Midterm Progress Report

Stephen Pinto

May 17, 2015

Background

Perceptual Audio Compression is a generic term for how dozens of prevalent audio codecs (e.g. MPEG and AAC) achieve impressive compression ratios for audio files. In a sentence, it is any lossy compression algorithm which seeks to compress audio signals at the expense of perfect reconstruction while maintaining good fidelity by human hearing standards. The algorithms seek to place quantization noise inherent in compression into spectral regions that are difficult for humans to hear. Developed in the early 90s, perceptual compression has since been much improved and become ubiquitous. For an in-depth treatment on the basics of a perceptual audio codec, see [BG03].

Compressive Sampling (CS) is a computational technique for data compression and reconstruction which centers around a convex optimization problem. Under certain conditions (thoroughly discussed in a series of papers by Candes and Tao among others and summarized in [CW08]), a signal that is sparse in some domain can be perfectly reconstructed from a compressed set of samples. CS is perhaps most well known for its application in the field of medical imaging.

CS has seen minimal applications in the world of audio applications ([DMD+13] being one great example for the purposes of declipping audio signals) and even fewer still to audio compression specifically (I have found none.) This proposed project will apply CS to a simple perceptual audio codec. Achieving compression ratios as high as cutting edge audio codecs is unreasonable, but achieving modest compression ratios with this novel technique – at least to the world of audio compression – is an achievable goal.

Problem Statement

The codec takes in a signal and splits it into overlapping blocks of a constant length, typically 512 or 1024 samples. Each block is encoded and subsequently decoded independently. As such, the following description of the encoding and decoding stages is applied to each individual block one by one.

The encoding half of the codec takes in a signal f of length n (the block) and outputs a signal y of length $m \ll n$, calculated as

$$y = \Phi H f.$$

The sensing basis, $\Phi \in \mathbb{R}^{m \times n}$, has rows randomly selected from the standard basis of $\mathbb{R}^n$. The perceptual weighting matrix, $H \in \mathbb{R}^{n \times n}$, is a circulant matrix diagonalized by the Discrete Fourier Transform matrix with eigenvalues equal to the inverse of the signal's masking threshold. This masking threshold indicates the necessary magnitude a signal must contain at a certain frequency for humans to hear it. The interested reader is again referred to [BG03] for details on calculating the masking threshold.

Candes et al. present a reconstruction method in [CWB08], adapted here, that makes the reconstruction of a single block an iterative process. Each iteration solves the convex optimization problem

$$\begin{aligned} \hat{x}^{(i)} = \underset{x}{\text{argmin}} \quad & ||W^{(i)}x||_1 \\ \text{subject to} \quad & ||y - \Phi H^{(i)} \Psi x||_2^2 \leq \kappa. \end{aligned} \tag{1}$$

where

$$W^{(i+1)} = \text{diag}\left(\frac{1}{|\hat{x}_1^{(i)}| + \delta}, ..., \frac{1}{|\hat{x}_n^{(i)}| + \delta} \right)$$

for some small constant δ and $H^{(i+1)}$ has eigenvalues equal to the inverse of the masking threshold of $x^{(i)}$. This iteration continues until $\hat{x}$ converges (i.e. $||\hat{x}^{(i)} - \hat{x}^{(i-1)}||_2 \leq \gamma$ for some γ - this typically takes less than 4 iterations.) The purpose of this project is to create a solver for a single iteration of the aforementioned heuristic method from [CWB08] that takes advantage of the structure in the convex optimization problem 1. Since this solver considers only one iteration, the iteration index superscripts will be dropped from here on out.

Problem 1 is equivalent to the QCLP (Quadratically Constrained Linear Program)

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^T s \\ \text{subject to} \quad & Wx - s \preceq 0, -Wx - s \preceq 0 \\ & y^T y - 2y^T \Phi H \Psi x + x^T \left(\Phi H \Psi\right)^T \left(\Phi H \Psi\right) x - \kappa \leq 0. \end{aligned}$$

This problem has two variables, each in $\mathbf{R}^n$, meaning we can bring together into one variable, $z \in \mathbf{R}^{2n}$, which makes the equivalent problem

$$\begin{aligned} \hat{z} = \underset{z}{\text{argmin}} \quad & \begin{bmatrix} 0 & \mathbf{1}^T \end{bmatrix} z \\ \text{subject to} \quad & \begin{bmatrix} W & -I \\ -W & -I \end{bmatrix} z \preceq 0 \\ & z^T \begin{bmatrix} \left(\Phi H \Psi\right)^T \left(\Phi H \Psi\right) & 0 \\ 0 & 0 \end{bmatrix} z + \begin{bmatrix} -2y^T \Phi H \Psi & 0 \end{bmatrix} z + y^T y - \kappa \leq 0. \end{aligned}$$

For cleaner reference, the problem is

$$\begin{aligned} \hat{z} = \underset{z}{\text{argmin}} \quad & f_0(z) = c^T z \\ \text{subject to} \quad & f_1(z) = z^T P z + q^T z + y^T y - \kappa \leq 0 \\ & f_2(z) = a_1^T z \leq 0 \\ & \ldots \\ & f_{2n+1}(z) = a_{2n}^T z \leq 0. \end{aligned} \tag{2}$$

