and networks, electronic circuit design, data analysis and modeling, statistics, and finance. Convex optimization has also found wide application in com-binatorial optimization and global optimization, where it is used to find bounds on the optimal value, as well as approximate solutions. We believe that many other applications of convex optimization are still waiting to be discovered.

There are great advantages to recognizing or formulating a problem as a convex optimization problem. The most basic advantage is that the problem can then be solved, very reliably and efficiently, using interior-point methods or other special methods for convex optimization. These solution methods are reliable enough to be embedded in a computer-aided design or analysis tool, or even a real-time reactive or automatic control system. There are also theoretical or conceptual advantages of formulating a problem as a convex optimization problem. The associated dual

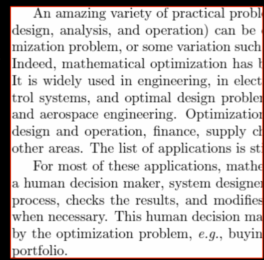
The training image: 717×717 pixels, providing a set of 54,289 training patch-pairs.

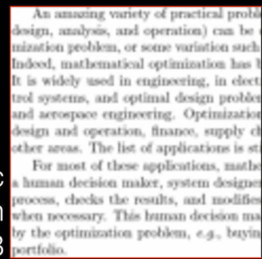
Super-Resolution – Results (1)



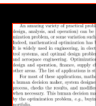
SR Result PSNR=16.95dB

Ideal Image





Bicubic interpolation PSNR=14.68dB



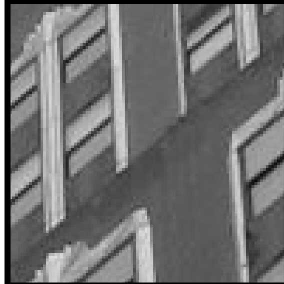
Given Image

Super-Resolution – Results (2)

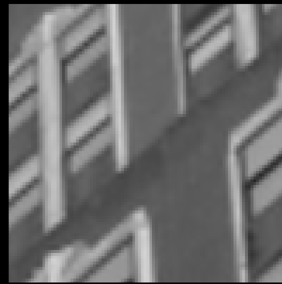
Given image

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

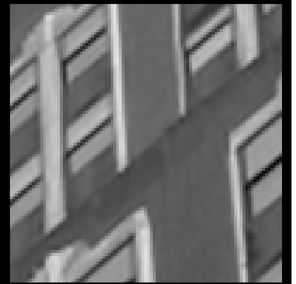
Super-Resolution – Results (2)



The Original



Bicubic Interpolation



SR result

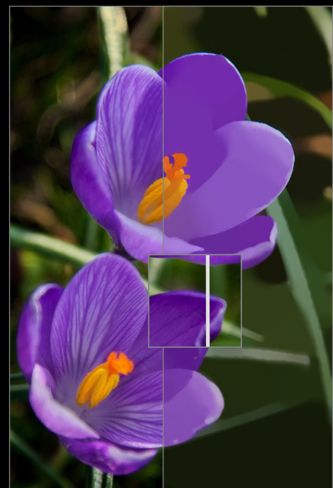
Super-Resolution – Results (2)

The Original

Bicubic Interpolation

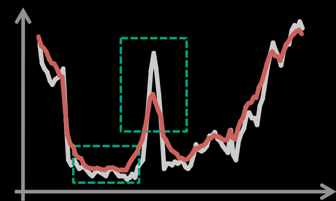
SR result

L0-Image Smoothing



General goals:

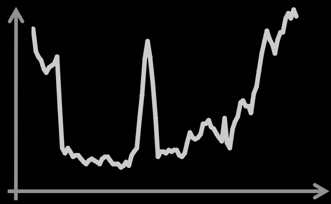
- Suppress insignificant details
- Maintain major edges



