EECS 127/227AT Optimization Models in Engineering
Spring 2019 Homework 7

Release date: 10/17/19.
Due date: 10/24/19, 23:00 (11 pm). Please LaTeX or handwrite your homework solution and submit an electronic version.

Submission Format
Your homework submission should consist of a single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

1. General optimization

In this exercise, we test your understanding of the general framework of optimization and its language. We consider an optimization problem in standard form:

$$p^* = \min_{x \in \mathbb{R}^n} \; f_0(x) \;:\; f_i(x) \leq 0, \;\; i = 1, \ldots, m.$$

In the following we denote by $\mathcal{X}$ the feasible set. For the following statements, provide a proof or counter-example.

(a) [Short justification, 1–2 lines.] Any optimization problem can be expressed as one with a linear objective.

(b) [Short justification, 1–2 lines.] Any optimization problem can be expressed as one without any constraints.

(c) [Short justification, 1–2 lines.] Any optimization problem can be recast as a linear program, provided one allows for an infinite number of constraints.

(d) [Short justification, 1–2 lines.] If one inequality is strict at the optimum, then we can remove it from the original problem and obtain the same solution.

(e) [A few lines of justification.] If the problem involves the minimization over more than one variable, say y and x, then we can exchange the minimization order without altering the optimal value:

$$\min_x \min_y \; F_0(x, y) = \min_y \min_x \; F_0(x, y)$$

(f) [A few lines of justification.] If the problem involves the minimization of an objective function of the form

$$f_0(x) = \max_y \; F_0(x, y),$$

in which case

$$p^* = \min_{x \in \mathcal{X}} \max_y F_0(x, y)$$

then $p^* \geq d^*$, where

$$d^* := \max_y \min_{x \in \mathcal{X}} \; F_0(x, y),$$

where $\mathcal{X}$ is the feasible set of the original problem. *Hint:* consider the function $y \to \min_{x'} F_0(x', y)$ and a similar function of x.

2. Gradient Descent vs Newton-Raphson

In this problem, we will explore the performance (in terms of convergence properties) of first order and second-order optimization algorithms.
Gradient Descent (GD) is a first-order iterative optimization algorithm that uses the first derivative information to find the optimal value of a function.
Newton-Raphson (NR) Method, applied in optimization settings, is a second-order iterative algorithm that effectively finds the solution of the first derivative of a function, which is nothing but the optimal value of the function.

To find the optimal value of a function $f(x)$, we usually start with an initial guess x_0 and then iterate over till convergence properties are met.
Let the optimization problem in hand is

$$\min_{x \in \mathbb{R}^m} f(x)$$

Using gradient descent, the iteration step is,

$$x_{n+1} = x_n - \nabla f(x_n), \text{ for n = 0,1,2,}\cdots$$

Using Newton-Raphson method, the iteration step is,

$$x_{n+1} = x_n - [Hf(x_n)]^{-1} \nabla f(x_n), \text{ for n = 0,1,2,}\cdots$$

where $Hf(x_n)$ is the hessian of $f(x)$ computed at x_n

(a) For a paraboloid given by,

$$f(x) = x_1^2 + x_2^2 - 8x_1 + 2x_2 + 17$$

i. [Find algebraic expression.] Find the expression for the first derivative of $f(x)$, $\nabla f(x)$.
ii. [Find algebraic expression.] Find the expression for hessian of $f(x)$, $Hf(x)$.
iii. [Find numeric expression.] Compute the value of x^*, at which the optimum is achieved for $f(x)$.
iv. [Jupyter notebook.] With an initial assumption $x_0 = \begin{bmatrix} 8 \\ 3 \end{bmatrix}$, perform 100 iterations of gradient descent and Newton-Raphson with a step size = 0.9, in Jupyter Notebook. Plot the path taken by x in the 100 steps towards optimum for both the algorithms.
v. [A few sentences.] What did you observe about the path taken by x towards optimum for both the algorithms in this case?

(b) For a halfpipe given by,

$$f(x) = \cosh(\epsilon x_1^2 + x_2^2), \text{ where } \epsilon = 0.05, \text{ and cosh(x) is the hyperbolic cos function}$$

i. [Find algebraic expression.] Find the expression for the first derivative of $f(x)$, $\nabla f(x)$.
ii. [Find algebraic expression.] Find the expression for hessian of $f(x)$, $Hf(x)$.
iii. [Jupyter Notebook.] With an initial assumption $x_0 = \begin{bmatrix} -2 \\ 0.9 \end{bmatrix}$, perform 5000 iterations of gradient descent and Newton-Raphson with a step size = 0.1, in Jupyter Notebook. Plot the path taken by x in the 5000 steps towards optimum for both the algorithms.

iv. [A few sentences.] What did you observe about the path taken by x towards optimum for both the algorithms in this case?

(c) [A few sentences.] Which of the algorithms provides a more efficient path towards the optimum (x^*), starting from the same initial point? Justify your answer with proper reasoning.

