Exercises

These exercises range from straightforward to somewhat involved, and also include several programming problems, which will be either straightforward or involved depending on your familiarity with MATLAB. The expectation is not that everyone will do every problem, but that people will choose problems that they think are interesting or at the appropriate level. More difficult problems are marked with a $\star$.

The implementation exercises were designed to be done with grayscale images in MATLAB using the MATLAB Wavelet Toolbox. The exercises include various MATLAB commands. You may use other software and/or images if you would like.

1. Lecture 1

(1) Let $\mathbf{y} \in \mathbb{R}^N$. Let $\mathbf{1}$ denote the n-vector with all entries equal to one. We wish to approximate $\mathbf{y}$ by $\alpha\mathbf{1}$, where $\alpha \in \mathbb{R}$.
  - (a) Show that the best approximation in the ℓ_2 sense occurs when $\alpha = \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.
  - (b) Show that the best approximation in the ℓ_1 sense occurs when α is the median of the entries of $\mathbf{y}$.

(2) Compute a two-dimensional Haar wavelet decomposition of an image using `mdwt` in MATLAB using Matlab's Wavelet Toolbox. Then threshold the coefficients and reconstruct the image. Display the original image, the original coefficients, the reconstructed image, and the thresholded coefficients. Also,
  - Calculate the relative ℓ_2 error between the image and the reconstruction and the percentage of zero coefficients for different threshold values. Then plot the relative error vs. the percentage of zeros. Discuss.
  - Sort the coefficients in decreasing magnitude and plot with `loglog`. Try to find a linear envelope for the tail of the coefficient vector. This is easier to do if you use the maximum number of levels to decompose the image. Also, you can ignore the sharp dropoff for the last few coefficients.

(3) **Nonlinearity of transform coding.** Consider the trivial transform coder, in which we retain the k signal vlues with the largest magnitude and set all others to zero. Denote the transform coding of a signal f by Tf.
  - (a) Give an example of two signals that demonstrate the nonlinearity of this coder.
  - (b) Consider signals in $\mathbb{R}^n$. For a fixed k and $n > 2k$, is there a limit to how large $\|(Tu + Tv) - T(u + v)\|/\|Tu + Tv\|$ can be?

(4) Use the "peppers" image, which we will denote by I_0. Perform a 3-level Haar wavelet decomposition of this image, threshold the coefficients, and reconstruct the image. Display the original and compressed images and compare. Adjust the threshold value so that 98.5% of the coefficients are zero. Call the reconstructed image I_1.

Take the residual $I_0 - I_1$, and repeat the above process using a wavelet packet decomposition of the residual using `wpdencmp`. Adjust the threshold until 96.7% of the coefficients are zero. Call the compressed residual I_2. Plot the original image I_0, the reconstructed wavelet compressed image I_1, and the reconstructed wavelet and wavelet packet compressed image $I_1 + I_2$, and compare. *Tip:* Use MATLAB's `linkaxes` command and zoom in on different parts of the image.

You may repeat this exercise for other images, but you may need to adjust the sparsity percentages.

The following two definitions are needed for the next exercise.

Definition 1 (l_p norm)**.** *Let* $1 < p < \infty$*, the* l_p *norm on* $\mathbb{R}^N$ *is*

$$\|x\|_p = \left(\sum_{i=1}^{N} |x_i|^p \right)^{\frac{1}{p}} \tag{1.1}$$

Definition 2 (weak l_q norm)**.** *Let* $0 < q < \infty$*, the weak* l_q *norm on* $\mathbb{R}^N$ *is*

$$\|x\|_{wq}^q = \sup_{\varepsilon > 0} \varepsilon^q \left| \left\{ i \middle| |x_i| > \varepsilon \right\} \right| \tag{1.2}$$

(5) Show that $\|x\|_{wq} \leq \|x\|_q$.

(6) **An introduction to P and NP.**

(a) Formally formulate the decision problem version of the problem of answering "is this list sorted" and argue that the corresponding language is in P.

(b) Formally formulate the decision problem version of "is this graph 3-colorable."[1] Prove that the corresponding language is in NP.

(c) There are some problems which are even harder than NP; the canonical example is the *halting problem.* Given an algorithm A which takes an input x and either runs forever or stops and outputs 1 or 0, let $\langle A \rangle$ denote a description of A (for example, in machine code—it doesn't matter what the encoding process is, as long as it is fixed). Let L be the language

$$H = \{ (\langle A \rangle, x) \ : \ A(x) \text{ eventually stops.} \} .$$

Show that H is *not* in NP.

(d) ⋆ Perhaps you found that last part unsatisfying. It's true that H is not in NP, but it's also not decidable (that is, there's no algorithm which can decide in finite time whether $x \in H$). In this part you'll show that some (decidable) problems are harder than NP.

