

HOMWORK 2

SMOOTHNESS, CONVEXITY, & GRADIENT DESCENT *

10-425/10-625 INTRODUCTION TO CONVEX OPTIMIZATION
<http://425.mlcourse.org>

OUT: 09/21/2023
DUE: 10/01/2023
TAs: Asad & Tiancheng

Instructions

- **Collaboration Policy:** Please read the collaboration policy in the syllabus.
- **Late Submission Policy:** See the late submission policy in the syllabus.
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code.
 - **Written:** You will submit your completed homework as a PDF to Gradescope. For each problem, please clearly indicate the question number (e.g. 3.2). Submissions can be handwritten, but must be clearly legible; otherwise, you will not be awarded marks. Alternatively, submissions can be written in \LaTeX . You may use the \LaTeX source of this assignment (included in the handout .zip) as your starting point. For multiple choice / select all questions, simply write the letter(s) (e.g. A, B, C) corresponding to your chosen answer.
 - **Programming:** You will submit your code for programming questions to Gradescope. There is no autograder. We will examine your code by hand and may award marks for its submission.
- **Materials:** The data and reference output that you will need in order to complete this assignment is posted along with the writeup and template on the course website.

Question	Points
Implications of Smoothness	14
Strict and Strong Convexity	11
Gradient descent convergence analysis	15
Subgradient method for LASSO regression	22
Collaboration Questions	2
Total:	64

*Compiled on Friday 22nd September, 2023 at 16:24

1 Implications of Smoothness (14 points)

Let f be convex and twice continuously differentiable.

1. ∇f is Lipschitz with constant L ;
2. $(\nabla f(x) - \nabla f(y))^T(x - y) \leq L\|x - y\|_2^2$ for all x, y ;
3. $\nabla^2 f(x) \preceq LI$ for all x ;
4. $f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|_2^2$ for all x, y .

Prove the following statements:¹

- 1.1. (3 points) $1 \Rightarrow 2$

(Hint: use the Cauchy-Schwartz inequality.)

- 1.2. (4 points) $3 \Rightarrow 4$

(Hint: Use the Taylor expansion with Lagrange form of the remainder, $f(y) = f(x) + \nabla f(x)^T(y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(\lambda x + (1 - \lambda)y)(y - x)$ which holds for any twice differentiable function f .)

- 1.3. (3 points) $4 \Rightarrow 2$

(Hint: Was that $f(y) \leq \dots$ or $f(x) \leq \dots$ or is it *both*?)

- 1.4. (4 points) $3 \Rightarrow 1$

(Hint: Start by applying the integral form of the mean-value theorem, so we have for all x, y : $\nabla f(y) - \nabla f(x) = \int_0^1 \nabla^2 f(x + t(y - x))(y - x) dt$.)

¹There's a lovely proof of $2 \Rightarrow 3$, but it seemed best to leave that one as an optional exercise.

2 Strict and Strong Convexity (11 points)

2.1. (3 points) **Proof:** Let f be convex and twice continuously differentiable with convex domain $\text{dom}(f)$.

1. $g(x) = f(x) - \frac{\alpha}{2}\|x\|^2$ is convex.

2. for all $x, y \in \text{dom}(f)$ and $\lambda \in [0, 1]$

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) - \frac{\alpha}{2}\lambda(1 - \lambda)\|x - y\|^2$$

Show that 1 \Rightarrow 2.

2.2. (3 points) **Proof:** Show that strong convexity implies strict convexity.

2.3. (1 point) **True or False:** If f is a strictly convex function, then its minimizer must be unique.

A. True

B. False

2.4. (1 point) **True or False:** For a twice differentiable function f , the function f is strictly convex if and only if $\nabla^2 f(x) \succ 0$ for all x .

A. True

B. False

2.5. (3 points) **Proof:** Show that strong convexity does not imply differentiability. (Hint: Consider the function $f(x) = x^2 + |x|$. Is it strongly convex? Is it differentiable?).

3 Gradient descent convergence analysis (15 points)

Here we will assume nothing about convexity of f . We will show that gradient descent reaches an ϵ -substationary point x , such that $\|\nabla f(x)\|_2 \leq \epsilon$, in $O(1/\epsilon^2)$ iterations. Important note: you may assume that f is β -smooth and satisfies the following inequality

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{\beta}{2}\|y - x\|_2^2, \quad \text{for all } x, y. \quad (1)$$

3.1. (3 points) Plug in $y = x^+ = x - \eta \nabla f(x)$ to (1) to show that

$$f(x^+) \leq f(x) - \left(1 - \frac{\beta\eta}{2}\right)\eta\|\nabla f(x)\|_2^2.$$

3.2. (3 points) Use $\eta \leq 1/\beta$, and rearrange the previous result, to get

$$\|\nabla f(x)\|_2^2 \leq \frac{2}{\eta}(f(x) - f(x^+)).$$

3.3. (3 points) Sum the previous result over all iterations from $1, \dots, k+1$ to establish

$$\sum_{i=0}^k \|\nabla f(x^{(i)})\|_2^2 \leq \frac{2}{\eta}(f(x^{(0)}) - f^*).$$

3.4. (3 points) Lower bound the sum in the previous result to get

$$\min_{i=0, \dots, k} \|\nabla f(x^{(i)})\|_2 \leq \sqrt{\frac{2}{\eta(k+1)}(f(x^{(0)}) - f^*)},$$

3.5. (3 points) Show that the bound above implies that gradient descent can find an ϵ -substationary point in $O(1/\epsilon^2)$ iterations.

