Problem set 1
Convex Optimization (Winter 2018)

Due Friday, January 12th at 5pm

- Upload a pdf of your work to Canvas by the due date. Late homeworks will be penalized 25% per day and will not be accepted after Monday (1/15) at 5pm unless you have made arrangements with us well before the due date. We strongly encourage you to typeset your solutions.
- **This assignment will also be used to evaluate your command of the course prerequisites and readiness for the class. As a result, we ask that you complete this assignment *on your own*, without consulting your peers. Collaboration will be allowed for all future assignments.**
- The material needed for this homework will be covered in lectures 1 and 2

Required Programming Exercise (material covered in lecture 1)

In this required exercise, you will experiment with the bisection search algorithm and get familiar with ways of implementing oracle accesses. Starter Python code is provided on Canvas. In order to compile and run the code, you need to install Python3 and the NumPy package. An easy way to set up both of them is to install *Anaconda*, a free Python distribution that includes many Python packages for science, math, and data analysis.

1. Complete the bisection search code in the file "optimization.py" by filling in the lines marked with "TODO" comments.
2. Double-check that your code works by using your completed bisection search code to find an ϵ-suboptimal solution to the problem of minimizing a function of which you know the minimum (e.g. $f(x) = x^2 - 3x + 4$).
3. Use your bisection search code to find the optimum of the following functions: *weird_func* (defined in "optimization.py") on the interval $[-5, 5]$ up to suboptimality $\epsilon = 10^{-15}$, and

$$f(x) = x^4 + x^2 + 3 + (x-3)e^{x-2} + 2\sin(x) - x\cos(x)$$

on the interval $[-1000, 1000]$ up to suboptimality $\epsilon = 10^{-15}$. For each function, record the suboptimal point, the function value, and the number of iterations needed in your problem set writeup. Turn in all of the code you used.

From the programming exercise, you should turn in the following:

1. Your code in the optimization.py file
2. In your problem set writeup, include the suboptimal points, function values, and numbers of iterations from part 3 above.

Required Written Problems (material covered in lecture 2)

All of the problems in this section are required.

1. Consider the ℓ_1 unit ball $B_1 = \{x \in \mathbb{R}^n : \|x\|_1 = \sum_{i=1}^n |x_i| \leq 1\}$ and the ℓ_∞ unit ball $B_\infty = \{x \in \mathbb{R}^n : \|x\|_\infty = \max_i |x_i| \leq 1\}$.
   (a) Show that B_1 is a convex polyhedron by explicitly expressing it as an intersection of halfspaces. How many halfspaces are required in order to express B_1?
   (b) Explicitly express B_1 as a convex hull of a finite number of points. How many points are required in this characterization?
   (c) Contrast this with the ℓ_∞ unit ball. Explicitly express B_∞ as an intersection of halfspaces and as the convex hull of a finite number of points. How many halfspaces are needed? How many points are needed?
   (d) Consider $\hat{x}_1 = [0.1, -0.2, 0.6, -0.1]$ and $\hat{x}_2 = [0.4, 0, 0.5, -0.1]$ on the boundary of B_1. Identify the set of all supporting hyperplanes of B_1 at $\hat{x}_1$ and $\hat{x}_2$. What is the dimensionality of each set?

2. We would like to define the direction of steepest descent with respect to a norm $\|\cdot\|$ at a point x. A first attempt at a definition might be

$$\triangle x = \lim_{t \to 0} \operatorname*{argmin}_{\|v\|=t} f(x+v)$$

   However this does not work because if the argmin is not unique, then the limit is not well defined. If we assume that the argmin is unique, and that f is smooth, then we can rearrange the limit and argmin and say

$$\begin{aligned} \triangle x &= \lim_{t \to 0} \operatorname*{argmin}_{\|v\|=t} f(x+v) = \operatorname*{argmin}_{\|v\|=1} \lim_{t \to 0} \frac{f(x+tv) - f(x)}{t} \\ &= \operatorname*{argmin}_{\|v\|=1} \nabla f(x)^T v \end{aligned}$$

   This last expression is well-defined even in the case that the argmin is not unique, or f is not smooth. Therefore, we define the directions of steepest descent on f with respect to $\|\cdot\|$ at a point x as:

$$\triangle x = \operatorname*{argmin}_{\|v\|=1} \nabla f(x)^T v$$

   where there may be many directions of steepest descent descent if there are many minimizers.

(a) For $H \in \mathbb{R}^{n\times n}, H \succ 0$, consider the norm $\|x\|_H^2 = x^T H x$. What are the direction(s) of steepest descent on f with respect to this norm at a point x?

(b) What are the direction(s) of steepest descent on f w.r.t. the ℓ_1 norm at a point x?

(c) What are the direction(s) of steepest descent on f w.r.t. the ℓ_∞ norm at a point x?

