COMPSCI 514: Optional Problem Set 5

Due: December 8th by 11:59pm in Gradescope. Submissions accepted until December 11th at 11:59pm without penalty.

The core problems on this problem set are optional. If you complete them, the core problem grade will replace the lowest core problem grade on the previous problem sets. The challenge problems will count as normal challenge problems.

Instructions:

- Each group should work together to produce a single solution set. One member should submit a solution pdf to Gradescope, marking the other members as part of their group.
- You may talk to members of other groups at a high level about the problems but not work through the solutions in detail together.
- You must show your work/derive any answers as part of the solutions to receive full credit.

1. Convex Functions and Sets (12 points)

1. (2 points) Let $f_1, \ldots, f_n : \mathbb{R}^d \to \mathbb{R}$ be functions such that f_i is G_i-Lipschitz. Prove that $F(\vec{x}) = \sum_{i=1}^n f_i(\vec{x})$ is G-Lipchitz for $G = \sum_{i=1}^n G_i$.
2. (2 points) Let $f_1, \ldots, f_n : \mathbb{R}^d \to \mathbb{R}$ be convex functions. Prove that $F(\vec{x}) = \sum_{i=1}^n f_i(\vec{x})$ is convex.
3. (2 points) Let f and g be convex functions. Is their product $f \cdot g$ a convex function? Either prove that it is or give a counterexample.
4. (2 points) Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function. Is its derivative f' convex? Either prove that it is or give a counterexample.
5. Consider the minimum cut problem on a graph G with n nodes and Laplacian $\mathbf{L}$. We have seen that this problem is equivalent to solving:

$$\min_{\substack{\mathbf{x} \in \{-1,1\}^n \\ \mathbf{x} \neq [1,1,\ldots,1],\ \mathbf{x} \neq [-1,-1,\ldots,-1]}} c(\mathbf{x}) \text{ where } c(\mathbf{x}) = \mathbf{x}^T \mathbf{L} \mathbf{x}.$$

   (a) (2 points) Prove that the objective function $c(\mathbf{x}) = \mathbf{x}^T \mathbf{L} \mathbf{x}$ is convex. **Hint:** It may be helpful to write $\mathbf{x}^T \mathbf{L} \mathbf{x}$ as a sum of terms and use part (2) here.

   (b) (2 points) Is min-cut a convex optimization problem over a convex constraint set?

2. Gradient Descent Example (10 points)

At the beginning of the year, your restaurant needs to decide how many delivery drivers to hire. Each driver costs S dollars per year. Each driver can delivery 50 orders per day. If you receive more orders in a given day than your drivers can handle, you must pay an outside company r dollars to perform each excess delivery.

1. (2 points) Formulate your total cost for the year as a function of x, the number of drivers that your hire. The function will depend on the number of orders each day, given by $o_1, \ldots, o_{365}$ (these may not be known ahead of time, but can be estimated accurately from last year's data).

2. (2 points) Prove that this cost function is convex. **Hint:** You may want to use Problem 1.2 here.

3. (2 points) What is the gradient descent update with stepsize η for this function? **Note:** There are some points where the derivative of the function will not be defined. For simplicity you can ignore these points – find an equation for the derivative (i.e., the gradient) that can be evaluated everywhere and is correct wherever the derivative is defined..

4. (2 points) Prove that the cost function is G-Lipschitz for as small a G as you can. G will depend on the problem parameters.

5. (2 points) Does the update given in step (3) make intuitive sense? Describe in a few sentences.

3. Understanding Convergence in Gradient Descent (8 points)

1. (3 points) In our analysis of gradient descent in class, we show that $f(\hat{\theta}) - f(\vec{\theta}_*) \leq \epsilon$ for the best iterate $\hat{\theta} = \arg\min_{\vec{\theta}_1,\ldots,\vec{\theta}_t} f(\vec{\theta}_i)$. Prove that if we instead set $\bar{\theta} = \frac{1}{t}\sum_{i=1}^{t} \vec{\theta}_i$ (i.e., $\bar{\theta}$ is the average iterate) then we also have $f(\bar{\theta}) - f(\vec{\theta}_*) \leq \epsilon$. This strategy is often used, e.g., when using stochastic gradient descent for large datasets, since determining the best iterate can be much more expensive than just storing a running average. **Hint:** Use convexity to argue that $f(\bar{\theta}) \leq \frac{1}{t}\sum_{i=1}^{t} f(\vec{\theta}_i)$. Then plug this into the analyis shown in class.

