CS 446 / ECE 449 — Homework 1

your NetID here

September 2, 2021

Instructions.

- Homework is due **Wednesday, September 15, at noon CST**; you have **3** late days in total for **all Homeworks**.
- Everyone must submit individually at gradescope under `hw1` and `hw1code`.
- The "written" submission at `hw1` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw1`, gradescope will ask you to m**ark out boxes around each of your answers**; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full **academic integrity** information. You should cite any external reference you use.
- We reserve the right to reduce the auto-graded score for `hw1code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- When submitting to `hw1code`, only upload `hw1.py` and `hw1_utils.py`. Additional files will be ignored.

1. Linear Regression.

(a) Consider a linear regression problem with a dataset containing N data points $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$. The accumulated loss function is given by:

$$L_{OLS}(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2$$

where $\boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{X} \in \mathbb{R}^{N\times(d+1)}$ and $\boldsymbol{w} \in \mathbb{R}^{d+1}$.

i. Find the Hessian matrix of $L_{OLS}(\boldsymbol{w})$.
Hint: You may want to use the fact that $||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^\top(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$

ii. Recall that a twice-continuously differential function $f(\boldsymbol{x})$ is strictly convex i.f.f. its Hessian is positive definite for all $\boldsymbol{x}$. Prove that if N is less than the input dimension d, $L_{OLS}(\boldsymbol{w})$ can not be strictly convex.

iii. No matter what $\boldsymbol{X}$ is, prove that for $\forall \boldsymbol{w}_1, \boldsymbol{w_2} \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^{d+1}} L_{OLS}(\boldsymbol{w})$, we have $\boldsymbol{X}\boldsymbol{w_1} = \boldsymbol{X}\boldsymbol{w}_2$. Note that $\boldsymbol{w}_1, \boldsymbol{w}_2$ can be different.
Hint: Use the convexity of the loss function and the convex combinations of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$.

(b) Consider the same dataset with an L2-norm regularization added to the OLS loss function. Linear regression with L2 regularization is also called Ridge regression. Recall the composite loss function of ridge regression:

$$L_{ridge}(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \frac{\lambda}{2}||\boldsymbol{w}||_2^2$$

i. One advantage of ridge regression is that for a positive regularization constant ($\lambda > 0$), the matrix $(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})$ is always invertible. Prove that the matrix $(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})$ is invertible by showing that it's positive definite.

ii. Knowing that $L_{ridge}(\boldsymbol{w})$ is a convex function, show that the estimator for ridge regression is:

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

Solution.

2. Programming - Linear Regression.

Recall that the empirical risk in the linear regression method is defined as $\widehat{\mathcal{R}}(\boldsymbol{w}) := \frac{1}{2N}\sum_{i=1}^{N}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} - y^{(i)})^2$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ is a data point and $y^{(i)}$ is an associated label.

(a) **Implement linear regression using gradient descent in the `linear_gd(X, Y, lrate, num_iter)` function of `hw1.py`.**

The arguments for this function are: `X` as the training features, a tensor with shape $N \times d$; `Y` as the training labels, an $N \times 1$ tensor; `lrate` as learning rate (step size), a float number; and `num_iter` as the number of iterations for gradient descent to run. The objective of this function is to find parameters $\boldsymbol{w}$ that minimize the empirical risk $\widehat{\mathcal{R}}(\boldsymbol{w})$ using gradient descent (only gradient descent).

To keep consistent with the standard program and get correctly scored, **prepend** a column of ones to `X` in order to accommodate the bias term in $\boldsymbol{w}$, thus $\boldsymbol{w}$ should has $d+1$ entries. Then use $\boldsymbol{w} = 0$ as the initial parameters, and return

Hint. If your are new to machine learning or programming with pytorch, we offer some kind suggestions. First, try using the vector/matrix operations provided in pytorch and avoid using for-loops. This will improve both the efficiency and style of your program. Second, create your own test cases for debugging before submission. With very few samples in your own test case, it is convenient to compare the program output with your manual calculation. Third, to avoid matrix computation error, remember to check the shapes of tensors regularly.

Library routines: `torch.matmul (@), torch.tensor.shape, torch.tensor.t, torch.cat, torch.ones, torch.zeros, torch.reshape`.

(b) **Implement linear regression by using the pseudo inverse to solve for w in the `linear_normal(X,Y)` function of `hw1.py`.**

The arguments for this function are: `X` as the training features, a tensor with shape $N \times d$ tensor; `Y` as the training labels, an $N \times 1$ tensor. To keep consistent with the standard program and get correctly scored, **prepend** a column of ones to `X` in order to accommodate the bias term in $\boldsymbol{w}$, thus $\boldsymbol{w}$ should has $d+1$ entries.

Library routines: `torch.matmul (@), torch.cat, torch.ones, torch.pinverse`.

(c) **Implement the `plot_linear()` function in `hw1.py`.** Follow the steps below.

Use the provided function `hw1_utils.load_reg_data()` to generate a training set `X` and training labels `Y`. Then use `linear_normal()` to calculate the regression results $\boldsymbol{w}$. Eventually plot the points of dataset and regressed curve. Return the plot as output. Note that `plot_linear()` should return the figure object and you should **include the visualization in your written submission**.

