COMPRESSED SENSING IN PYTHON

Sercan Yıldız
syildiz@samsi.info

February 27, 2017
Outline

A Brief Introduction to Compressed Sensing

A Brief Introduction to CVXOPT

Examples
A Brief Introduction to Compressed Sensing
Signal Processing and Compression

- Signals over time and/or space
- Often, signals are sparse in an appropriate domain.

- Basic idea behind lossy sound/image compression:
  - Transform signal to the frequency domain.
  - Keep frequencies with the largest magnitudes, discard the rest.
  - Examples: MP3, JPEG, MPEG
**Sampling**

- Sampling at discrete points:
  - Reduces a continuous-time (analog) signal to a discrete-time (digital) signal
  - If the signal is bandlimited (bounded frequency) and the sampling rate is large enough, the signal can be exactly recovered:

**Theorem (Nyquist-Shannon sampling theorem)**

*A bandlimited signal can be exactly reconstructed from its samples if the sampling rate is more than twice the maximum frequency in the signal.*

- Compressed sensing combines compression and sampling with efficient sampling protocols that capture and condense the information content in a sparse signal into a small amount of data.
Compressed Sensing (CS)

- Perfect reconstruction can be possible at sub-Nyquist sampling rates if additional information about the signal (such as sparsity) is available.
- Why are low sampling rates attractive?
  - Number of sensors may be limited.
  - Measurements may be expensive.
  - Sensing process may be slow.
- Reconstruction of undersampled signal requires optimization!
- In the remainder, we consider discrete signals of finite length...and make things a little more concrete.
Compressed Sensing (CS)

- Given: Signal \( f \) with the sparse representation \( f = \Psi x \) for some square unitary matrix \( \Psi \)
- We would like to design an \( n \times m \) sensing matrix \( \hat{\Phi} \) (for \( m << n \)) that captures as much information about \( f \) as possible. The matrix \( \hat{\Phi} \) will generate the observations
  \[
  y = \hat{\Phi}^\top f.
  \]

- Questions:
  - How do we design \( \hat{\Phi} \)?
  - How do we reconstruct \( f \) from \( y \)?
A key property of good sensing matrices is their incoherence with $\Psi$.

The coherence of two orthogonal matrices $\Phi$ and $\Psi$ is

$$\mu(\Phi, \Psi) = \sqrt{n} \max_{j,k} \Phi_j^\top \Psi_k.$$  

Coherence measures the largest correlation between any two elements of $\Phi$ and $\Psi$.

Examples of low coherence pairs:
- $\Phi$: identity matrix, $\Psi$: Fourier matrix
- $\Phi$: random orthogonal matrix, $\Psi$: fixed orthogonal matrix (whp)

Given a low coherence pair $(\Phi, \Psi)$, we choose the sensing matrix $\hat{\Phi}$ as an $n \times m$ column submatrix of $\Phi$. 

Sensing Matrices
Let $A = \hat{\Phi}^\top \Psi$.

To recover the signal $x$ in the frequency domain, we solve the “basis pursuit” problem

$$\min_x \|x\|_1 \quad \text{subject to} \quad Ax = y.$$  

- $\ell_1$-norm promotes sparsity.

- How can we attack basis pursuit?
  - Reformulate as a linear program:
    $$\min_{x,u} \sum_j u_j \quad \text{subject to} \quad Ax = y, \quad -u \leq x \leq u.$$  
  - Reformulate as an $\ell_1$-regularized least squares problem:
    $$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1.$$  
  - Special-purpose algorithms
When does CS work?

**Theorem (Candès and Romberg, 2007)**

Suppose the true signal $\tilde{x}$ is $s$-sparse. Let $\hat{\Phi}$ consist of $m$ uniformly random measurements from $\Phi$. If

$$m \geq C\mu(\Phi, \Psi)s \log n$$

for some constant $C > 0$, then the basis pursuit problem recovers $\tilde{x}$ with high probability.