(i) As a warmup, show that there are problems which can be solved in time n^2 which can't be solved in time n. To be more precise, find a language L so that there is a deterministic algorithm A which runs in time $O(n^2)$ and decides if an input x of length n is in L, but so that there is no A' which can do the same in at most n steps. (Hint: Try a diagonalization argument. That is, if $A_1, A_2, \ldots$ are an enumeration of all of the machines that run for n steps on inputs of length n, try find a language that each A_i is wrong on in at least one place).

(ii) This one's a bit harder. Show that there is a language L so that there is a verification algorithm A which will decide (with a witness) whether $x \in L$, but there is no verification algorithm A' which can do this task in polynomial time.

[1] a coloring on a graph is an assignment of colors to the vertices so that any two vertices connected by an edge have a different color. A 3-coloring is a coloring that uses only 3 colors.

2. Lecture 2

(1) Show that the best k-sparse approximation of a vector $u \in \mathbb{R}^n$ in terms of ℓ_1 and ℓ_2 error is obtained by hard thresholding, i.e. retain the k entries of u with the largest magnitudes and set all of the other entries to zero.

(2) Calculate (or use a computer to estimate) the coherence for the following dictionaries:
   (a) the Fourier-Dirac dictionary,
   (b) a wavelet packet dictionary,
   (c) a d-by-N matrix with standard normally distributed entries (remember to normalize the columns), and
   (d) a random orthoprojector.

   For the last two parts, you may want to run some experiments in Matlab.

The following 4 exercises pertain to the exact hitting set problem.

(3) The *exact hitting set problem* is as follows: given a finite universe $\mathcal{V}$, a collection $\mathcal{Y}$ of subsets $Y_1, Y_2, \dots Y_N$, is there a subset S of $\mathcal{V}$ such that each subset in $\mathcal{Y}$ contains *exactly one* element in S? Show how to reduce this problem to an exact cover problem.

(4) Define the incidence matrix Φ by

$$\Phi_{ij} = \begin{cases} 1 & \text{if } v_j \in Y_i \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, \dots, N \text{ and } j = 1, \dots, |\mathcal{V}|.$$

   How can you express the exact hitting set problem as a linear algebra problem using the matrix Φ?

(5) In a standard Sudoku game, there are 81 cells arranged in 9 rows and 9 columns. The field is further divided into 9 3-by-3 blocks. The numbers 1 through 9 can be entered into the cells according to the following rules.
   - **Row-column:** Each cell contains exactly one number.
   - **Row-number:** Each row contains exactly one copy of each number.
   - **Column-number:** Each column contains exactly one copy of each number.
   - **Block-number:** Each block contains exactly one copy of each number.

   The game of Sudoku can be formulated as an exact hitting set problem. Each cell-number choice corresponds to a certain element in the universe of possibilities. For example, one possibility is the number 4 in the cell located in row 2 and column 5, which we will denote as $r2c5n4$. Each constraint then corresponds to a subset of these possibilities. A solution to the Sudoku game is a subset S of the possibilities such that each constraint subset contains exactly one element of S. Write down the constraint subsets. Think about how many constraints of each type exist, and of a sensible way to index the constraints.

(6) We will explore a somewhat simpler Sudoku game played on a 4×4 grid. The `Sudoku` directory contains code for building the incidence matrix Φ. You can solve some Sudoku puzzles using the `testcode` script. You can input a Sudoku puzzle by typing it in as a matrix, entering a zero for each blank. The code uses an iterative algorithm to find a sparse solution to the corresponding linear system. Sometimes this algorithm finds the correct solution and sometimes it does not.
   - View the matrix Φ and the Gram matrix $\Phi^T\Phi$ using `imagesc`.
   - Find the largest number of blanks a Sudoku puzzle can have and still be solvable by the algorithm.

(7) Let $A \in \{0,1\}^{m \times n}$. We say that A is *d-disjunct* if there is no column of A that is contained in the "union" of any d other columns of A. More precisely, A is d-disjunct if for all $S \subset [n]$ with $|S| = d$, for all $i \in [n]$, there is some $j \in [n]$ so that $A_{ji} = 1$ but $A_{j\ell} = 0$ for all $\ell \in S$. The decision problem d-Disjunct is, given A and d, to decide whether or not A is

d-disjunct.

We say that a language is in **co-NP** if its complement is in NP. For example, the problem NOT-X3C of deciding that there is *no* exact cover is co-NP complete. Use a reduction from NOT-X3C to show that D-DISJUNCT is co-NP hard.

