ECE 6270, Spring 2021

Homework #3

Due Sunday, Feb 14, at 11:59pm Suggested Reading: B&V, Sections 4.2.1-4.2.3 and 9.2-9.3 (You can skip the convergence analysis in 9.3 for now.)

1. Prepare a one paragraph summary of what we talked about in the last week of class. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other classes?). The more insight you give, the better.

2. Provide feedback to your peers on Homework #2 in Canvas.

3. **Monotone gradients.** In class we showed that or a differentiable function f being convex is equivalent to the statement that

$$f(\boldsymbol{x}) \geq f(\boldsymbol{y}) + \langle \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{y}) \rangle, \tag{1}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$. Here we will provide another equivalent characterization of convexity for differentiable f. Specifically, the first-order condition in (1) is equivalent to the statement that

$$\langle \boldsymbol{y} - \boldsymbol{x}, \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x}) \rangle \geq 0 \tag{2}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$. This is often called the *monotone gradient* condition.

   (a) Prove that (1) implies (2).

   (b) (Optional) Prove that (2) implies (1), and thus the conditions are actually equivalent. [Hint: Consider the function $\phi(\alpha) = f(\boldsymbol{x} + \alpha(\boldsymbol{y} - \boldsymbol{x}))$ and recall that the Fundamental Theorem of Calculus tells us that $\phi(1) - \phi(0) = \int_0^1 \phi'(\alpha) d\alpha$.]

   (c) Consider a one-dimensional differentiable convex function $f(x)$. Assume that $f(x)$ has a unique global minimum $x^\star$. What does the above condition say about $f'(x)$ for $x > x^\star$? What about $f'(x)$ for $x < x^\star$?

4. **Strong convexity.** In analyzing gradient descent and other algorithms we will sometimes consider the assumption that f is *strongly convex*, i.e., there exists an $m > 0$ such that

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{y} - \boldsymbol{x}, \nabla f(\boldsymbol{x}) \rangle + \frac{m}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 \tag{3}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$. Note that above we assume f is differentiable in our definition of strong convexity, and that any f satisfying (3) must be convex. When f is strongly convex, the following statements are equivalent:

   (i) f is strongly convex with constant m, i.e., it satisfies (3).

   (ii) $g(\boldsymbol{x}) = f(\boldsymbol{x}) - \frac{m}{2}\|\boldsymbol{x}\|_2^2$, where $\operatorname{dom} g = \operatorname{dom} f$, is convex.

   (iii) $\langle \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}) \rangle \geq m\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$.

   In this problem you will show that these are equivalent.

(a) Prove that (i) is equivalent to (ii).

(b) Prove that (ii) is equivalent to (iii).

5. **Lipschitz gradients.** In our convergence analysis of gradient descent we will also consider the assumption that ∇f is *Lipschitz*, i.e., there exists an $M > 0$ such that

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq M\|\boldsymbol{x} - \boldsymbol{y}\|_2 \tag{4}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$. When f is convex and differentiable, the following statements are equivalent:

(i) ∇f is Lipschitz with constant M, i.e., it satisfies (4).
(ii) $\langle \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\rangle \leq M\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$.
(iii) $g(\boldsymbol{x}) = \frac{M}{2}\|\boldsymbol{x}\|_2^2 - f(\boldsymbol{x})$ is convex.
(iv) $f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \boldsymbol{y} - \boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle + \frac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_2^2$.
(v) $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{y} - \boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle + \frac{1}{2M}\| \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|_2^2$.
(vi) $\langle \boldsymbol{x} - \boldsymbol{y}, \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\rangle \geq \frac{1}{M}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2^2$.

In the statements above, $\boldsymbol{x}, \boldsymbol{y}$ can be arbitrary vectors in $\operatorname{dom} f$. In this problem you will show that these are all equivalent.

(a) Prove that (i) implies (ii).

(b) Prove that (ii) implies (iii). [Hint: From problem 3 we know that showing that $g(\boldsymbol{x})$ is convex is equivalent to establishing the monotonicity of the gradient.]

(c) Prove that (iii) implies (iv).

(d) Prove that (iv) implies (v). This is the hardest, so I will give you a short outline of how to approach this. First, use (iv) and what you know about convexity to show that

$$f(\boldsymbol{x}) - f(\boldsymbol{y}) \leq \langle \boldsymbol{x} - \boldsymbol{z}, \nabla f(\boldsymbol{x})\rangle + \langle \boldsymbol{z} - \boldsymbol{y}, \nabla f(\boldsymbol{y})\rangle + \frac{M}{2}\|\boldsymbol{z} - \boldsymbol{y}\|_2^2$$

holds for any $\boldsymbol{z} \in \operatorname{dom} f$. Since this holds for any $\boldsymbol{z}$, you can find the $\boldsymbol{z}$ that minimizes this upper bound – find this $\boldsymbol{z}$ by taking the gradient with respect to $\boldsymbol{z}$ and setting this equal to zero, then plug this back into this expression and simplify.