- Taking $m \geq 4s$ seems to work well in practice.
- Signals are not always exactly sparse: Many coefficients of the true signal $\tilde{x}$ may be small but not zero.
**When does CS work?**

**Definition (Candés and Tao, 2005)**

For each integer $s = 1, 2, \ldots$, the *isometry constant* $\delta_s$ of a matrix $A$ is the smallest number such that

$$(1 - \delta_s)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_s)\|x\|_2^2$$

holds for all $s$-sparse vectors $x$. We will say that $A$ has the *restricted isometry property (RIP)* of order $s$ if $\delta_s$ is “small.”

- How is RIP useful?
  - If $\delta_{2s} \approx 0$, then $\|A(x_1 - x_2)\|_2^2 \approx \|x_1 - x_2\|_2^2$ for all $s$-sparse vectors $x_1, x_2$.
  - In other words, $x_1$ and $x_2$ remain distinguishable even after left-multiplication with $A$. 
**When does CS work?**

**Theorem (Candés, Romberg, and Tao, 2006)**

Assume $\delta_{2s} < \sqrt{2} - 1$. Let $\bar{x}$ be the true signal, and let $\bar{x}_s$ be the $s$-sparse vector consisting of the $s$ largest (in absolute value) entries of $\bar{x}$. Then the solution $x^*$ to the basis pursuit problem satisfies

$$
\|x^* - \bar{x}\|_2 \leq C' \|\bar{x} - \bar{x}_s\|_1 / \sqrt{s} \quad \text{and} \quad \|x^* - \bar{x}\|_1 \leq C' \|\bar{x} - \bar{x}_s\|_1
$$

for some constant $C' > 0$.

- **Why is this surprising?**
  - In a traditional compression scheme, we would sample $f$, calculate the transform coefficients $\bar{x}$, and compress $\bar{x}$ into $\bar{x}_s$.
  - If $A$ satisfies the assumption, using CS techniques we can compute a solution $x^*$ of quality close to $\bar{x}_s$ from the sample $y$ only.

- **Where do we find matrices $A$ that satisfy the RIP?**
  - There are several randomized schemes for sampling matrices $A$ that satisfy the RIP whp.
  - Given a low-coherence pair $(\Phi, \Psi)$, one can sample $m$ columns of $\Phi$ uniformly at random. If $m \geq Cs(\log n)^4$, $A = \hat{\Phi}^\top \Psi$ satisfies the RIP whp.
Interested in More?

- **Surveys:**

- **Textbooks:**

---

For more information on compressed sensing, consider the following books:

- *Compressed Sensing: Theory and Applications* by Yonina C. Eldar and Gitta Kutyniok
- *Sampling Theory: Beyond Bandlimited Systems* by Yonina C. Eldar
- *A Mathematical Introduction to Compressive Sensing* by Simon Foucart and Holger Rauhut

_images sources:
- [Compressed Sensing](https://images-na.ssl-images-amazon.com/images/I/514fK1MG-9L_.SX327_BO1,204,203,200_.jpg)
- [A Mathematical Introduction to Compressive Sensing](https://images-na.ssl-images-amazon.com/images/I/41s%2B850qu0L_.SX331_BO1,204,203,200_.jpg)
Interested in More?

- Other key references:
A Brief Introduction to CVXOPT
Introduction to CVXOPT

- CVXOPT is a free convex optimization package for Python.
- It can be used with iPython or on the command line by executing Python scripts.
- It provides built-in solvers for
  - linear cone programs: `cvxopt.solvers.conelp`
  - quadratic cone programs: `cvxopt.solvers.coneqp`
  - convex programs with linear objectives: `cvxopt.solvers.cpl`
  - convex programs with nonlinear objectives: `cvxopt.solvers.cp`
- It provides routines for implementing customized solvers and interfaces to external solvers (GLPK, MOSEK, and DSDP5).
- It also provides the module `cvxopt.modeling` for modeling and solving linear programs and optimization problems with convex piecewise-linear cost and constraint functions.
Essentials for Today

- **cvxopt.matrix and cvxopt.spmatrix**
  - CVXOPT extends the built-in Python objects with a `cvxopt.matrix` object for dense matrices and an `cvxopt.spmatrix` object for sparse matrices.
  - To enter a problem in matrix form into CVXOPT, data must be provided using one of these matrix objects.
  - NumPy arrays can be converted to CVXOPT matrices.