2. (3 points) Suppose in gradient descent that I want to ensure that the function value never increases by more than ϵ in any given step. I.e., that $f(\vec{\theta}_{i+1}) - f(\vec{\theta}_i) \leq \epsilon$ for all i. Assuming that f is G-Lipschitz, how small should I set the step size η to ensure this? **Hint:** Use the two equivalent notions of Lipschitzness: $\|\vec{\nabla} f(\vec{\theta})\|_2 \leq G$ for all $\vec{\theta} \in \mathbb{R}^d$ and $|f(\vec{\theta}_1) - f(\vec{\theta}_2)| \leq G \cdot \|\vec{\theta}_1 - \vec{\theta}_2\|_2$ for all $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^d$.

3. (2 point) How does the step size in part (2) compare to the step size for the analysis shown in class where we set $\eta = \frac{R}{G\sqrt{t}}$ and $t = \frac{R^2G^2}{\epsilon^2}$? Describe in a few sentences why this makes intuitive sense.

Challenge Problems

C1. Gradient Descent with a Decaying Step Size 

In class we showed that gradient descent with step size $\eta = \frac{R}{G\sqrt{t}}$ converges to an ϵ approximate minimizer in $t = \frac{R^2G^2}{\epsilon^2}$ steps, for a convex G-Lipschitz function starting from an initial point $\vec{\theta}_1$

within a radius R of the optimum. This fixed step size analysis requires that we pick ϵ ahead of time and set η based on ϵ. However, in many applications we don't want to fix ϵ, but want to attain higher and higher accuracy as we run for longer. Here, we will analyze a variant of gradient descent with a gradually decreasing step size that allows us to do this.

Consider gradient descent with the update $\vec{\theta}_{i+1} = \vec{\theta}_i - \eta_i \vec{\nabla} f(\vec{\theta}_i)$, where the step size is set as

$$\eta_i = \frac{f(\vec{\theta}_i) - f(\vec{\theta}_*)}{\|\vec{\nabla} f(\vec{\theta}_i)\|_2^2}.$$

Note that using this step size requires knowledge of $f(\vec{\theta}_*)$, but not of $\vec{\theta}_*$, which may be reasonable in some settings. More complex approaches can remove the need to know this value.

1. Let $d_i = f(\vec{\theta}_i) - f(\vec{\theta}_*)$ be our error at step i. Prove that with the above step size:

$$d_i^2 \leq G^2 \cdot \left(\|\vec{\theta}_i - \vec{\theta}_*\|_2^2 - \|\vec{\theta}_{i+1} - \vec{\theta}_*\|_2^2 \right).$$

   Hint: Start with the single step analysis shown in class, applied with step size η_i.

2. Argue via Cauchy-Schwarz that $\frac{1}{t}\sum_{i=1}^t d_i \leq \frac{1}{\sqrt{t}}\sqrt{\sum_{i=1}^t d_i^2}$.

3. Use parts (1) and (2) to show that after t steps:

$$\frac{1}{t}\sum_{i=1}^{t} \left[f(\vec{\theta}_i) - f(\vec{\theta}_*) \right] \leq \frac{GR}{\sqrt{t}}.$$

4. Conclude that for any $\epsilon > 0$, after $t = \frac{G^2 R^2}{\epsilon^2}$ steps, letting $\hat{\theta} = \arg\min_{\vec{\theta}_1, \ldots, \vec{\theta}_t} f(\vec{\theta}_i)$,

$$f(\hat{\theta}) - f(\vec{\theta}_*) \leq \epsilon.$$

C2. Online Optimization and Portfolio Management

Adapted from HW3 of https://www.cs.princeton.edu/courses/archive/fall16/cos521/.