L0-Smoothing Method

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

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

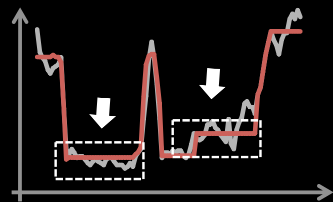


which corresponds to the L0-norm of gradient

Two Features

1. Flattening insignificant details

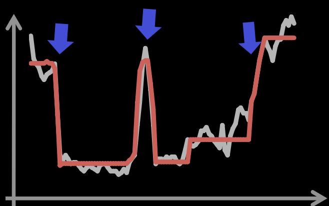
By removing small non-zero gradients



Two Features

2. Enhancing prominent edges

Because large gradients receive the same penalty as small ones



$$\#\{p \mid |\nabla f_p| \neq 0\} = \#\{p \mid |\alpha \nabla f_p| \neq 0\}$$

Our Framework in 1D

- Constrain # of non-zero gradients

$$c(f) = \#\{p \mid |f_p - f_{p+1}| \neq 0\} = k$$

- Make the result similar to the input

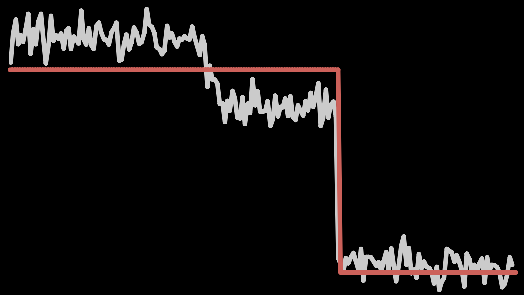
$$\min_f \sum_p (f_p - g_p)^2$$

- Objective function

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

Our Framework in 1D

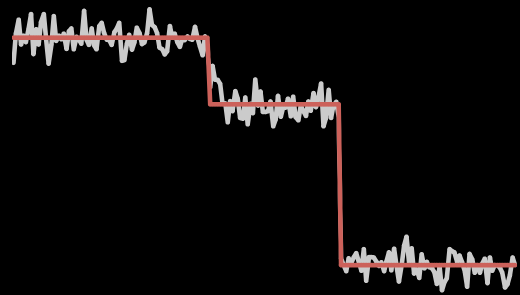
- Input 1D signal g



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

Our Framework in 1D

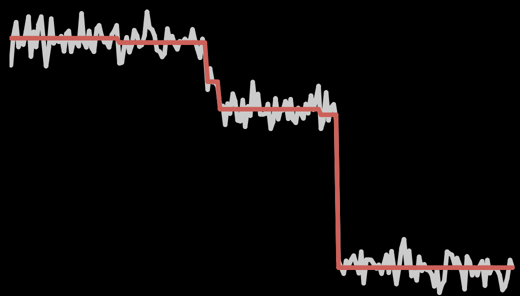
- Input 1D signal g



$$\min_f \sum_p (f_p - g_p)^2 \quad s.t. \quad c(f) = 2$$

Our Framework in 1D

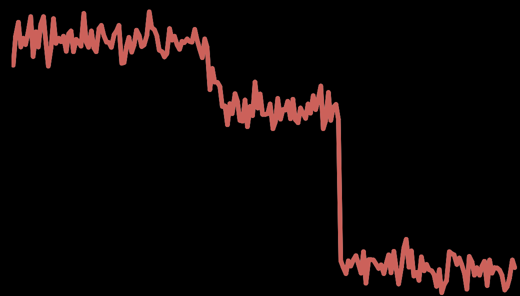
- Input 1D signal g



$$\min_f \sum_p (f_p - g_p)^2 \quad s.t. \quad c(f) = 5$$

Our Framework in 1D

- Input 1D signal g



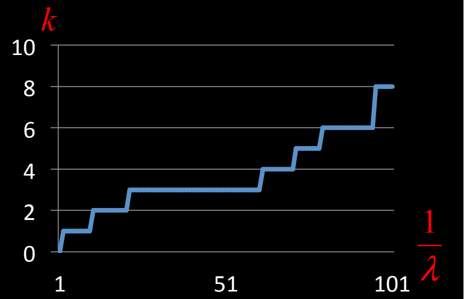
$$\min_f \sum_p (f_p - g_p)^2 \quad s.t. \quad c(f) = 200$$

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

$$\min_f \sum_p (f_p - g_p)^2 + \lambda \cdot c(\quad \boldsymbol{h}, \boldsymbol{v} \quad)$$

$$+\beta \cdot \sum_p \left((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right)$$

Separately estimate f and (h, v)

Iterative Optimization

- Compute f given h, v

$$E(f) = \sum (f_p - g_p)^2 + \beta \cdot \left((\partial_x f_p - h_p)^2 + (\partial_y f_p - v_p)^2 \right)$$

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

Iteration #

Smoothing Strength

Input

Smoothing Strength

λ=0.01

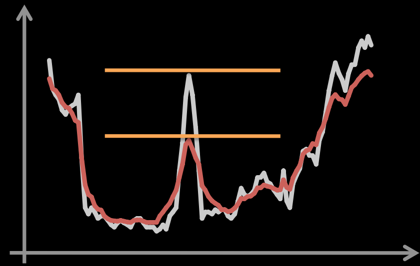
Smoothing Strength

λ=0.02

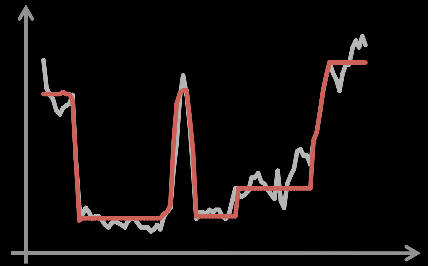
Smoothing Strength

λ=0.03

Comparison



Total Variation



L0 Smoothing