Wavelets, compression and compressed sensing in magnetic resonance imaging

Prof. Brian Frost

ECE-435-1, Fall 2023

Introduction

In class, we discussed the theory of convex optimization, and its application to compressed sensing. In this project, you will explore these ideas practically using real MRI data. The data we will use is on OpenNeuro, found at this url:
`https://openneuro.org/datasets/ds001499/versions/1.3.0`.

As can be read in the ReadMe on this website, there are four patients: CSI1, 2, 3 and 4. Most of the sessions here were for functional recordings (fMRI), but we are interested in anatomical recordings (MRI). For patients CSI1-3, the anatomical recordings are from session 16, and for patient CSI4, the anatomical recordings are from session 10. The images are either T1- or T2-weighted.

The datasets are titled, for example, "sub-CSI1_ses-16_run-01_T1w.nii.gz." Extracted, these .nii files are called NIfTI files, which is a standardized file type used in MRI, fMRI, PET and other functional imaging modalities. They can be read into MATLAB via `niftiread`, and Python by using `load` in the `nibabel` library. These contain three-dimensional 16-bit integer arrays, in the *spatial* domain. That means the IFT has already been applied by the machine.

As usual, you are expected to write a report in which you display and explain all of your results, along with answering any specific questions asked in this document. While I suggest the use of MATLAB or Python, any software package may be used so long as you give me your code and instructions on how to run it. You will need to take wavelet transforms, so MATLAB's Wavelet toolbox, or Pythons `pywt` will be helpful.

I. Navigating the volumes

To start, consider Patient CSI1's anatomical session, and load in both the T1- and T2-weighted data. Create a figure with 6 subplots that displays, in grayscale and scaled magnitude (imagesc in MATLAB), sagittal, coronal and axial cross-sections from each volume. Label each in a format like "Axial slice 150, T1-weighted." Make note of how the dimensions of the array corresponds to anatomical axes.

In your report, use the following table of T1 and T2 values to describe which structures are shown most brightly. Does T1/T2 weighting reward large or small T values?

Structure	T1 (ms)	T2 (ms)
White Matter	510	67
Grey Matter	760	77
Cerebrospinal Fluid	2650	280

Look at the corresponding json files in the run's folder. Explain how the inversion time, echo time and repetition time relate to the weighting. Explain by appealing to either conceptual NMR principles, or the imaging equations. They don't give units for the pulse sequence times – what do you think the units are, guided by the values in the table above?

For the next several problems, we will want a "guiding image," which will serve as a visual marker for the success/failure of the methods we will attempt here. I suggest the center-index fixed-x image.

II. Auxiliary Functions

Although you will probably write several other functions as you complete this project, I think you will do well to make sure you have these "right off the bat." This isn't a requirement but rather a suggestion.

1. A function that takes an array and a number `s` between 0 and 100. The returned array is the same size as the input, but the lowest-magnitude `s`% values in the input array are 0 in the output. This is a simple compression.

2. A function that takes two arrays, one of which is considered a ground truth, and computes the mean square error.

3. A function that takes an array and a name of a wavelet domain, and plots the approximation and detail images for the given wavelet transform at two levels. They can be shown either concatenated on one set of axes, or in a single figure in many subplots (former preferred). This is trivial in Python but at the cost of clarity. In MATLAB, this requires understanding the `wavedec2`, `appcoef2` and `detcoef2` functions. An easy example can be found **here**.

III. Fourier Domain Compression?

Show the (scaled) magnitude of your guiding image in the Fourier domain. Make sure you are taking a 2D FT, and not column-wise 1D FTs. Apply `fftshift`, as well. Show a histogram of this magnitude with a few hundred bins, and comment on the sparsity. When you take histograms, be sure your data is reshaped to 1D.

Note that this is the domain in which our images are acquired. If many of these points are low-magnitude, maybe we can just sparsely sample this domain and not have to do much else! Consider your guiding image as ground truth, and observe the reconstructed image after zeroing the smallest Fourier coefficients with several `s` values. Note also the mean-squared errors. Comment on the results here in a detailed fashion, showing comparisons between the original and reconstructed images labeled by error and what percentage of coefficients were zeroed.

You should find that you can't achieve significant compression on account of the visual quality reconstructed of the image, even if the mean-squared error appears low. This is because the phase of the Fourier domain information is very important, even in the very low-magnitude components. Thereby even though we have sparsity in the Fourier domain, we cannot use this domain for image compression!

IV. Wavelet Domains

For this problem, we will consider three wavelet bases – Haar, Daubuchies 4 and Coiflet 3. Familiarize yourself with the workings of the wavelet transform functions in the toolbox you are using. To prove to yourself you know what is doing on, take a 2-level 2D wavelet decomposition of a standard MATLAB image (for example, 'cameraman') and display the approximation image, and the 6 detail coefficient images, using the function you wrote above.

Try plotting the magnitude of the 2D, 2-level wavelet transforms in these three bases to a standard basis element in 256×256 matrix space (a standard element is one which is 0 everywhere except one pixel at which it is 1).How do they look? Plot their Fourier transform magnitudes. How are these elements localized in space/frequency?

Use either MATLAB or Google to see canonical spatial representations of these wavelets and see if your commentary above matches what you read, if you'd like.

Now display the 2-level decompositions of the guiding image in these three bases. Is the image sparse in these bases? Show histograms for the *total set of coefficients* in these three bases after 1-level, 2-level and 3-level decomposition, for a total of 9 histograms. Which domain looks like the best candidate for compression/compressed sensing?

Now just to be sure you know what you're doing, pick the domain you like best and reconstruct the image where the lowest 10% of the coefficients are zeroed. Display this alongside the original image, and the mean-squared error.

V. How Sparse?

Say the percent of coefficients being deleted is `s`. Plot the mean squared error between the reconstructed signal and the true signal as a function of `s` for the guide image for each basis. Do you see diminishing returns? Plot a few reconstructed images with each basis. Toy around with the number of levels, and see if it works for images other than your guide image, including one from the T2 dataset.

Make qualitative statements, and also say how many coefficients we can toss if we can accept 10% error. Use a for loop to see what the max error among all images would be with this number of coefficients kept.

VI. Compressed Sensing

Write a gradient descent algorithm that simulates the compressed sensing of MRI images based on any of: (1) sparsity in the wavelet domain, (2) TV regularization or (3) TGV regularization. It is your choice which you pick, and you will receive extra credit for using and comparing multiple results. If you choose method (1), pick the wavelet domain based on what you see works best qualitatively/quantitatively in Section V.

Ground truth is the guiding image in the k-domain, i.e. the FT of the guiding image. This is undersampled by a random sensing matrix (you generate a Bernoulli random vector to perform the undersampling with probability p). The domain of sparseness is the DWT/TV/TGV of the IFT of the k-domain. Show the average convergence of this algorithm in objective function as a function of iteration for 1000 sensing matrices with the same p. Do this for many values of p and give qualitative/quantitative arguments for a good p to use. I am being purposefully vague because

I want you to explore the space. Try different values of the sparsifying parameters λ early on and settle on a good one before you do the rest of your tests.

Suggested Completion Timeline

By Week 12 complete Section I above, write functions (1) and (2) from section II and consider Section III. By Week 13, Complete Sections III and IV. By Week 14, Complete V and start VI. By Week 15 be done with the project, spending the last week exploring section VI. Update your report each week, and come to class with questions each week. You can hand in the report by the last day of the semester.