Homework 9 and 10
APPM 4720/5720 Spring 2017
Advanced Convex Optimization

Due date: Friday, Mar 24 2017
Theme: Duality again

Instructor: Dr. Becker

Instructions Collaboration with your fellow students is allowed and in fact recommended, although direct copying is not allowed. The internet is allowed for basic tasks. Please write down the names of the students that you worked with. An arbitrary subset of these questions will be graded.

Reading Read the rest of chapter 5 in [BV2004].

Background: compressed sensing

We will introduce compressed sensing along with an example. The basic idea of compressed sensing is that if a signal $x \in \mathbb{R}^n$ is compressible (i.e., it only has k nonzero entries), and we acquire it with suitably *incoherent* linear sampling, then we can recover the true signal with about $k \log(n/k)$ measurements, rather than all n samples that we would need for a standard matrix inversion. We will leave the notion of "incoherence" vague for now, but note that the time and frequency domains are roughly incoherent (for the same reason that we have a time-frequency uncertainty principle). For background on compressed sensing, see the review article E. J. Candès and M. Wakin. An introduction to compressive sampling. IEEE Signal Processing Magazine, March 2008 21-30.

Let us make an concrete example with an audio signal. We'll take the "Hallelujah" chorus from Handel's *Messiah*, which, conveniently, is included as a .mat file with Matlab. Run[1] `load handel.mat` to load the variables `y` and `Fs`, where $y \in \mathbb{R}^N$ with $N = 73113$, and $Fs = 8192$ meaning the sample rate is 8.1 kHz. According to the Shannon-Nyquist sampling theorem, this means the data contains frequencies up to about 4 kHz, which is not that high (kids can hear up to about 20 kHz, older adults up to maybe 15 kHz), but still contains most of the notes of interest — for comparison, the note "A" above middle "C", called "A_4", is 440 Hz, and so 4 kHz allows for 3 octaves above this.

We can see the frequencies in the signal y in the upper-left plot of Fig. 1. There are two dominant tones, one at about 575 Hz and another at 1.13 kHz. Of course the tones are not static in time, as this is music; see Fig. 2 for the spectrogram of the signal, which shows the spectrum (frequencies) over time.

Suppose we want to store less data, and only sample every-other-entry from y. This creates an effective sample rate of $Fs/2$, leading to a max resolvable frequency of 2 kHz, which does not affect the sound quality much. But if we try to sample every-fourth-entry from y, so the sample rate is $Fs/4$, then the max resolvable frequency is 1024 Hz, which is below our second dominant tone. If we just downsample the signal, we alias the higher-frequencies, and have a new spurious note at 924 Hz, which is the aliased version of the 1.13 kHz note (that is, our new cutoff frequency is $8192/8 = 1024$ and 924 is about the same amount under 1024 as 1130 is over 1024). See the top-right plot of Fig. 1. You can hear the sound of this new signal following the Matlab/python code posted online, and it should sound weird.

If we can't hear that highest frequency, it is better to just remove it and not have it alias. The bottom row of Fig. 1 shows (left) a low-pass filtered version of the signal, and (right) the downsampled version of this. We have lost the high notes, but at least there are no weird aliased notes folded in.

[1] scripts for recreating this section in Matlab and Python are on the course github page