(8) Recall from class that (the decision problem version of) ERROR is the following problem: given $\Phi \in \mathbb{R}^{m\times n}, k > 0$, $\epsilon > 0$, and $y \in \mathbb{R}^m$, decide if there is an $x \in \mathbb{R}^n$ so that x is k-sparse, and

$$\|\Phi x - y\| < \epsilon.$$

Adapt the reduction from class to show that ERROR is NP-complete. (Hint: the "adaptation" should not require much adapting).

3. Lecture 3

(1) Take a 32×32 block from a random location in an image, and call this block I. Let Φ be a redundant dictionary consisting of a Dirac basis and a two-dimensional discrete cosine transform (DCT) basis. Implement OMP to compute the k-sparse approximations of I for $k = 1, 2, 3, \dots, 100$. (You can create a general algorithm that can be applied to any dictionary, or you can create an algorithm tailored to this specific choice of dictionary.) For each value of k display the k-sparse approximation and the residual.

Tips:

- To compute the DCT coefficients and DCT modes you may want to use `dctmtx`.
- For each value of k, you will construct a k-by-32^2 matrix A consisting of the k optimal atoms written as vectors. You can find the the vector c of optimal coefficients by finding a least squares solution of $Ac = I$ using the backslash command, where I is the image written as a vector.

(2) Prove the **Welch bound**, that the coherence μ of a $d \times N$ dictionary is bounded by

$$\mu \geq \sqrt{\frac{N-d}{d(N-1)}}.$$

Hint 1: Consider the trace of $\Phi^*\Phi$. On the one hand, this is N (why?). On the other hand, you can bound it above: First use the Cauchy Schwartz inequality and the fact that

$$\sum_{i=1}^{N}\sum_{j=1}^{N} |\langle x_i, x_j\rangle|^2 = \sum_{i=1}^{N} \lambda_i^2,$$

where λ_i are the eigenvalues of $\Phi^*\Phi$, to show that

$$\sum_{i \neq j} |\langle x_i, x_j\rangle|^2 \geq \frac{N}{d} - N.$$

(It helps to notice that at most d of the λ_i are nonzero—why?). Then bound μ in terms of $\sum_{i\neq j} |\langle x_i, x_j\rangle|^2$.

Hint 2: There's a proof on Wikipedia if you get stuck.

(3) Recall from Professor Willett's course (or be informed now) that we say that $\Phi \in \mathbb{R}^{m\times n}$ satisfies the **restricted isometry property** with constants k and δ if, for all k-sparse vectors x,

$$(1-\delta)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1+\delta)\|x\|_2^2.$$

Recall from class that OMP is the following algorithm (I've added some extra lines in order to define notation):

```
• $r_0 = x$, $c = 0$, $t = 1$, $S_0 = \emptyset$
• while $t \leq k$:
        $j_t = \arg\max_\ell |\langle r_{t-1}, \varphi_\ell\rangle|$
        $S_t = S_{t-1} \cup \{j_t\}$
        $c_t = \arg\min_{\mathrm{supp}(c)\subset S_t} \|x - \Phi c\|_2$
        $r_t = x - \Phi c_t$
        $t = t + 1$
```

Show that, if Φ has the RIP with constants $k+1$ and $\delta \leq \frac{1}{4\sqrt{k}}$, that OMP will solve Exact for any k-sparse vector c^*, in k iterations. Here's an outline of one way the proof could go[2]:

[2]See Davenport and Wakin 2010, "Analysis of Orthogonal Matching Pursuit using the Restricted Isometry Property"

(a) Show that Φ has the RIP of order k' with constant δ and u and v have $\|u \pm v\|_2 \leq k'$, then

$$|\langle \Phi u, \Phi v\rangle - \langle u, v\rangle| \leq \delta \|u\|_2 \|v\|_2.$$

(Hint: use the parallelogram law.)

(b) Show that for any k-sparse $c \in \mathbb{R}^N$, $\|c\|_\infty \geq \frac{\|c\|_2}{\sqrt{k}}$.

(c) The proof will go by induction. For the following steps, assume the inductive hypothesis that $S_t \subseteq \text{supp}(c^*)$. Notice that the inductive hypothesis is satisfied to begin with.

(d) Let Π_t denote the projection on the columns of Φ indexed by S_t. Show that $r_t = (I - \Pi_t)x$.

(e) Let c_t^* denote the restriction of c^* to the complement of S_t. That is, $(c_t^*)_j = 0$ if $j \in S_t$, and $(c_t^*)_j = (c^*)_j$ otherwise. Show that $(I - \Pi_t)\Phi c^* = (I - \Pi_t)\Phi(c_t^*)$.

