15-381: Artificial Intelligence: Representation and Problem Solving (Fall 2018)

Homework 3: Optimization

Release: September 20, 2018,
Due: October 2, 2018, 11:59pm

1 Problem Formulation [5+5 = 10 Points]

Formulate the following as an optimization problem, and explain the meaning of the variables and constraints. For each of them, determine if it is a convex optimization problem.

1) Find the dimensions (radius, height) of a cylinder with a given surface area A that has the largest volume V.
2) Given a set of points S = $\{x_1, ..., x_m\}$ where $x_i \in \mathbb{R}^n$, find a point in $\mathbb{R}^n$ that has the minimum average Euclidian distance to all points in S.

2 Convex Functions [5+5 = 10 Points]

For each of the following claims, prove it to be true, or provide a counterexample.

1) Let $f(x) : \mathbb{R}^n \to \mathbb{R}, g(x) : \mathbb{R}^n \to \mathbb{R}$ be convex functions. $h(x) = f(x)g(x)$. Claim: $h(x)$ is convex.
2) Let $f(x) : \mathbb{R} \to \mathbb{R}, g(x) : \mathbb{R} \to \mathbb{R}$ be convex functions. $h(x) = f(g(x))$. Claim: $h(x)$ is convex if $f(x)$ is nondecreasing.
3) **BONUS [5 Points]:** The convex envelope of a function f (not necessarily convex), denoted conv f, is defined as the largest convex function majorized by f. Prove that conv $sin(x) = -1$.

3 Gradient Descent [5+5+5+5+5 = 25 Points]

In this problem we will be attempting to show that iterative gradient descent steps will lead us to the global minimum of the objective function $f : \mathbb{R}^n \to \mathbb{R}$. To do so, we will make two assumptions about f in this case.

Assumption 1 f is convex

Assumption 2 The gradient of f is *Lipschitz continuous with constant* $L > 0$, i.e. for all $x, y \in \mathbb{R}^n$, we have:

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2.$$

Note: $\|x\|_2$ denotes the Euclidean norm of the vector x.

1) To begin, given a function f that fits the above criteria, provide the first order Taylor expansion of the function at some point y.

2) Prove the following inequality for $x, y \in \mathbb{R}^n$,

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|x - y\|_2^2. \tag{1}$$

3) Now, let us fix a step size of $\frac{1}{L}$. That is, from any x, we move to

$$x' = x - \frac{1}{L}\nabla f(x) \tag{2}$$

during gradient descent. Given that Equation (1) holds, derive the following inequality for the amount of progress gradient descent makes in locally decreasing the value of f:

$$f(x') \leq f(x) - \frac{1}{2L}\|\nabla f(x)\|_2^2. \tag{3}$$

4) To analyze the convergence of gradient descent, we want to bound the difference between the current value of the objective $f(x')$ and the global minimum $f(x^\star)$. Show that the following upper bound on $f(x') - f(x^\star)$ holds:

$$f(x') - f(x^\star) \leq \frac{L}{2}(\|x - x^\star\|_2^2 - \|x' - x^\star\|_2^2). \tag{4}$$

Hints:

- What can you use to upper bound $f(x)$ in terms of $f(x^\star)$?
- $2a^Tb + \|b\|_2^2 = \|a\|_2^2 + 2a^Tb + \|b\|^2 - \|a\|_2^2 = \|a + b\|_2^2 - \|a\|_2^2$.

5) Let $x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}$ be a sequence of gradient descent updates starting from some initial point $x^{(0)}$, with $x^{(t+1)} = x^{(t)} - \frac{1}{L}\nabla f(x^{(t)})$. Prove the following two inequalities to show the process converges:

$$f(x^{(\tau)}) - f(x^\star) \leq \frac{1}{\tau}\sum_{t=1}^{\tau}(f(x^{(t)}) - f(x^\star)) \leq \underbrace{L\frac{\|x^{(0)} - x^\star\|_2^2}{2}}_{\text{constant}} \times \frac{1}{\tau}. \tag{5}$$

4 Linear Programming [5+5+5 = 15 Points]

Consider the following problem:

There are n sellers, each selling a divisible item. The items all have weight 1. There are m buyers, each carrying a bag with capacity 1. The ith item has a value of v_{ij} to the jth buyer. How do you allocate the items such that the total value of allocated items is maximized?