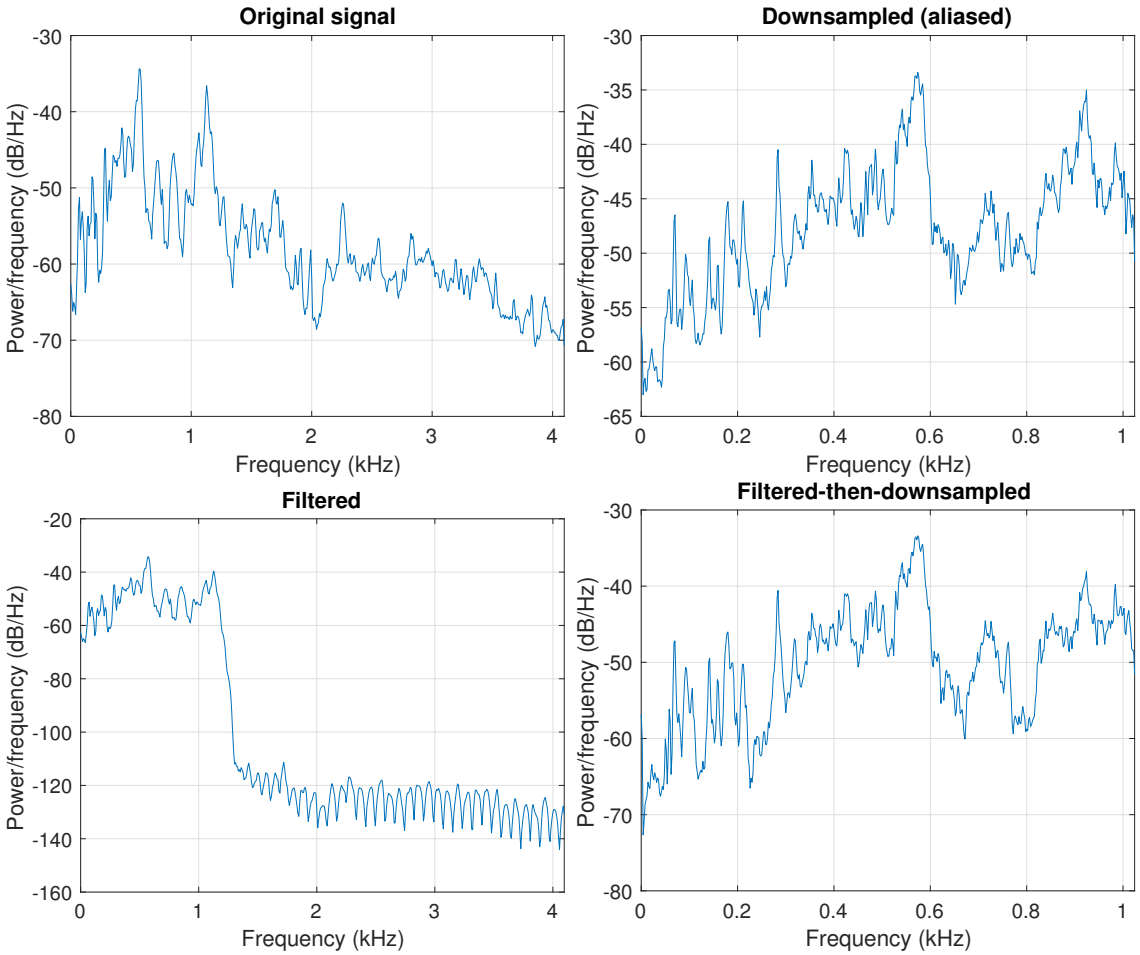

Figure 1: Estimates of the periodogram of the "Handel" signal using the pwelch method with a Modified Bartlett-Hanning window of length 1000.

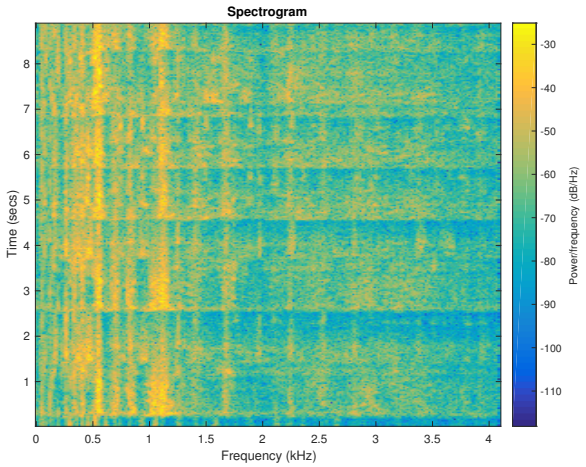

Figure 2: Spectrogram of the "Handel" sound file. In Matlab, this is generated via `load handel.mat; spectrogram(y,5e2,[],[],Fs)`.

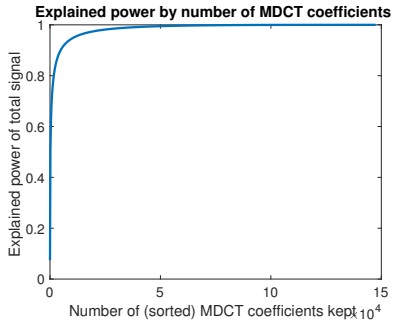

Figure 3: MDCT coefficients of the Handel signal

Is there any way to recover our signal using only $N/4$ measurements? By looking at the spectrogram in Fig. 2, we see that most entries in the time-frequency plane are negligible (remember, this is in dB, which is a logarithmic scale), so we could set these to zero and have very little effect on sound quality.