Bonus: Change the step size in Jupyter Notebook for both the cases and notice the change in optimization paths. (Note: This will not be graded)

3. Gradient Descent × The Laplacian

We are given an undirected simple graph $G = (V, E)$ where $V = \{1, \ldots, n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges joining vertices. Since G is undirected, for $i, j \in V$, we have $(i, j) \in E \Leftrightarrow (j, i) \in E$. Let d_i be the degree of vertex $i \in V$, i.e. the number of edges incident to the vertex. We define the Laplacian matrix $L \in \mathbb{R}^{n \times n}$ as

$$L_{ij} = \begin{cases} d_i & i = j \\ -1 & (i, j) \in E \\ 0 & \text{otherwise} \end{cases}.$$

Equivalently, $L = D - A$, where D is the diagonal matrix defined by $D_{ii} = d_i$, and A is the adjacency matrix.

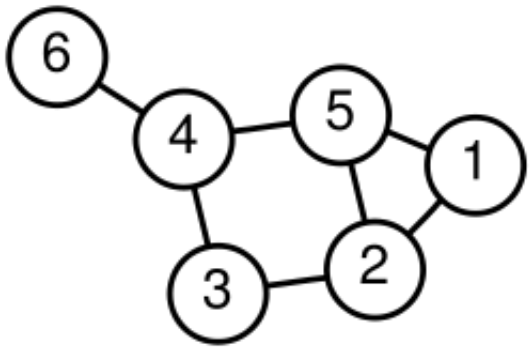

Figure 1: An undirected simple graph.

(a) [Write the matrix.] Form the Laplacian for the graph shown in Figure 1.

(b) [A couple lines of justification.] Show that in general, the Laplacian L for any undirected $G = (V, E)$ is symmetric.

(c) [Algebraic justification.] Show that for any $x \in \mathbb{R}^n$,

$$x^\top L x = \frac{1}{2} \sum_{(i,j) \in E} (x_i - x_j)^2.$$

Deduce that $L \succeq 0$.

(d) [Short justification and eigenvector.] Show that zero is always an eigenvalue of L. Exhibit a corresponding eigenvector.

(e) [Algebraic justification.] Consider the problem of assigning weights $x \in \mathbb{R}^n$ to each vertex $i \in V$ such that the sum of the weights is close to one and each pair of adjacent vertices has similar weights. That is, we want the solution to the optimization problem

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^n} \sum_{(i,j) \in E} (x_i - x_j)^2 + 2\lambda \left(\sum_{i \in V} x_i - 1 \right)^2$$

where $\lambda \in \mathbb{R}$ is a constant. Here, the first term inside the argmin penalizes adjacent vertices with highly differing weights, while the second term forces the sum of the weights to be close to one. Show that this optimization problem is equivalent to

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{1}{2} x^\top (L + \lambda \mathbf{1}\mathbf{1}^\top) x - \lambda \mathbf{1}^\top x$$

where $\mathbf{1}$ is the all-ones vector in $\mathbb{R}^n$.

(f) [Algebraic expression and short justification.] What is the optimal x^*?

(g) [Algebraic justification.] Suppose we use gradient descent with step size $\eta > 0$ to find the optimal x^*. Write the gradient descent step, i.e. express x_{k+1}, the $(k+1)$th step of gradient descent, in terms of x_k, L, η, and λ.

(h) [Algebraic justification.] Show that $x_{k+1} - x^* = (I - \eta(L + \lambda \mathbf{1}\mathbf{1}^\top))(x_k - x^*)$.

(i) [Algebraic justification.] Deduce that $\|x_k - x^*\|_2 \leq \rho^k \|x_0 - x^*\|_2$ for some $\rho \in \mathbb{R}$, where x_0 is the starting point of the gradient descent. Express ρ in terms of η, λ and the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$ of L. Assume λ is given such that $\lambda_1 > n\lambda > \lambda_{n-1}$.

(j) [Algebraic expression and justification.] Find the number of time steps gradient descent takes to converge to some $\varepsilon > 0$ around x^* as a function of η, assuming $\|x_0 - x^*\|_2 > \varepsilon$ (since we are not lucky enough to start gradient descent near the optimal point). That is, find $t(\eta)$ such that $\|x_{t(\eta)} - x^*\|_2 \leq \varepsilon$. Express $t(\eta)$ in terms of x_0, x^*, λ, and $\lambda_1, \ldots, \lambda_n$. Note that $t(\eta)$ should be an integer-valued function.

(k) [Algebraic expression and justification.] To maximize our rate of convergence, we find the optimal step size $\eta^* = \operatorname{argmin}_{\eta > 0} t(\eta)$. Express η^* and $t(\eta^*)$ in terms of $\lambda, \lambda_1, \ldots, \lambda_n$.

(l) [A few sentences.] It turns out the ratio $\kappa = \lambda_1/\lambda_{n-1}$ is inversely related to how connected the graph G is. For example, highly connected graphs such as the complete graph K_n or the hypercube have relatively small values of κ, while poorly connected graphs such as the 2D grid or a single path have large values of κ. (If interested, look into conductance of a graph and Cheeger's inequality). Assuming we choose the optimal step size η^*, how does the connectivity of the graph affect the rate of convergence? Do you expect gradient descent for graphs with higher values of κ to converge slower or faster than that for graphs with lower κ?