(f) $\star$ Show that if Φ has the RIP of order k and constant δ, then for any u with $\|u\|_0 \leq k - |S_t|$ and $\text{supp}(u) \cap S_t = \emptyset$, then

$$(1 - 2\delta)\,\|u\|_2^2 \leq \|(I - \Pi_t)\Phi u\|_2^2 \leq (1 + \delta)\|u\|_2^2,$$

where Π_t is as in the previous part. That is, $(I - \Pi_t)$ has the RIP with constant 2δ. Hints: First write $\|\Phi u\|_2^2 = \|\Pi_t \Phi u\|_2^2 + \|(I - \Pi_i)\Phi u\|_2^2$. It suffices to show that $\|\Pi_t \Phi u\|$ is suitably small. Use part (3a).

(g) $\star$ We are finally ready to do the inductive step! Use parts the previous parts to show that j_t is in the support of c_t^* (and not in S_t). Hints: Apply part (3a) to part (3f) to get a bound on $\langle r_t, \varphi_\ell\rangle - c_t^*$, after rewriting it using parts (3d, 3e). Use part (3b) to show that we must pick an index in the support of c_t^*.

(h) Explain why we are done.

4. Lecture 4

Definition 3. *A* norm *is a function $f : \mathbb{R}^n \to \mathbb{R}$ with the following properties:*

- Nonnegativity. *$f(x) \geq 0$ for all $x \in \mathbb{R}^n$.*
- Nondegeneracy. *The only vector with norm zero is the zero vector, i.e. $f(x) = 0 \iff x = 0$.*
- Homogeneity. *For any $\alpha \in \mathbb{R}$, $x \in \mathbb{R}^n$, $f(\alpha x) = |\alpha| f(x)$.*
- Triangle inequality. *For any $x, y \in \mathbb{R}^n$, $f(x + y) \leq f(x) + f(y)$.*

Definition 4. *A* convex function *is a function $f : \mathbb{R}^n \to \mathbb{R}$ such that for all $\alpha \in [0, 1], u, v \in \mathbf{R}^n$,*

$$f\left((1-\alpha)u + \alpha v\right) \leq (1-\alpha)f(u) + \alpha f(v).$$

Definition 5. *The l_0* "norm" *is defined as*

$$\|x\|_0 = \#\{i \;:\; x_i \neq 0\}.$$

Definition 6. *The l_1* norm *is defined as*

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

(1) Why is l_0 not a norm? Why is it not a convex function? Why is l_1 a convex function?

(2) * Recall from Professor Raskhodnikova's class last week that communication complexity is a good way to prove lower bounds. In this problem, we'll use it to prove lower bounds for the Compressed Sensing problem. Recall that the goal of Compressed Sensing is to define a distribution $\mathcal{D}$ over $d \times n$ matrices so that for any fixed $x \in \mathbb{R}^n$, if Φ is drawn from $\mathcal{D}$, then $\hat{x}$ is recoverable from Φx so that

$$\|x - \hat{x}\|_1 \leq C\|x - x_k\|_1$$

for some constant C, with probability at least 2/3 over the choice of Φ. We'll show that $d = \Omega(k \log(n/k))$.[3] You may want to suppose for simplicity that the entries of Φ have b bits of precision, for some universal constant b. We'll need the following facts.

- There is a set $S \subset \{0,1\}^N$ with $\log(|S|) = \Omega(k \log(N/k))$ so that each $s_i \in S$ is k-sparse so that $\|s_i - s_j\|_0 \geq k$ for all $i \neq j$.
- Suppose that Alice is given $x \in \{0,1\}^N$ and Bob is given $i \in \{1, 2, \ldots, N\}$, as well as $x_1, \ldots, x_{i-1}$. Alice sends one message to Bob, and Bob must output x_i with probability at least 2/3. Alice and Bob have shared randomness. This problem is called Augmented Indexing, and the communication complexity (that is, the number of bits that Alice sends Bob) is known to be $\Omega(N)$.

Given these facts, use a reduction from Augmented Indexing to show that $d = \Omega(k \log(n/k))$.

Hints: (These will basically give away the answer, although it leaves the details to you; you might want to try it on your own first).

- The idea is that Alice will break up x into ℓ chunks of size $\log |S|$, and each chunk will be a pointer to some k-sparse string s_i.
- Alice will send $\Phi\left(\sum_{j=1}^{\ell} \sigma_j s_j\right)$ for appropriate σ_j, where σ_j is very large for small j and less large for large j.
- Bob will be able to compute $\Phi\left(\sum_{j=i}^{\ell} \sigma_j s_j\right)$. (Why?)
- If σ_j are chosen appropriately, then the "head" of the signal $\sum \sigma_j s_j$ will just be $\sigma_i s_i$. If the compressed sensing algorithm is good, Bob can recover s_i with minimal noise.

[3]This exercise follows the argument in Do Ba, Indyk, Price, and Woodruff, "Lower bounds for sparse recovery," 2010.