1) Formulate the above problem as a LP.

2) Write down the dual of the above LP.

3) Now, assume there is a new seller (now $n + 1$ sellers in total). How can we tell if the current solution is still optimal without resolving a LP?

   Hint: Consider the definition of the dual.

5 Programming 1: Simplex Algorithm [25 Points]

For this problem, you will be implementing the simplex algorithm in Python. Your code should go in `simplex.py`. You are **not** allowed to use any convex optimization library for this problem.

5.1 Format

Implement the basic simplex algorithm as described in lecture. You should use the `numpy` package for matrix manipulations. If you haven't used `numpy` before, some examples of common operations are provided for you in `numpy_examples.py`.

The inputs to your `simplex` method will be in the form of numpy arrays and matrices:

- I: an array consisting of a feasible basis index set.
- c: a cost vector.
- A, b: a matrix-vector pair; your solution x should satisfy $Ax = b$.

Your output should be a tuple consisting of:

- v: the value of the optimal solution.
- x: the optimal solution.

5.2 Testing

We will grade your code based on whether it is able to obtain the optimal solution to a number of different linear programs. (Your code does not need to deal with infeasible or unbounded problems.) You have been given a set of test cases in the directory `simplex_tests` as well as `handout_autograder.py`, which you can use to test your code. Notice that because we are doing numerical computations, computations that should actually yield 0 will often be approximately 0 (e.g., in $[-10^{-15}, 10^{-15}]$). To avoid problems with this, whenever you want to check if a number is strictly negative, you should check if it is $< -10^{-12}$ and whenever you want to check if a number is ≥ 0 you should check if it is $> -10^{-12}$.

6 Programming 2: MILP Representation [15 Points]

For this problem, you will be using a third party (`Gurobi`) solver to optimize a given problem. You can find the download link here. Please note that Gurobi, because it is proprietary software, requires a free academic license to run that can be found here. To install the associated python package (needed for imports), follow the instructions given here for Linux, here Windows, and here for Mac. Your code should go in `milp.py`. Your code will be tested for speed and correctness on test cases.

6.1 Problem Description

No Poverty and Zero Hunger are listed as the top two Sustainable Development Goals by the United Nations. Food rescue service provides a promising way to reduce food waste, overcome food insecurity and improve environmental sustainability. Organizations providing food rescue service rescue the surplus food from different food providers and re-distributing to local communities that are in need of food.

Now you are asked to help a food rescue organization FS to decide how to re-distribute the food in an efficient way. The problem is abstracted in the following way: There are M food providers (referred to as providers) and N local communities in need of food (referred to as communities). There are K type of food. Provider i has $A_{ik} \in \mathbb{N}$ unit of food k, and community j needs $B_j\mathbb{N}$ unit of food in total, but it may have some special needs on the type. $C_{jk}\mathbb{N}$ represents the minimum amount of food of type k community j needs. The food sent to a community has to come from a single provider. The transportation cost per unit of food from provider i to community j is T_{ij}.

Given M, N, K, A, B, C, T, you are asked to (i) determine whether it is feasible to satisfy all communities' need, and (ii) if feasible, provide an optimal re-distribution plan that minimizes total transportation cost.

6.2 Format

The inputs to your `setup` method will be in the form of numpy arrays and matrices:

- I: An array of the form $[M, N, K]$
- A: The provider food stores matrix. $A[i][k]$ is the amount of food type k provider i has
- B: The community food needs array. $B[j]$ is the minimum of food (of any type) community j needs
- C: The community food needs matrix. $C[j][k]$ is the amount of food type k community j needs
- T: The transportation cost matrix. $T[i][j]$ is the cost to transport one food unit from provider i to community j

Your output should be:

- m: the `Gurobi` model of the problem

6.3 Testing

We will grade your code based on whether it is able to obtain the optimal solution to a number of different constraints, as well as if the constraints given are feasible. You have been given a set of test cases in the directory `simplex_tests`. `handout_autograder.py` can be used to grade your work.

If you wish to test on additional/specific test cases, we have provided a utility (`text_to_npz.py`) that converts test cases in txt form, such as the sample provided in `sample_test_case.txt` into the `.npz` format the autograder uses for testing. `test_milp.py` will then test your `milp.py` on the `.npz` file provided to it as a command line argument.