(e) Prove that (v) implies (vi).

(f) Prove that (vi) implies (i).

6. **Convergence Rates.** A central focus when considering different optimization algorithms is the *rate of convergence*. It is often not enough to merely argue that the algorithm converges – we want to know how quickly it will do so, and the rate of convergence allows us to quantify this. In the problems below, we assume that $\{x_k\}$ is a sequence that converges to $x^\star$. We say that $\{x_k\}$ converges *linearly* with a rate of convergence of β if

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^\star|}{|x_k - x^\star|} = \beta$$

for some $\beta \in (0, 1)$. If $\beta = 1$ we say that $\{x_k\}$ converges *sublinearly*, and if $\beta = 0$ we say that $\{x_k\}$ converges *superlinearly*.

(a) Suppose that $\{x_k\}_{k=0}^{\infty}$ satisfies $|x_{k+1} - x^\star| \leq \beta |x_k - x^\star|$ for some $0 < \beta < 1$ (and hence converges linearly with rate β). Prove that $|x_k - x^\star| \leq \epsilon$ for all

$$k \geq \frac{\log\left(\frac{|x_0 - x^\star|}{\epsilon}\right)}{\log \beta}.$$

(b) Now suppose that $\{x_k\}_{k=0}^{\infty}$ satisfies $|x_k - x^\star| = \frac{k}{k+1}|x_{k-1} - x^\star|$. Is this convergence linear, sublinear, or superlinear? How large must k be to ensure that $|x_k - x^\star| \leq \epsilon$

(c) A finer-grained distinction among different kinds of linear/superlinear convergence is the *order of convergence*. We say that $\{x_k\}$ converges with order q if

$$\lim_{k \to \infty} \frac{|x_{k+1} - x^\star|}{|x_k - x^\star|^q} = \gamma$$

for some $\gamma > 0$ (not necessarily less than 1). For $q = 1$ this reverts to linear convergence, but $q = 2$ is called *quadratic* convergence, $q = 3$ *cubic* convergence, and so on. Note that q need be an integer. It might not be initially obvious, but quadratic convergence is *much* faster than linear convergence. Consider the two sequences defined by

$$x_k = \frac{1}{2^k} \qquad z_k = \frac{1}{2^{2^k}}.$$

Both converge to zero. Show that $\{x_k\}$ converges linearly and compute the rate. Show that $\{z_k\}$ converges quadratically. Submit a plot (on a log scale) that illustrates the difference in how quickly these converge to zero.

7. **Bisection method.** The bisection method is a strategy for one-dimensional problems of the form

$$\underset{x}{\text{minimize}} \;\; f(x) \quad \text{subject to} \quad x_l \leq x \leq x_u, \tag{5}$$

where $x_l, x_u \in \mathbb{R}$ satisfy $x_l < x_u$ and $f : \mathbb{R} \to \mathbb{R}$ is convex. If f is also differentiable, the bisection algorithm can be expressed as follows:

Set initial bounds: $a = x_l, b = x_u$
Initialize $k = 0$
while not converged **do**
  $x_k = (a + b)/2$
  if $f'(x) > 0$ **then**
    $b = x$
  else
    $a = x$
  end if
  $k = k + 1$
end while

(a) Provide an intuitive explanation for why this algorithm makes sense in light of the monotone gradient property of convex functions.

(b) Assume that f is strictly convex, and hence (5) has a unique solution, denoted $x^\star$. Let x_k denote the estimate provided by the bisection method after k iterations. Argue that x_k converges to $x^\star$.

(c) Determine whether x_k converges linearly, sublinearly, or superlinearly. If linear, compute the rate of convergence β.

(d) Propose a variant of the bisection method for solving the unconstrained problem $\text{minimize}_{x\in\mathbb{R}} \; f(x)$.

8. **Logistic regression.** Logistic regression is a simple, but surprisingly powerful, tool for solving a fundamental problem in machine learning: learning a *classifier* that can distinguish between two classes from a set of training data. Specifically, suppose that we have a set of *training data* $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, where each $\boldsymbol{x}_n \in \mathbb{R}^D$ is a "feature vector" and each $y_n \in \{0, 1\}$ is a "label" that indicates which of two classes $\boldsymbol{x}_n$ corresponds to. The goal in learning a classifier is to find a function h such that $h(\boldsymbol{x})$ predicts the correct label y for $\boldsymbol{x}$ that we have never seen before. We can do this by trying to learn a function h that gives us $h(\boldsymbol{x}_n) = y_n$ on (most of) the training set. In this problem you will explore this idea and implement it on a simple example.