Hint. If your are new to plotting machine learning visualizations, we offer some kind suggestions. `matplotlib.pyplot` is an "extremely" useful tool in machine learning, and we commonly refer to it as `plt`. Please first get to know the most basic usages by examples from its official website (such as scatter plots, line plots, etc.). As for our programming question specifically, you may divide and conquer it by first plotting the points in the dataset, then plotting the linear regression curve.

Library routines: `torch.matmul (@), torch.cat, torch.ones, plt.plot, plt.scatter, plt.show, plt.gcf` where `plt` refers to the `matplotlib.pyplot` library.

Solution.

3. Programming - Logistic Regression.

Recall the empirical risk $\widehat{\mathcal{R}}$ for logistic regression (as presented in lecture 3):

$$\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-y^{(i)}\boldsymbol{w}^\top \boldsymbol{x}^{(i)})).$$

Here you will minimize this risk using gradient descent.

(a) In your **written submission**, derive the gradient descent update rule for this empirical risk by taking the gradient. Write your answer in terms of the learning rate (step size) η, previous parameters $\boldsymbol{w}$, new parameters $\boldsymbol{w}'$, number of examples N, and training examples $\boldsymbol{x}^{(i)}$. Show all of your steps.

(b) Implement the `logistic()` function in `hw1.py`. You are given as input a training set `X`, training labels `Y`, a learning rate (step size) `lrate`, and number of gradient updates `num_iter`. Implement gradient descent to find parameters $\boldsymbol{w}$ that minimize the empirical risk $\widehat{\mathcal{R}}_{\log}(\boldsymbol{w})$. Perform gradient descent for `num_iter` updates with a learning rate (step size) of `lrate`. Same as previous questions, initialize $\boldsymbol{w} = 0$, return $\boldsymbol{w}$ as output, and prepend `X` with a column of ones.

Library routines: `torch.matmul (@), torch.tensor.t, torch.exp.`

(c) Implement the `logistic_vs_ols()` function in `hw1.py`. Use `hw1_utils.load_logistic_data()` to generate a training set `X` and training labels `Y`. Run `logistic(X,Y)` from part (b) taking `X` and `Y` as input to obtain parameters $\boldsymbol{w}$ (use the defaults for `num_iter` and `lrate`). Also run `linear_gd(X,Y)` from Problem 2 to obtain parameters $\boldsymbol{w}$. Plot the decision boundaries for your logistic regression and least squares models along with the data `X`. [Note: As we learned in the class that the decision rule of Least Squares and Logistic Regression for predicting the class label is $\text{sign}(\hat{\boldsymbol{w}}^\top \boldsymbol{x})$, the decision boundary can be obtained from $\hat{\boldsymbol{w}}^\top \boldsymbol{x} = 0$, i.e., for $d = 2$, we have $x_2 = -(\hat{w}_0 + \hat{w}_1 \times x_1)/\hat{w}_2$.] Include the visualizations in your **written submission**. Which model appears to classify the data better? Explain in the **written submission** that why you believe it is better for this problem.

Library routines: `torch.linspace, plt.scatter, plt.plot, plt.show, plt.gcf.`

Solution.

4. Convexity, Lipschitz Continuity, and Smoothness

(a) Convexity

i. Show that if a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, then for any matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^n$, the function $g : \mathbb{R}^m \to \mathbb{R}$ defined by $g(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$ is convex, where $\boldsymbol{x} \in \mathbb{R}^m$.

ii. Prove that if the differentiable function f is λ-strongly convex and the differentiable function g is convex then $f + g$ is λ-strongly convex.

iii. Given m convex functions $\{f_i : \mathbb{R}^n \to \mathbb{R}\}_{i=1}^m$, denote

$$f(\boldsymbol{x}) = \max_{i\in[m]} \ f_i(\boldsymbol{x}),$$

where $[m] = \{1, 2, \ldots, m\}$. Prove that f is convex.

(b) Lipschitzness and Smoothness

We say a function $f : \mathbb{R}^n \to \mathbb{R}^d$ is ρ-Lipschitz if $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$, it holds that $\|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)\|_2 \leq \rho\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$.

i. Prove that if $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^d$ are ρ-Lipschitz functions, then the composite $g \circ f : \mathbb{R}^n \to \mathbb{R}^d$ defined by $(g \circ f)(\boldsymbol{x}) = g(f(\boldsymbol{x}))$ is ρ^2-Lipschitz.

ii. Given a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ whose gradient is β-Lipschitz, prove that for $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ we have

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) \leq \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_2^2.$$

Hint: You are not required to follow the hints, but please consider them if you have no idea for proof. (1) Define a tool function $\phi(t) = f((1-t)\boldsymbol{x} + t\boldsymbol{y})$, thus $f(\boldsymbol{y}) - f(\boldsymbol{x}) = \phi(1) - \phi(0) = \int_0^1 (\texttt{figure it out})$; (2) If you get stuck at the final steps, taking a look at the Cauchy–Schwarz inequality may be helpful.

Solution.