4 Subgradient method for LASSO regression (22 points)

Given labels $Y \in \mathbb{R}^n$, and a feature matrix $X \in \mathbb{R}^{n \times p}$ with rows $x_1, \dots, x_n \in \mathbb{R}^p$, LASSO is linear regression with ℓ_1 regularization. The optimization problem is:

$$\min_{\beta} \quad \frac{1}{n} \|X\beta - Y\|_2^2 + \lambda \|\beta\|_1$$

where $\lambda \in \mathbb{R}$ is the weight of the regularizer and $\beta \in \mathbb{R}^p$ are the model parameters we wish to learn. Below we use $f(\beta; X, Y) = \frac{1}{n} \|X\beta - Y\|_2^2 + \lambda \|\beta\|_1$ to denote this objective function and $h(\beta; X, Y) = \frac{1}{n} \|X\beta - Y\|_2^2$ to denote the unregularized objective.

There are many methods for solving the LASSO problem. In this section, you will implement two such algorithms and compare their behavior. Both algorithms follow the same high-level structure:

Algorithm 1 Training Algorithm

- 1: Initialize $\beta^{(0)}$ to the 0 vector
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: Update the parameters $\beta^{(t)} = \dots$
 - 4: Compute train loss $e_{\text{train}}^{(t)} = f(\beta; X_{\text{train}}, Y_{\text{train}})$
 - 5: Compute test loss $e_{\text{test}}^{(t)} = f(\beta; X_{\text{train}}, Y_{\text{train}})$
 - 6: **return** β
-

The difference between the algorithms is in how they update the parameters in line 3:

- **Algorithm 1A: The Subgradient Method** Update the parameters as in gradient descent, except that we select a vector $g^{(t)}$ from the subgradient ∂f .

$$\begin{aligned} g^{(t)} &\in \partial f(\beta^{(t-1)}; X_{\text{train}}, Y_{\text{train}}) \\ \beta^{(t)} &= \beta^{(t-1)} - \eta g^{(t)} \end{aligned}$$

- **Algorithm 1B: Truncated Gradient** Compute an intermediate vector of parameters $\beta^{(t-\frac{1}{2})}$ without the ℓ_1 penalty. And whenever an entry of the vector $\beta^{(t-\frac{1}{2})}$ has absolute value less than $\eta\lambda$, set it to zero. The full update rule is:

$$\begin{aligned} g^{(t)} &= \nabla h(\beta^{(t-1)}; X_{\text{train}}, Y_{\text{train}}) \\ \beta^{(t-\frac{1}{2})} &= \beta^{(t-1)} - \eta g^{(t)} \\ \beta_j^{(t)} &= \text{sign}(\beta_j^{(t-\frac{1}{2})}) \max(0, |\beta_j^{(t-\frac{1}{2})}| - \eta\lambda), \quad \forall j \end{aligned}$$

The data is provided as numpy arrays saved to two files: `train.npy` for training and `test.npy` for testing. The *last* column contains the labels Y and the other columns contain the features X . You should use the `np.load()` function to read each `.npy` file. The first column of the features is a vector of all ones; this ensures that we don't need an explicit intercept parameter since β_1 is doing that job.

Initialize β to be 0, use $\lambda = 0.1$, and learning rate $\eta = 0.01$ to run the optimization for $T = 200$ iterations.

Remember to submit your full code to the corresponding programming slot on Gradescope; here is no autograder.

- 4.1. (5 points) **Plot:** Create a plot for the loss on training data and the loss on test data throughout the training process. Here the loss is the objective function with either $X = X_{\text{train}}$ or $X = X_{\text{test}}$. The x-axis should show the iteration and the y-axis the loss. Provide two plots: one for The Subgradient Method and one for Truncated Gradient.

(Hint: Use the provided function `plot_loss_curve()` to plot the train/test losses.)

- 4.2. (5 points) **Plot:** Create a histogram of the learned weights β . Provide two plots: one for The Subgradient Method and one for Truncated Gradient.

(Hint: Use the provided function `plot_weight_histogram()` to plot each histogram.)

- 4.3. (3 points) **Table of numerical values:** Report the number of parameters (i.e. entries $\beta_j \in \mathbb{R}$ in the vector β) that are exactly zero ($\beta_j = 0$) and that are non-zero ($\beta_j \neq 0$) for the two algorithms (The Subgradient Method vs. Truncated Gradient).

(Hint: You can use the function `np.count_nonzero()`.)

	# of Exact Zeros	# of Non-Zeros
Subgradient Method		
Truncated Gradient		

- 4.4. (3 points) **Short answer:** How does the choice of algorithm (The Subgradient Method vs. Truncated Gradient) affect the sparsity² of the learned weights β ?
- 4.5. (4 points) **Table of numerical values:** Report the final train loss and final test loss for the two algorithms (The Subgradient Method vs. Truncated Gradient).

	Subgradient Method	Truncated Gradient
Final train loss		
Final test loss		

- 4.6. (2 points) **Short answer:** What is the empirical tradeoff between sparsity and training loss? And between sparsity and test loss?

²Sparsity refers to the number of zeros present in the parameter vector.

5 Collaboration Questions (2 points)

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found in the syllabus.

- 5.1. (1 point) Did you collaborate with anyone on this assignment? If so, list their name or Andrew ID and which problems you worked together on.
- 5.2. (1 point) Did you find or come across code that implements any part of this assignment? If so, include full details.