Consider the *logistic* function

$$g(t) = \frac{1}{1 + e^{-t}}.$$

Note that $g(t)$ is always a number between 0 and 1. In logistic regression, we set $t = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b$, where $\boldsymbol{a} \in \mathbb{R}^D$ and $b \in R$, so that for any $\boldsymbol{x}$ we can compute $g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b)$, returning a number between 0 and 1. Our goal is to learn the parameters $\boldsymbol{a}$ and b so that we can interpret $g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b)$ as an estimate of the probability that $\boldsymbol{x}$ belongs to class 1 (and hence, $1 - g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b)$ is the probability that $\boldsymbol{x}$ belongs to class 0). To form a classifier, we then simply compare $g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b)$ to a threshold in order to estimate the label based on which one has a higher predicted probability, i.e.,

$$h(\boldsymbol{x}) = \begin{cases} 1 & \text{if } g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b) \geq \frac{1}{2}, \\ 0 & \text{if } g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b) < \frac{1}{2}. \end{cases}$$

(a) Show that $h(\boldsymbol{x})$ can equivalently be written as

$$h(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{a}^{T}\boldsymbol{x} + b \geq 0 \\ 0 & \text{if } \boldsymbol{a}^{T}\boldsymbol{x} + b < 0. \end{cases}$$

(b) Draw a picture illustrating the set $\{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{a}^{T}\boldsymbol{x} + b \geq 0\}$ for

$$\boldsymbol{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad b = \frac{1}{2}.$$

(c) The key question in fitting a logistic regression model is deciding on how to set $\boldsymbol{a}$ and b based on the training data. To see how this is done, note that if $y_n = 1$, we would like to have $g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b) \approx 1$, in which case

$$\log(g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b)) \approx 0.$$

Similarly, if $y_n = 0$, then we would like $g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b) \approx 0$, or said differently, $1 - g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b) \approx 1$, in which case

$$\log(1 - g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b)) \approx 0.$$

Note that in both cases, the terms on the left-hand side are always negative, so that to make them ≈ 0 corresponds to making these terms as *large* as possible. Combining these desired criteria together, we can express the problem of fitting $\boldsymbol{a}$ and b as

$$\underset{\boldsymbol{a},b}{\text{maximize}} \sum_{n:y_n=1} \log(g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b)) + \sum_{n:y_n=0} \log(1 - g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b)).$$

It is somewhat more convenient to re-express this as the equivalent problem:

$$\underset{\boldsymbol{a},b}{\text{maximize}} \sum_{n=1}^{N} y_n \log(g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b)) + (1 - y_n) \log(1 - g(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b)).$$

Show that, by plugging in the formula for g, this problem reduces to

$$\underset{\boldsymbol{a},b}{\text{maximize}} \sum_{n=1}^{N} y_n(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b) - \log(1 + e^{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}_n + b}).$$

(d) We would now like to use an iterative algorithm like gradient descent to solve this optimization problem. To do this, we will need to calculate the gradient. Note that if we use the notation

$$\widetilde{\boldsymbol{x}}_n = \begin{bmatrix} \boldsymbol{x}_n \\ 1 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a} \\ b \end{bmatrix},$$

then we can re-write our objective function as

$$f(\boldsymbol{\theta}) = \sum_{n=1}^{N} y_n \boldsymbol{\theta}^{\mathrm{T}} \widetilde{\boldsymbol{x}}_n - \log(1 + e^{\boldsymbol{\theta}^{\mathrm{T}}\widetilde{\boldsymbol{x}}_n}).$$

With this notation, compute a formula for the gradient $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ and the Hessian matrix $\boldsymbol{D}_f^2(\boldsymbol{\theta})$. [Hint: You may use the answers from homework 1 to avoid any actual computation here, but be careful as the notation in this problem is a little different...]

(e) Do you think $f(\boldsymbol{\theta})$ is M-smooth for some finite value of M? Do you think $f(\boldsymbol{\theta})$ is strongly convex? (You do not have to calculate M or m here.)

(f) Implement a solver for estimating $\boldsymbol{\theta}$ using gradient descent. Test your implementation on the dataset produced by the following code:

```
import numpy as np
from sklearn import datasets

np.random.seed(2020) # Set random seed so results are repeatable
x,y = datasets.make_blobs(n_samples=100,n_features=2,centers=2,cluster_std=6.0)
```

Use an initial guess of $\boldsymbol{\theta}^{(0)} = \mathbf{0}$. To begin, consider a fixed step size of $\alpha \approx 0.001$ (although you should feel free to play around with α a bit). Report how many iterations your algorithm takes to converge. Remember that you are trying to *minimize* $-f(\boldsymbol{\theta})$. If your algorithm is not converging, a sign error is a likely culprit.

(g) Now implement the bisection algorithm from the previous problem, and create a modified version of gradient descent in which you update α at each iteration using a line search conducted by running the bisection algorithm to find the optimal α (up to some tolerance that you should select). Report how many iterations are now required when using this strategy to find the "optimal" step size at each iteration. Also report the total number of iterations taken by the combined bisection searches. Does the bisection strategy seem worth the additional costs?