Approach

Since the objective and constraint functions are either affine or quadratic, a simple log barrier interior point method is the most straightforward way to solve the problem. The barrier function is

$$\phi(z) = -\sum_{i=1}^{2n+1} \log\left(-f_i(z)\right), \tag{3}$$

making the centering step

$$z^*(t) = \underset{z}{\text{argmin}}\ t f_0(z) + \phi(z). \tag{4}$$

Calling this new objective function $g(z)$, the Newton step and decrement for minimizing the centering step are

$$\begin{aligned} \Delta z_{nt} &= -\nabla^2 g(z)^{-1} \nabla g(z) \\ \lambda^2 &= \nabla g(z)^T \nabla^2 g(z)^{-1} \nabla g(z). \end{aligned} \tag{5}$$

The derivatives of our unconstrained objective function, $g(z)$, are

$$\begin{aligned} \nabla g(z) &= t\nabla f_0(z) + \nabla \phi(z) = tc - \sum_{i=1}^{2n+1} \frac{\nabla f_i(z)}{f_i(z)} \\ \nabla^2 g(z) &= \nabla^2 \phi(z) = \frac{\nabla^2 f_1(z)}{-f_1(z)} + \sum_{i=1}^{2n+1} \frac{\nabla f_i(z) \nabla f_i(z)^T}{f_i(z)^2}. \end{aligned} \tag{6}$$

The most expensive part of calculating $\nabla g(z)$ is finding the product

$$Pz = \begin{bmatrix} Z^T Z & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix},$$

where $Z = \Phi H \Psi$. Left multiplying a vector by Z costs $\mathcal{O}(n \log n)$. This is because left multiplying by Ψ is equivalent to calculating the IDCT of x ($\mathcal{O}(n \log n)$ with the FFT algorithm), left multiplying by H is equivalent to convoluting with some vector $h \in \mathbf{R}^n$ (also $\mathcal{O}(n \log n)$ using the FFT algorithm), and left multiplying by Φ, which is equivalent to just picking out $m < n$ elements of a vector, costs $\mathcal{O}(m)$. Similar analysis for left multiplication by Z^T yields a cost of $\mathcal{O}(n \log n)$, giving the total cost of calculating $\nabla g(z)$ as $\mathcal{O}(n \log n)$.

Inverting $\nabla^2 g(z)$ is more expensive. Expanding out the expression for $\nabla^2 g(z)$ in equation 6 gives

$$\nabla^2 g(z) = \sum_{i=2}^{2n+1} \frac{a_{i-1} a_{i-1}^T}{f_i(z)} + \frac{-2P}{f_1(z)} + \frac{\nabla f_1(z) \nabla f_1(z)^T}{f_1(z)^2} = A_z + \frac{-2P}{f_1(z)} + \frac{\nabla f_1(z) \nabla f_1(z)^T}{f_1(z)^2}, \quad (7)$$

where A_z is a rank $2n$, symmetric block diagonal with 3 non-zero diagonals – making it invertible in $\mathcal{O}(n)$ time. With placeholder variables, equation 7 becomes

$$\nabla^2 g(z) = A_z - \frac{2}{f_1(z)} U^T U + b b^T, \quad (8)$$

where

$$U = \begin{bmatrix} Z & 0 \end{bmatrix}$$

and b is a scaled $\nabla f_1(z)$. The matrix inversion lemma, detailed in §C.4.3 of [BV04], states

$$(A + BC)^{-1} = A^{-1} - A^{-1} B (I_m + C A^{-1} B)^{-1} C A^{-1}, \quad (9)$$

where $A \in \mathbf{R}^{n \times n}$ is nonsingular, $B \in \mathbf{R}^{n \times m}$, and $C \in \mathbf{R}^{m \times n}$. Assuming A^{-1} is fast to calculate, applying equation 9 reduces the cost of calculating the inverse of the sum to $\mathcal{O}(m^3)$ – the time needed to calculate the dense $m \times m$ inverse inside the parentheses. This implies we can calculate the inverse of $\nabla^2 g(z)$ with a rank m update

$$(A_z - U^T U)^{-1} = A_z^{-1} - \frac{2}{f_1(z)} A_z^{-1} U^T \left(\frac{2}{f_1(z)} U A_z^{-1} U^T - I_m \right)^{-1} U A_z^{-1} = K$$

followed by a rank one update

$$\left(A_z - \frac{2}{f_1(z)} U^T U + b b^T \right)^{-1} = K - \frac{K b b^T K}{1 + b^T K b}.$$

The most expensive part is the dense rank m inverse at $\mathcal{O}(m^3)$ while left multiplying by the Hessian, say for $\nabla^2 g(z)^{-1} \nabla g(z)$, costs $\mathcal{O}(m^2)$. Depending on how much we compress, i.e. how much smaller m is than n, we can gain some significant savings over $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ respectively.

Implementation

I implemented the solver in Python and am using CVXPY as a verification tool. Currently, the solver is converging on a solution 10x faster than CVXPY. Unfortunately, the solution it converges on isn't the right one. I'm still working that out.

References

[BG03] Marina Bosi and Richard E. Goldberg. *Introduction to Digital Audio Coding and Standards*. Kluwer Academic Publishers, 2003.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CW08] Emmanuel J. Candes and Michael B. Wakin. An Introduction To Compressive Sampling. *IEEE Signal Processing Magazine*, 25(March 2008):21–30, 2008.

[CWB08] Emmanuel J. Candes, Michael B. Wakin, and Stephen P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14:877–905, 2008.

[DMD+13] B Defraene, N Mansour, S De Hertogh, T van Waterschoot, M Diehl, and M Moonen. Declipping of Audio Signals Using Perceptual Compressed Sensing. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(12):2627–2637, 2013.