More specifically, we are going to consider using version of the modified discrete cosine transform (MDCT). The DCT is like the FFT/DFT in that it maps to a frequency space, but has the advantage of being real-valued. We will take the DCT of blocks of the signal (we'll use a block-length of 1024), so that we capture the instantaneous note, rather than an average of all notes. We also want to window each block so that we have sharp frequency peaks, but this means we should have the blocks overlap. This is a linear map, and we'll represent it by the symbol D, and it maps $\mathbb{R}^n$ to $\mathbb{R}^{2n}$ (actually, it will map $\mathbb{R}^{73113}$ to $\mathbb{R}^{146226}$ since we will zero-pad y to be an even multiple of the block size). We use a partition-of-unity window so that $D^TD = I_n$). The code to construct D and D^T is provided on the github page.

If we let $c = D(y)$ be the MDCT coefficients of our audio signal, only a few coefficients of c are large. If we sort them by absolute value, we see in Fig. 3 that we only need to keep a handful of them (much less than a quarter) to faithfully resolve the signal to, say, 90% accuracy.

Compressed sensing takes advantage of the sparsity of c as long as we have incoherent measurements. Also note that since $D^TD = I_n$, that given $c = Dy$, we can recover $y = D^Tc$. Note: from now on, we will switch from "c" to "x" to align with usual optimization notation, since this will be our optimization variable.

Let Ψ be the sampling operator that takes, *on average*, every fourth sample, and collect the data $b = \Psi(y)$. It is important that we take samples at random, and not take exactly every fourth sample, otherwise nothing can be done about the loss of high frequency. Then compressed sensing solves the basis pursuit problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \|x\|_1 \\ \text{subject to} & \Psi D^T x = b \end{array} \tag{BP}$$

Solving (BP) is the goal of this week's homework. It is a linear program, but not easy to solve with usual LP methods since they cannot take advantage of the structure of the MDCT. We will exploit the fact that we can use last week's proximal/projected gradient descent (or FISTA) code to solve a similar least-squares problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \|\Psi D^T x - b\|_2 \\ \text{subject to} & \|x\|_1 \leq \tau \end{array} \tag{LS$_\tau$}$$

Note that when we actually solve (LS$_\tau$), we can replace its objective with $\frac{1}{2}\|\Psi D^T x - b\|_2^2$, so that we have a smooth objective. The (BP) and (LS$_\tau$) can be made to be equivalent if we choose the right value of τ; in particular, if x_{BP} solves (BP), then we should choose $\tau^\star = \|x_{BP}\|_1$ (but of course we don't know this value). For all $\tau \geq \tau^\star$, the optimal value of (LS$_\tau$) is 0; for all $\tau < \tau^\star$, the optimal value is strictly bigger than 0.

Notation: let $A = \Psi D^T$ and $f_0(x) = \|Ax - b\|_2$.

Homework 9: derivatives via the dual problem

Read Homework 10 if you want motivation for this section.

Problem 1: Introduce a slack variable z such that $Ax + z = b$, and Lagrange multipliers λ (for the $\|x|_1 \leq \tau$ constraint) and ν (for the $Ax + z = b$ constraint), and write down the Lagrangian and dual function for (LS_τ) (really, for the modified version of (LS_τ) that has the slack variables).

Problem 2: Can we guarantee strong duality?

Problem 3: Let x_τ be the solution to (LS_τ), and let $\sigma(\tau) = \|Ax_\tau - b\|_2$ be the corresponding value of the objective. Suppose λ_τ and ν_τ are dual optimal variables. What is σ' (that is, $d\sigma/d\tau$) ? *Hint: we covered this in class.*

Problem 4: What are the KKT conditions for (LS_τ)?

Problem 5: Suppose we are given the primal solution x_τ, use the stationarity KKT condition to find the optimal Lagrange multiplier ν. Then use this to find the optimal Lagrange multiplier λ.

Homework 10: Compressed sensing

You may use the files on the class github website, or your classmates' files from the previous homework.

Problem 1: Solve (BP) using CVX/CVXPY, and making the matrix A and b to be whatever you want (but small dimensions, e.g., $A \in \mathbb{R}^{10\times 20}$). Call this solution x_{BP}.