Assume that you have a fixed sum of money that you would like to invest in n stocks to maximize your total return over T days. For day $t = 1, 2, \ldots, T$, let $r_i^{(t)}$ be the relative price change of stock i on day t. I.e.,

$$r_i^{(t)} = \frac{\text{Price of stock } i \text{ on day } t}{\text{Price of stock } i \text{ on day } t-1}.$$

Let $\vec{r}^{(t)} \in \mathbb{R}^n$ be the vector with i^{th} entry equal to $r_i^{(t)}$.

One common portfolio management strategy is *Constant Rebalanced Portfolio* (CRP): decide on a fixed proportion of money to put into each stock and buy/sell individual stocks each day to maintain this proportion. Intuitively, when the price of a stock drops, you will buy more shares to maintain its portion of the portfolio, and when the price goes up, you will sell, causing the strategy to generally 'buy low and sell high'.

Let Δ^n be the n-dimensional simplex: the set of all vectors $\vec{x} \in \mathbb{R}^n$ with $\vec{x}(i) \geq 0$ for all i and $\sum_{i=1}^n \vec{x}(i) = 1$. That is, the set of all valid allocations of a portfolio's value to n stocks. For a portfolio allocation $\vec{x} \in \Delta^n$, we can see that the return on day t is given by $\langle \vec{r}^{(t)}, \vec{x} \rangle$ and the total return after T days (assuming 0 transaction costs) is:

$$\prod_{t=1}^{T} \langle \vec{r}^{(t)}, \vec{x} \rangle.$$

1. Show that finding a portfolio allocation maximizing the return over T days is equivalent to finding:

$$\vec{x} \in \arg\min_{\vec{x} \in \Delta^n} f(\vec{x})$$

where $f(\vec{x}) \stackrel{\text{def}}{=} \sum_{t=1}^{T} -\log(\langle \vec{r}^{(t)}, \vec{x} \rangle)$. Show that this optimization problem is a convex optimization problem over a convex constraint set, and hence can be approximately minimized with projected gradient descent.

Hint: You may use that for a scalar function $f : \mathbb{R} \to \mathbb{R}$, if the second derivative $f''(y)$ is positive for all y, then f is convex.

2. What is $\vec{\nabla} f(\vec{x})$?

Download the S&P 500 stock data on the course page, which contains prices for 490 stocks over 1000 trading days from 2001-2005.

3. If an investor places all their money in the single best stock, what is the return obtained over 1000 days (as a percentage of the initial quantity invested)? What if they place an equal amount of money initially in each stock? Finally, what if they use a CRP strategy that allocates an equal percentage of the portfolio to each stock (i.e., uses allocation $\vec{x}$ with $\vec{x}(i) = 1/n$ for all i)? **Hint:** Start by using the price data to compute $\vec{r}^{(t)}$ for $t = 1, \ldots, 999$ (let the first price be the price at day 0).

4. Implement projected gradient descent to find a CRP allocation $\vec{x}$ that performs better than the single best stock over 1000 days. Report the total return when using this allocation and list the 10 stocks that are allocated the largest percentage of the portfolio (along with their respective allocations). If fewer than 10 stocks have nonzero allocations, just list those with nonzero allocations. The following may be helpful to implement the projection step onto Δ^n: https://czxttkl.com/2020/12/23/projected-gradient-descent/. You may try a number of different step sizes η, reporting the best result obtained. Write a couple sentences discussing how you picked η and the iteration count.

5. The above optimization of course requires knowing the stock prices ahead of time, so isn't of much use in practice. What we can do instead is to implement an *online* version of gradient descent to find a portfolio re-balancing strategy that an investor could actually use. In this strategy, we will use a different allocation $\vec{x}^{(t)}$ at each day t. To obtain $\vec{x}^{(t+1)}$ from $\vec{x}^{(t)}$, we perform a gradient update based on the return at day t. In particular, we first set $\vec{x}^{(t+1)} = \vec{x}^{(t)} - \eta \vec{\nabla} f_i(\vec{x}^{(t)})$, where $f_i(x) = -\log(\langle \vec{r}^{(t)}, \vec{x} \rangle)$. We then project back to the constraint set.