- **cvxopt.modeling**
  - Use `cvxopt.modeling.variable` to define (a vector of) variables.
  - Affine and convex piecewise-linear functions can be created with compositions of linear expressions, `max`, and `abs`.
  - Use `cvxopt.modeling.op` to create an optimization problem.
  - Call the method `cvxopt.modeling.op.solve` to solve the optimization problem: This method converts the problem to a linear program and solves it using the CVXOPT linear programming solver.


**Discrete Fourier Transform (DFT)**

- Let $\Psi \in \mathbb{C}^{n \times n}$ be the square unitary matrix

\[
\Psi = \begin{bmatrix}
    \vdots & \vdots & \vdots & \vdots \\
    \frac{1}{\sqrt{n}} e^{i 2\pi p/n} & \frac{1}{\sqrt{n}} e^{i 4\pi p/n} & \cdots & \frac{1}{\sqrt{n}} e^{i 2\pi p} \\
    \vdots & \vdots & \ddots & \vdots \\
\end{bmatrix}
\]

- Recall Euler’s identity? $e^{ix} = \cos x + i \sin x$.

- For one-dimensional signals:
  - (Orthonormalized) DFT: $f \rightarrow \Psi^\top f$
  - (Orthonormalized) Inverse DFT: $x \rightarrow \Psi x$

- For two-dimensional signals:
  - DFT/IDFT acts on the rows first and columns later.
**Discrete Cosine Transform (DCT)**

- Similar to the DFT but real-valued
- Let $\Psi \in \mathbb{R}^{n \times n}$ be the square orthogonal matrix

$$
\Psi = \begin{bmatrix}
\vdots & \vdots & \vdots \\
\frac{1}{\sqrt{n}} & \frac{1}{\sqrt{2n}} & \cos \left( \frac{\pi (2p+1)}{2n} \right) & \cdots & \frac{1}{\sqrt{2n}} & \cos \left( \frac{\pi (2p+1)(n-1)}{2n} \right) \\
\vdots & \vdots & \vdots
\end{bmatrix}
$$

- For one-dimensional signals:
  - (Orthonormalized) DCT (of type II): $f \rightarrow \Psi^\top f$
  - (Orthonormalized) Inverse DCT (of type II): $x \rightarrow \Psi x$
- For two-dimensional signals:
  - DCT/IDCT acts on the rows first and columns later.
- DFT/DCT represents a signal as a sum of sinusoids of varying magnitudes and frequencies.
- For a “typical” sound/image signal, the sample data is correlated, and the DFT/DCT is sparse: Most of the information is concentrated in just a few coefficients of $x = \Psi^\top f$. 
EXAMPLE #1: SOUND SENSING

- Example #1.a: Artificial sound wave
Example #1: Sound Sensing

- Example #1.a: Artificial sound wave
  - Percentage sampled: 10%

![True Signal in the Time Domain](image1)

![Signal in the Frequency Domain](image2)

![True and Recovered Signals in the Time Domain](image3)
EXAMPLE #1: SOUND SENSING

- Example #1.a: Artificial sound wave
  - Percentage sampled: 10%

- Example #1.b: Real sound wave
Example #1: Sound Sensing

- Example #1.a: Artificial sound wave
  - Percentage sampled: 10%

- Example #1.b: Real sound wave
  - Percentage sampled: 20%
Example #2: Image Sensing

- Example #2.a: SAMSI
Example #2: Image Sensing

- Example #2.a: SAMSI
  - Percentage sampled: 25%
**Example #2: Image Sensing**

- Example #2.a: SAMSI
  - Percentage sampled: 25%

- Example #2.b: Boat
Example #2: Image Sensing

- Example #2.a: SAMSI
  - Percentage sampled: 25%

- Example #2.b: Boat
  - Percentage sampled: 25%
Questions?