Hint: The function $f(x) = \dfrac{x-1}{x+1}$ is monotonically increasing for $x \geq 1$.

4. Fourier representation of images

In next week's homework, you will be using Fourier and Wavelet transforms in conjunction with LASSO optimization in order to do image compression. As there are no signals prerequisites for this class, we thought it might be beneficial to go over some of the basics that will make next week's assignment easier and make more sense. And, as this isn't a signals class, we will focus more on the linear algebra involved in signals theory, and just provide some broader context for what these transforms are doing. If these concepts seem interesting to you, we highly recommend you take some of the signals classes that Berkeley has to offer (EE120 and EE123 to name a few).

Fourier Transform Basics

The French mathematician Jean Baptiste Joseph Fourier discovered in 1822 that any periodic function can be expressed as the sum of sines and/or cosines of different frequencies, each multiplied by a different coefficient (called a Fourier series). It does not matter how complicated the function is; if it is periodic and satisfies some mild mathematical conditions, it can be represented by such a sum.

It was later discovered that functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighing function. The formulation in this case is the Fourier transform, and its utility is even greater than the Fourier series in many theoretical and applied disciplines. Both representations share the important characteristic that a function, expressed in either a Fourier series or transform, can be reconstructed (recovered) completely via an inverse process, with no loss of information.

This is one of the most important characteristics of these representations because it allows us to work in the Fourier domain and then return to the original domain of the function without losing any information[1].

There are many "flavors" of the Fourier transform, but we will focus on the most applicable one for this class: the Discrete Fourier Transform (DFT). The DFT is defined for a one dimensional, length N signal $x[n]$, as:

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi}{N} kn}$$

, where $x[n]$ is a signal in some time or spacial domain, and X_k is its representation in frequency domain. You may be wondering where this $e^{-\frac{j2\pi}{N} kn}$ comes from. Using Euler's formula, we see that this term is equivalent to $\cos(\frac{j2\pi}{N} kn) - j \sin(\frac{j2\pi}{N} kn)$, which is nothing more than a complex combination of cosines and sines (as was discussed above).

Now, how can we express this idea of a transform in more familiar, linear algebra terms? A one dimensional signal can be represented as a vector, and a **Fourier transform is just a change of basis from the unit vectors along each axis to these complex exponentials**.

(a) [Algebraic expression and justification.]. Assume we have a length 3 signal $x \in \mathbb{R}^3$. Using the formula above, express the Fourier transform, $X \in \mathbb{R}^3$ as $X = c \cdot \mathcal{F}x$, where $\mathcal{F} \in \mathbb{R}^{3\times 3}$ and $c \in \mathbb{R}$.

[1] Rafael C. Gonzalez, Richard E. Woods - Digital Image Processing-Prentice Hall (2008)

(b) [Algebraic expression.] Come up with a relationship for the i,j entry of $\mathcal{F}$ and each basis function $\phi = e^{-\frac{j2\pi}{N}}$

Another important property of the DFT matrix is that it is an **unitary** matrix (and equivalently and orthogonal change of basis). While the proof is not too complicated, it is somewhat out of the scope of the class. If you are interested you can check out the orthogonality of the DFT basis function proof and the orthogonality of DFT Matrix proof.

Fourier Transform and Images

Now that we have established the definition and math related to the Fourier transform, we can look at the more practical application of applying and using the Fourier transform with images.

Before building some intuition of the motivations of using the Fourier transform, a quick note that when dealing with images we use a 2D DFT. It is the same as a 1D DFT along each dimension, and all the properties discussed above still apply. Additionally, inputs are now matrices instead of vectors, which is convenient for our representation of images.

Now that we have some of the math and the background covered, it's time to cover the pertinent (at least to you) material - why are we discussing the DFT? Because the DFT is an orthonormal change of basis, it is an equivalent representation of our image. Moreover, in the Fourier domain, each "pixel" in the image corresponds to a different frequency present in each image. So, using the DFT, we can manipulate the various frequencies present in our image - and this enables us to do very powerful operations with some very simple code.

Finally, we will also be using the Wavelet transform next week. While the details are well beyond the scope of this class, you can think of the Wavelet transform as analogous to the DFT, but with an additional (and very useful) property — it is a sparse transformation. That is, natural signals are sparse in the Wavelet domain.

(c) [1–2 sentences.] What concept learned recently in class relates closely to the benefits of the Wavelet transform?

Now, take a look at the ipython notebook `image_transform_fundamentals`. You will not have to write any code, but it will walk you through some of the practical aspects of the topics we talked about, and help you build some intuition of the DFT.

Note: The notebook imports `pywt`. Please install this package with `pip install pywavelets`. (*Don't* `pip install pywt` *– this is deprecated*).