   Implement this approach and compare its performance to the static allocation of part (4) and to the baselines of part (3). Again, you may try a number of different step sizes η, reporting the best result obtained. In reality, η itself would be learned based on prior data or in an online way. Write a couple sentences discussing how you picked η and how your implementation differs from the gradient descent implementation of part (4).

C3. Gradient Descent, Linear Systems, and the Power Method

Consider any $\mathbf{A} \in \mathbb{R}^{n \times d}$ with SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Consider the least squares regression problem, where, given a vector $\mathbf{b} \in \mathbb{R}^n$, the goal is to find:

$$\mathbf{x}_* \in \arg\min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2. \tag{1}$$

Recall from Problem Set 4 that one can compute $\mathbf{x}_*$.in closed form as $\mathbf{x}_* = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}$. In this problem we will look at a faster approximation method related to gradient descent and the power method.

1. Define the least squares regression function as $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$. Is $f(\mathbf{x})$ G-Lipschitz for some finite G? Why or why not? Can the gradient descent analysis shown in class be applied? **Hint:** Compute the gradient as a function of $\mathbf{A}, \mathbf{b}$, and $\mathbf{x}$. This will be useful in part (2) as well.

2. For some stepsize η, let $\mathbf{B} = \mathbf{I} - 2\eta\mathbf{A}^T\mathbf{A}$ and $\mathbf{x}_\star = \arg\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$. Prove that the gradient descent update for optimizing $f(\mathbf{x})$ is equivalent to:

$$\mathbf{x}^{(i+1)} = \mathbf{B}(\mathbf{x}^{(i)} - \mathbf{x}_\star) + \mathbf{x}_\star.$$

   Hint: Write down the gradient update equation and an expansion of the above equation. Then identify what you need to prove for these two equations to be equal and use the fact that $\mathbf{x}_* = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{b}$.

3. Prove that for $\eta = \frac{1}{2\lambda_1(\mathbf{A}^T\mathbf{A})}$, the eigenvalues of $\mathbf{B}$ all fall in the range $\left[0, 1 - \frac{\lambda_d(\mathbf{A}^T\mathbf{A})}{\lambda_1(\mathbf{A}^T\mathbf{A})}\right]$.

4. Use the above to show for after t iterations, the t^{th} iterate of gradient descent satisfies $\|\mathbf{x}^{(t)} - \mathbf{x}_\star\|_2 \leq \left(1 - \frac{\lambda_d(\mathbf{X}^T\mathbf{X})}{\lambda_1(\mathbf{X}^T\mathbf{X})}\right)^t \cdot \|\mathbf{x}^{(0)} - \mathbf{x}_\star\|_2$, assuming that we use the η found in part (3). **Hint:** This question and the one below it will use some ideas from the power method proof given in class.

5. Use the above to show for any $\epsilon \geq 0$, after $t = O\left(\frac{\lambda_1(\mathbf{A}^T\mathbf{A})}{\lambda_d(\mathbf{A}^T\mathbf{A})} \cdot \log(1/\epsilon)\right)$ iterations, the t^{th} iterate of gradient descent satisfies $\|\mathbf{x}^{(t)} - \mathbf{x}_\star\|_2 \leq \epsilon\|\mathbf{x}_\star\|_2$, assuming that we initialize with $\mathbf{x}^{(0)} = \mathbf{0}$ and use the η found in part (3).

6. The ratio $\frac{\lambda_1(\mathbf{A}^T\mathbf{A})}{\lambda_d(\mathbf{A}^T\mathbf{A})}$ is known as the *condition number* of $\mathbf{A}^T\mathbf{A}$. It is important since it governs the convergence rate of gradient descent and many other iterative methods in solving least squares regression and linear systems. Let ϵ be a small constant (e.g., $\epsilon = .01$). Ignoring constants, compare the asymptotic runtime of computing $\mathbf{x}^{(t)}$ in part (6) with that of computing an exact solution via SVD. When is one faster than the other? **Note:** Assume that a full SVD requires $O(nd^2)$ time to compute and that multiplying a vector by $\mathbf{A}$ or $\mathbf{A}^T$ requires $O(nd)$ time.