Compressive Sensing in Video Encoding

Brian Azizi

Laboratory for Scientific Computing, University of Cambridge

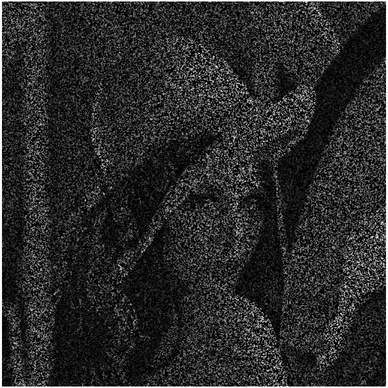
May 18, 2016

Thanks to: Dr Anita Faul (Supervisor)
Dr Nikos Nikiforakis
Georgios Pilikos

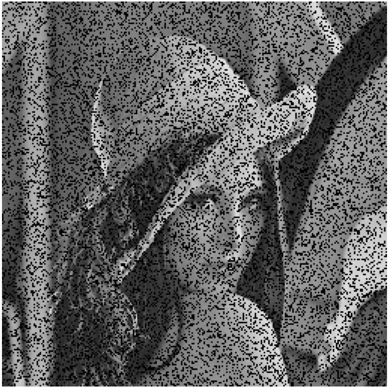
Outline

- Project Outcome & Demonstration
- Theoretical Background
- Further Work
- More demos of results
- Bibliography

2D Demonstration



Corrupted Image (70% missing)



Scale 1 Reconstruction

See 2014 MPhil Thesis by Georgios Pilikos

2D Demonstration

Cascade to scale 2

Cascade to scale 3

See 2014 MPhil Thesis by Georgios Pilikos

3D Demonstration

Current prototype:

Corrupted Video (60% missing)

Scale 1 Reconstruction

3D Demonstration

- No cascade of reconstructions yet - ask me later this week
- For reference, the original video:

Theoretical Background

- Three building blocks:
  - *Compressive Sensing* - Novel signal processing framework allowing for near-perfect reconstruction of heavily under-sampled signals
  - *Discrete Wavelet Transforms* - Required for obtaining sparse representations of the signals
  - *Relevance Vector Machine* - Machine Learning Algorithm for performing the reconstruction via regression

Compressive Sensing: Reconstruction

- General idea: Reconstruct signal $\boldsymbol{x} \in \mathbb{R}^N$ from measurements $\boldsymbol{y} \in \mathbb{R}^M$ where $M << N$
- Corresponds to solving under-determined linear system $\Omega \boldsymbol{x} = \boldsymbol{y}$, $\Omega \in \mathbb{R}^{M \times N}$
- In general, shouldn't be possible by fundamental theorem of Linear Algebra ("as many equations as unknowns")

- However, for a certain class of signals it is possible to get perfect reconstruction. Namely if
  1. $\boldsymbol{x}$ is *sparse* (i.e. most elements are zero), or
  2. it is possible to change basis so that the transformed signal $\boldsymbol{w} = \Psi^T \boldsymbol{x}$ is sparse
- So in the later case: $\boldsymbol{y} = \Omega \boldsymbol{x} = \Omega \Psi \boldsymbol{w} \equiv \Phi \boldsymbol{w}$
- We know $\boldsymbol{y} \in \mathbb{R}^M$ and $\Phi \in \mathbb{R}^{M \times N}$ and want $\boldsymbol{w} \in \mathbb{R}^N$

Compressive Sensing: Deterministic Methods

- Ideally, we want $\hat{\boldsymbol{w}} = \arg\min ||\boldsymbol{w}'||_0$ such that $\Phi \boldsymbol{w}' = \boldsymbol{y}$
- $||\boldsymbol{v}||_0$ is the L_0 *norm* $=$ number of non-zeros entries of $\boldsymbol{v}$
- This would give the sparsest solution

- Ideally, we want $\hat{\boldsymbol{w}} = \arg\min ||\boldsymbol{w}'||_0$ such that $\Phi \boldsymbol{w}' = \boldsymbol{y}$
- $||\boldsymbol{v}||_0$ is the L_0 *norm* $=$ number of non-zeros entries of $\boldsymbol{v}$
- This would give the sparsest solution
- But it turns out to be NP-complete - hence not feasible