3. We will define strong convexity more generally than in Boyd and Vandenberghe (Section 9.1.2). In particular, our definition is also valid for non-differentiable functions.

Definition: A function $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex if for every $x, y \in \mathbb{R}^n$ and every $\theta \in [0, 1]$:

$$f((1-\theta)x + \theta y) \leq (1-\theta)f(x) + \theta f(y) - \frac{\mu}{2}\theta(1-\theta)\|x-y\|_2^2$$

(a) Prove that a continuously differentiable function f is μ-strongly convex if and only if for every $x, y \in \mathbb{R}^n$,

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|_2^2 .$$

This generalizes the first order characterization of convexity (Section 3.1.3).

(b) (Do not turn in) prove a more general statement: a function is μ-strongly convex if and only if at every point x in the domain there exists a subgradient ∇x for which the above holds).

Note that for an optimum $x^* = \text{argmin} f(x)$, the above inequality implies that for any point $y \in \mathbb{R}^n$:

$$f(y) \geq f(x^*) + \frac{\mu}{2}\|y-x^*\|_2^2 .$$

so the optimum is unique and any ϵ-suboptimal point must be close to the optimum. This is different from non-strongly-convex functions, which might have many optima, or ϵ-suboptimal points that are arbitrarily far away from an optimum.

(c) (Do not turn in) Prove that a twice continuously differentiable function f is m-strongly convex if and only if its domain and convex and for every $x \in \mathbb{R}^n$, all eigenvalues of the Hessian at x are greater or equal to μ, i.e.:

$$\nabla^2 f(x) \succcurlyeq \mu I.$$

This generalizes the second order characterization of convexity (Section 3.1.4) and is the definition used in Section 9.1.2.

(d) Provide an example of a function $f : \mathbb{R} \mapsto \mathbb{R}$ that is strongly convex but not everywhere differentiable.

Optional Book Exercises

We encourage you to work through the following exercises from Boyd and Vandenberghe, but they are optional and should not be turned in.
Problems: 2.12, 2.15, 3.6, 3.16, 3.18, 3.24, 3.26.

Extension Problems

The problems in this section are optional and go beyond the material covered in class and that is required for class completion. They are recommended for students interested in exploring additional topics, getting a more rigorous treatment of the material, or doing research in optimization or related areas. Solutions will *not* be thoroughly graded together with the rest of the homework, but we encourage you to submit your work and we will record it and provide feedback.

0^{th} Order Optimization

We want to optimize a convex, L-Lipschitz function $f : [0, 1] \mapsto \mathbb{R}$ given an oracle that evaluates f at a point but does not return gradient information. That is, given x the oracle returns only $f(x)$.

1. Give an algorithm for finding an ϵ-suboptimal point for f. Analyze its query and runtime complexity.

   With you algorithm in hand, it is time to check if your algorithm is optimal in terms of query complexity. We want to be able to say the *no* algorithm can perform better, but we will start by showing that your analyzis of your suggested algorithm is tight.

2. Prove a lower bound on the number of queries *required by your algorithm* by constructing a specific function that requires your algorithm to make a large number of queries before it encounters an ϵ-suboptimal point. That is, prove a statement of the form "For every ϵ, there exists a convex function $f : [0, 1] \mapsto \mathbb{R}$ that is 1-Lipschitz, but that running my algorithm on it for less then such and such queries, the algorithm will not encounter an ϵ-suboptimal point".

   Next, we need to show that for any algorithm we can find a such a function.

3. Show that your algorithm is (nearly) optimal, by proving a statement of the form: "For any ϵ and any algorithms A that uses only a function evulation oracle, there exists a 1-Lipschitz convex function $f : [0, 1] \mapsto \mathbb{R}$ such that in its first such and such queries, the algorthm does not encounter a point that is ϵ-suboptimal".

Convexity, Log-Concavity, and Quasi-Convexity

Convexity is actually not the weakest notion possible that still allows for efficient optimization algorithms. It is possible to define more relaxed notions that still capture some of the benefits of convexity. In particular, consider the following two definitions:
Definition: A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is log-concave if for all $\theta \in [0, 1]$ and all $x, y \in \mathbb{R}^n$

$$f(\theta x + (1 - \theta)y) \geq f(x)^{\theta} f(y)^{1-\theta}$$

Definition: A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is quasi-convex if for all $\theta \in [0, 1]$ and all $x, y \in \mathbb{R}^n$

$$f(\theta x + (1 - \theta)y) \leq \max\{f(x), f(y)\}$$

1. Prove that for any function f, f is convex $\implies$ $-f$ is log-concave $\implies$ f is quasi-convex
2. Give an example of a function which is quasi-convex but not log-concave
3. Give an example of a function which is log-concave but not convex
4. Would bisection search or center-of-mass methods work for log-concave or quasi-convex functions? If so, what is the query complexity?