Compressed Sensing Project Notes

Victor Minden*

December 9, 2014

1 Explanation of choices in project code

My project code is broken up into Matlab core code and Python driver code in the following files:

- `matrix_sample.m`
- `random_rank_rm`
- `problem_instance.m`
- `run_experiment.m`
- `ClusterDriverMatlabSmall.py`

I chose to build each class of matrix (the 'type' string) in a `switch` statement inside of [`matrix_sample`, `random_rank_r`] because each function was very short and it made the total number of files much more manageable. Otherwise, I believe that my Matlab code is pretty much exactly what Prof. Donoho provided for us as a code sketch, minus some tweaks for small typos and the like.

As far as the Python code, my entire experiment package is run through `ClusterDriverMatlabSmall`. The way this file works is it holds a template Matlab script and a template shell script. These are the bare bones necessary to call `run_experiment` with the correct parameters and to submit a job using `qsub`. Then, the code loops over a list of matrix types for both A (sensing matrices) and X (true matrix) and generates a directory for each valid combination of types (we avoid complex observations of real matrices). Once these directories are in place, they are populated with a Matlab script and shell script generated from the templates, and then the shell script is executed using `qsub`.

2 Explanation of tests I have run

Because I built many matrices out of other matrices (for example, take a permutation matrix, element-wise multiply it with some signs, etc.), I was able to break up my unit tests.

Here is a listing of the tests I ran that passed, the code for these was provided with HW3

- Entry: assert that there is only one nonzero
- Perm: assert that there is only one nonzero per row/column
- RSPerm: assert that all entries are square-root of 1
- CSPerm: assert that all entries are 4th-root of 1

*Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305. Email: vminden@stanford.edu

- RGPerm: assert that mean entry is close to zero and second moment is close to $1/n$
- CGPerm: same as above except second moment is of complex modulus.
- RDirac: check Fro norm
- CDirac: check Fro norm
- RGauss: assert stats match as before
- CGauss: assert stats match as before

For the random rank-r matrices, I had three different tests that I used as needed: `assert(min(lambda) > -1e-3);` `assert(norm(A-A',’fro’) == 0);` and `assert(max(abs(abs(lambda)-1)) < 1e-3);` These all worked fine.

There was nothing particularly novel about the rest of my testing setup. The code is in HW3.

3 Opinions on what might be different

As I have mentioned before, I think that in the ’HERM’ and ’HPSD’ cases it should be valid to take any sort of measurement generated from ’RDirac’ and ’CDirac’, *i.e.*, we should not throw away anti-symmetric or anti-Hermitian measurements as we do for ’RSYM’ and ’RPSD’ The rationale for this is simple. For real symmetric X, we are guaranteed that $(A, X) = 0$ whenever A is anti-symmetric. This led us to call this a redundant measurement and throw it away. For Hermitian X, we have that (for example, four-by-four case), $(A, X) = a^*_{12}x_{12} + a^*_{21}x_{21} = a^*_{12}x_{12} \pm a_{12}x^*_{12} = c \pm \bar{c} \neq 0$, a meaningful measurement. Here the $\pm$ corresponds to whether A is anti-Hermitian or anti-symmetric.

Looking at it another way, lets consider the rank of $A_{\text{big}}X_{\text{big}}$, where A_{big} has rows that are given by $\text{Vec}(A_i)^*$ and X_{big} has columns given by $\text{Vec}(X_j)$. In other words, we want to know what the rank of the output matrix looks like if we were to take measurements of a whole bunch of X_j matrices.

When all the X_j are symmetric, then any anti-symmetric A_i leads to a row of $A_{\text{big}}X_{\text{big}}$ that is identically zero. This measurement is purely redundant, even as a singleton. However, when all the X_j are Hermitian, then the corresponding row of $A_{\text{big}}X_{\text{big}}$ is not necessarily zero at all. The ultimate rank of $A_{\text{big}}X_{\text{big}}$ is still limited by the fact that there are only $\sim n^2/2$ linearly independent measurements to be made of a Hermitian matrix, but no single measurement on its own is always negligible, it can only be neglected if drawn with a redundant combination of other measurements.