CS2601 Linear and Convex Optimization

Homework 6

Due: 2021.11.11

For this assignment, you should submit a **single** pdf file as well as your source code (.py or .ipynb files). The pdf file should include all necessary figures, the outputs of your Python code, and your answers to the questions. Do NOT submit your figures in separate files. Your answers in any of the .py or .ipynb files will NOT be graded.

In this assignment, you will implement gradient descent with constant step size. First complete the function `gd_const_ss` in `gd.py`.

1. Consider the following quadratic program,

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \tag{1}$$

where

$$\boldsymbol{Q} = \begin{pmatrix} \gamma & 0 \\ 0 & 1 \end{pmatrix}.$$

with $\gamma > 0$. In terms of coordinates,

$$f(x_1, x_2) = \frac{\gamma}{2}x_1^2 + \frac{1}{2}x_2^2.$$

We know the optimal solution is $\boldsymbol{x}^* = \mathbf{0}$ and $f^* = f(\boldsymbol{x}^*) = 0$.

(a). What is the smallest L such that f is L-smooth?

(b). What is the largest m such that f is m-strongly convex?

(c). Suppose $\gamma = 0.1$. Complete **p1.py** and run the gradient descent algorithm you implemented with $\boldsymbol{x}_0 = (1, 1)$ and step sizes 2.2, 1, 0.1, 0.01. (Note: for step size 2.2, you may want to limit the maximum number of iterations to a small number, say 10.) Does it converge in each case? When it does converge, how many iterations does it take? In each case, plot the 2D trajectory of the sequence $\boldsymbol{x}_k$ and the function values $f(\boldsymbol{x}_k)$. You can use the function provided in `utils.py` or write your own code. Note that $f(\boldsymbol{x}_k)$ is also the suboptimality gap since $f(\boldsymbol{x}^*) = 0$.

(d). For $\gamma = 1, 0.1, 0.01, 0.001$, run gradient descent with step size 1. How many iterations do you need in each case? How does the number of iterations change as γ decreases? You don't need to show the plots.

2. Linear regression. Consider the least squares problem in Problem 4(a) of Homework 5, i.e.

$$\min_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

where

$$\boldsymbol{X} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix}.$$

Use your implementation of gradient descent to solve this least squares problem. You may reuse the codes for Problem 1 by rewriting `f` and `fp` (you don't have to plot the traces as in Problem 1). Use any stepsize you find appropriate. Compare your solution with the solution you found in HW5, and the solution you find by solving the normal equation using `np.linalg.solve`. Do they agree with each other?

3. Logistic regression. In this problem, you will train a logistic regression classifier on a toy dataset given in **p3.py**. The features are two dimensional, so the i-th sample is $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^2 \times \{+1, -1\}$. As in Problem 2 of Homework 1, we will append a constant 1 to each $\boldsymbol{x}_i$ to make it three dimensional, and absorb the bias term b into $\boldsymbol{w}$ by defining a new $\boldsymbol{w} = (w_1, w_2, b)^T \in \mathbb{R}^3$. The objective function to be minimized is

$$f(\boldsymbol{w}) = \sum_{i=1}^{m} \log(1 - e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}})$$

where m is the number of samples. Its gradient is found in Problem 2(c) of HW1,

$$\nabla f(\boldsymbol{w}) = -\sum_{i=1}^{m} [1 - \sigma(y_i \boldsymbol{x}_i^T \boldsymbol{w})] y_i \boldsymbol{x}_i$$

where $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function.

Find the optimal $\boldsymbol{w}^*$ using your implementation of gradient descent in `gd.py`. Report the classification accuracy of the classifier on the training dataset given by

$$\text{accuracy} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i = \hat{y}_i\}$$

where $\hat{y}_i = \text{sgn}(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle)$. You can follow the guidance of p3.py or write your own code. Use matrix operations to calculate the function values and gradients. You may need some of the following functions, `numpy.matmul`, `numpy.multiply`, `numpy.exp`, `numpy.reshape`.

4. Suppose $f(\boldsymbol{x})$ is differentiable and α-strongly convex, and $g(\boldsymbol{x})$ is β-smooth. Show that the function $h(\boldsymbol{x}) = f(\boldsymbol{x}) - g(\boldsymbol{x})$ is convex if $\alpha \geq \beta$.