Compressive Sensing: Deterministic Methods

- Ideally, we want $\hat{\boldsymbol{w}} = \arg\min ||\boldsymbol{w}'||_0$ such that $\Phi \boldsymbol{w}' = \boldsymbol{y}$
- $||\boldsymbol{v}||_0$ is the L_0 *norm* = number of non-zeros entries of $\boldsymbol{v}$
- This would give the sparsest solution
- But it turns out to be NP-complete - hence not feasible
- Recently[1], there's been much success by doing L_1 instead of L_0 minimization: $\hat{\boldsymbol{w}} = \arg\min ||\boldsymbol{w}'||_1$ such that $\Phi \boldsymbol{w}' = \boldsymbol{y}$
- $||\boldsymbol{v}||_1 = \sum_{i=1}^{N} |v_i|$ is the L_1 norm

- Ideally, we want $\hat{\boldsymbol{w}} = \arg\min ||\boldsymbol{w}'||_0$ such that $\Phi\boldsymbol{w}' = \boldsymbol{y}$
- $||\boldsymbol{v}||_0$ is the L_0 *norm* = number of non-zeros entries of $\boldsymbol{v}$
- This would give the sparsest solution
- But it turns out to be NP-complete - hence not feasible
- Recently[1], there's been much success by doing L_1 instead of L_0 minimization: $\hat{\boldsymbol{w}} = \arg\min ||\boldsymbol{w}'||_1$ such that $\Phi\boldsymbol{w}' = \boldsymbol{y}$
- $||\boldsymbol{v}||_1 = \sum_{i=1}^{N} |v_i|$ is the L_1 norm
- These deterministic approaches are not relevant for us at the moment - we will use the RVM

Compressive Sensing: Acquisition

- Beside reconstruction, a large part of Compressive Sensing is concerned with efficient *acquisition* of signals, i.e. how to measure $\boldsymbol{y}$ directly without measuring the entirety of $\boldsymbol{x}$ first
- Not relevant for us at this stage
- Instead, we simulate $\boldsymbol{y}$ by taking $\boldsymbol{x}$ and randomly deleting a certain percentage of its entries
- E.g. for 2D signals: $\boldsymbol{x} =$  and $\boldsymbol{y} =$ 
- Then attempt to reconstruct $\boldsymbol{x}$ from $\boldsymbol{y}$

Vectorized Signals

- Technical Aside: I talk somewhat interchangibly about 2D signals (e.g. images) and 3D signals (e.g. videos)
- Under the hood, we actually need to store our signals as 1-dimensional vectors
- Reason: RVM operates on vectors
- For images, we stack the columns on top of each other to form one long vector in $\mathbb{R}^{hw}$ (height times width)
- For video, we stack the columns in each frame and then stack the frames ($\Rightarrow \boldsymbol{x} \in \mathbb{R}^{hwf}$)

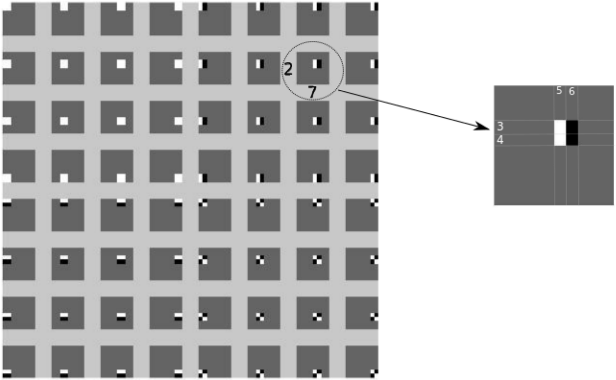
Sparse Representations

- It is possible to obtain sparse representations for a large variety of signals
- Common example in signal processing: Transforming sounds from time domain to frequency domain via (Discrete) Fourier Transform
- For natural images and image sequences (videos), there are two widely used transforms:
  - Discrete Haar Wavelet Transform
  - Discrete Cosine Transform
- There exist more transforms and probably better ones
- Ideally, we would like the data to tell us which one to use via some kind of Deep Learning - but this is getting off-topic

2D Haar Wavelets

- Current prototype uses Haar wavelets at the first scale
- Very simple: transform essentially consists of taking averages and differences of patches of pixels
- At scale 1, the individual basis functions have support over 2×2 patches

- 2D Haar wavelets at scale 1 can be visualized by



- White $\equiv +\frac{1}{2}$, black $\equiv -\frac{1}{2}$, dark gray $\equiv 0$
- So for example $x_{2,7} \to w_{2,7} = \frac{1}{2}(x_{3,5} - x_{3,6} + x_{4,5} - x_{4,6})$

2D Haar Wavelets: Example

- Haar wavelet transform of

$x =$  is $w =$ 

- Most entries in w are zero or very close to zero (shown as black) $\Rightarrow$ w is sparse

2D Haar Wavelets: Example

- Somewhat easier to see what's going on if we invert and translate the colours a little bit:

- Top left is an average of $\boldsymbol{x}$, remaining entries in $\boldsymbol{w}$ are detail coefficients that capture horizontal, verical and diagonal *edges* in $\boldsymbol{x}$

3D Haar Wavelets

- Think of a video as a volume created by stacking the individual frames
- To take the Haar wavelet transform of a video $\boldsymbol{x}$ we have two approaches:
  - Take the 2D transform of each frame individually $\rightarrow$ process $\boldsymbol{x}$ as a sequence of independent images
  - Use 3D Haar wavelets and process video as a volume $\rightarrow$ exploits continuity between frames
- We used the 3D wavelets approach because it is more general
- 3D Haar wavelets have support over $2^j \times 2^j \times 2^j$ blocks (at the jth scale)

Demo of Haar wavelet transform of soccer.yuv

$\boldsymbol{x} =$

(30 frames per second)

$\boldsymbol{w} =$

(20 frames per second)

Bayesian Compressive Sensing

- Recall the goal of Compressive Sensing: Find sparsest solution to $\Phi \boldsymbol{w} = \boldsymbol{y}$
- We mentioned deterministic methods (L_1 minimization)
- But we will use a probabilistic method by treating it as a Machine Learning problem
- Specifically, we use the RVM to do a regression:
  1. Input a *target vector* $\boldsymbol{y} \in \mathbb{R}^M$
  2. Input a *design matrix* $\Phi \in \mathbb{R}^{M \times N}$
  3. The RVM outputs a *sparse* coefficients vector $\boldsymbol{w}^* \in \mathbb{R}^N$

Bayesian Compressive Sensing

- More specifically, RVM gives us a *posterior distribution* for $\boldsymbol{w}^*$: $\boldsymbol{w}^* | \, \text{data} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$
- The *posterior mean* $\boldsymbol{\mu}$ is usually very sparse
- Reconstruct the original signal as $\boldsymbol{x}^* = \Psi \boldsymbol{\mu}$
- Even better: we can use Σ to get error bars in our reconstruction
- We can use error bars to create a cascade of reconstructions and boost the quality
- Focus for next couple of days

The Relevance Vector Machine: Theory

- For our purposes, it is okay to treat the RVM as a "black box"
- However, if interested in theory:
- The RVM [2] is a Bayesian Machine Learning algorithm
- It was developed by Mike Tipping[2] and later improved upon by him and Anita Faul[3]

RVM Theory

- It models the data as $p(y_i|\,\boldsymbol{x},\Phi) = \mathcal{N}(y_i|\,\boldsymbol{w}^T\boldsymbol{\phi}(x_i),\sigma^2)$
- Bayesian, so put a prior on $\boldsymbol{w}$: $p(\boldsymbol{w}) = \prod_{j=1}^{N}\mathcal{N}(w_i|\,0,\alpha_j^{-1})$
- Use training data (observations) to obtain a posterior for $\boldsymbol{w}$: $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\,\boldsymbol{\mu},\Sigma)$
- During training, many α_j become infinite $\Rightarrow w_j = 0$ with infinite precision
- Thus $\boldsymbol{w}$ is *sparse*

Looking Ahead

- Progress so far:
  - Implemented a working prototype
  - Use Haar Wavelet transform
  - Got some initial results
- Still to do:
  - Multi-scale Cascade of RVMs
  - Try more Basis Functions, in particular the Discrete Cosine Transform
  - Study: When does it work well, when not?
  - Lit review: How does it compare to other existing methods?
  - How can it be improved?

Outro

Questions?

More demos

Flickering lines (60% missing data)

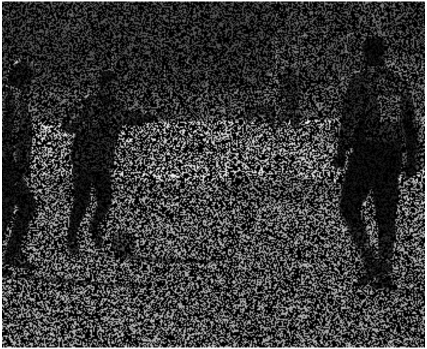
$\boldsymbol{y} =$

$\boldsymbol{x}^* =$

More demos

Uniform noise (60% missing data)

$\boldsymbol{y} =$



$\boldsymbol{x}^* =$

Selected References

- E. J. Candes, T. Tao, Decoding by linear programming, Information Theory, IEEE Transactions on 51 (12) (2005) 4203–4215.
- M. E. Tipping, Sparse bayesian learning and the relevance vector machine, The journal of machine learning research 1 (2001) 211–244.
- M. E. Tipping, A. C. Faul, et al., Fast marginal likelihood maximisation for sparse bayesian models., in: AISTATS, 2003.