Deliverables: this is short, so include both a printed write-up, as well as code uploaded to the google drive.

Problem 2: Using $\tau = \|x_{BP}\|_1$, solve (LS_τ) using your projected/proximal gradient descent (or FISTA) code from last week's homework (or from the solutions). You will need to have a function that projects onto the ℓ_1 ball, and this is provided to you on the github page. Verify that your solution x_τ agrees with x_{BP} to at least a few digits of accuracy.

Deliverables: include both a printed write-up of the main code, as well as code uploaded to the google drive. Please upload your gradient descent method from last week, but you do *not* need to include this in the print-out.

Problem 3: Now assume we do not know the right value of τ. For a given τ, we can find x_τ, and hence evaluate $\sigma(\tau)$. Our goal is to find the smallest τ such that $\sigma(\tau) = 0$, so that we are solving (BP). We can solve $\sigma(\tau) = 0$ via Newton's method:

$$\tau_{k+1} = \tau_k - \sigma(\tau)/\sigma'(\tau).$$

Using the results from Homework 9, solve for the right value of τ using a few iterations of Newton's method (each iteration requires a call to your proximal gradient descent code). Verify that you converge to the right τ found in the previous step, and that your final solution agrees with x_{BP}. Note: this approach was first developed in the paper Probing the Pareto frontier for basis pursuit solutions by E. van den Berg and M. P. Friedlander, SISC (31)2 2008.

Deliverables: include both a printed write-up of the main code, as well as code uploaded to the google drive.

Problem 4: Make the linear operators Ψ and Ψ^T (but do not make them explicit matrices, which would kill performance). Now, apply the method of Problem 3 using $A = \Psi D^T$ and $c = x$ with $b = \Psi(y)$ where y is the Handel audio signal. Recover c, and set $\hat{y} = D(c)$. Play the sound file $\hat{y}$. Have we done a good job recovering the sound?

Deliverables: include both a printed write-up of the main code, as well as code uploaded to the google drive. Include some brief discussion in your write-up.

Problem 5: Optional: we took, on average, one fourth of all entries from y. How much lower can you go and still faithfully recover the signal?

Problem 6: Optional: another method to solve (BP) is using the Augmented Lagrangian method, so implement this and compare with the result from Problem 1. See Nocedal and Wright's book and their "Penalty and Augmented Lagrangian" chapter. This approach is the same as the "Bregman iteration" approach of Bregman Iterative Algorithms for ℓ_1-Minimization with Applications to Compressed Sensing by Yin et al. 2008.

Rubric for the final project

The final project is due the last week of class, and consists of a 10 minute presentation and a short paper. There is no minimum length to the paper, but aim for 5 to 10 pages of text and figures. If the project has a computational aspect, you can upload the code to the google drive folder.

If you really want to, you can write a longer, much more detailed paper and skip the presentation.

All projects should be finished by the Monday of the last week of class, May 1. On that day, we will determine which groups present which day.

We encourage students to form groups (but three students/group max).

The project is worth 25% of the final grade in the class, and cannot be dropped.

The project will be graded according to the following rubric:

Component	Percent of grade	High standard	Medium standard
Clear non-trivial problem is formulated	15%	Problem is of suitable difficulty, and clearly defined	Problem a bit too easy or too hard, or ill-defined
Problem is well-motivated	10%	Problem is interesting, and the reasons why are clearly explained	Problem is interesting but this is not explained, or explanations are lacking
Problem involves significant amount of theory or computing	10%	Problem requires synthesis of material learned in class, though not necessarily at the level of a professional journal paper	Problem requires a bit of material learned in class, but not that much
Alternatives approaches are considered	15%	Several approaches are considered, and there is support behind the approach taken. Alternative approaches need not be exhaustively listed, but a few categories of alternative approaches should be described	A few but not enough alternatives are discussed, or, it is not clear why your approach was chosen
Final result achieved, or obstacles discussed	15%	Problem is solved, or it is made clear why it was infeasible to solve it	Problem is partly solved, but no discussion of why it wasn't fully solved
Generates valid conclusions	15%	Report or presentation makes concluding remarks that are useful for the reader/listener	No interesting final comments, other than a summary of work completed. No analysis
Communication (report and possibly presentation)	20%	Well-organized and clear, logic easy to follow. Words are precise, jargon is appropritae. Figures presented as necessary. Grammatically correct	Follows high standard most of the time