Introduction to Robot Intelligence (CSCI-UA 480-072) Homework 2

Instructor: Lerrel Pinto

February 9, 2023

Submission Instructions

You must submit solutions to both the theory and coding portion of this homework to be eligible for full credit on this assignment.

Please navigate to the "Assignments" page of the course website (**linked here**) in order to download or copy the coding portion of the assignment.

You are strongly encouraged to typeset your answers to the theory questions below using LaTeX, via the course homework template (**linked here**). You must submit your answers to the coding problems by filling out the provided iPython notebook. We encourage you to use Google Colab to write and test your code.

This problem set is due on **February 21, 2023, 11:59 PM**. When you have completed both portions of the homework, submit them **on the course Gradescope as two separate files, with the coding portion in `.ipynb` format** by the due date. **No other forms of submissions will be accepted. Late submissions will also not be accepted.**

You may not discuss the questions in this problem set with other students.

Theory Questions

Question 1: Optimization Roulette (16 points per part)

For each of the stated problems below, identify which of the following is the most appropriate method to employ: (1) **exact solve**; (2) **linear/logistic regression**; (3) **neural network trained with gradient descent**; or (4) **cross-entropy method**. Describe how precisely you would employ this method to solve the problem, and explain your reasoning for why your chosen method is the most appropriate. In some settings, two methods may be equally appropriate; if you think this is the case, identify the pair, explain why both methods are equally effective, and elucidate how both can be used to solve the problem.

You will be graded based on the quality and correctness of your answers and accompanying explanations.

Part (a): Boolean Satisfiability. Let $x_1, \dots, x_n$ be a set of *boolean* variables, i.e. $x_i \in \{\texttt{TRUE}, \texttt{FALSE}\}, \forall i : 1 \leq i \leq n$.

A boolean *or*-clause is defined by joining several boolean variables together with the "OR" operation, denoted by the symbol $\vee$. For example, $(x_1 \vee x_2 \vee x_3)$ is a length three *or*-clause, while $(x_1 \vee x_n)$ is a length two *or*-clause. The first of these clauses evaluates to `TRUE` if and only if at least one of x_1, x_2, x_3 is `TRUE`, while the second evaluates to `TRUE` if and only if at least one of x_1, x_n is `TRUE`.

Any boolean *formula* ϕ over variables $x_1, \dots, x_n$ can be written as set of boolean *or*-clauses joined together with the "AND" operation, denoted by the symbol $\wedge$; for example, $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_n)$. We say that a boolean formula is *satisfiable* if there exists an assignment to its variables such that all *or*-clauses evaluate to `TRUE`.

Given a fixed boolean formula $\phi(x_1, \dots, x_n)$, your goal is to use an optimization method to **efficiently** and **exactly** determine whether it is satisfiable.

Part (b): Data-Limited Binary Classification. You are a newly hired engineer working for the self-driving startup Druise. During your hiring interview, you told your manager that you took "Robot Intelligence" in college, and so they ask you to put your skills to the test by developing a "STOP" sign classifier, i.e. a function which given an image $x \in \mathcal{X}$, predicts whether or not the image contains a "STOP" sign. Unfortunately, the startup has limited resources to collect data, so you are given a dataset with only five hundred images $\{x_1, \dots, x_{500}\}$, each image x_i accompanied by the label $y_i : y_i \in \{$"`STOP SIGN`", "`NOT A STOP SIGN`"$\}$.

Part (c): Big Data Multi-Class Classification. You were successful at producing a decent "STOP" sign classifier at Druise, and have since left the company to join the force of its bitter rival, Vaymo. Your new manager asks you to produce a general-purpose road sign classifier using a massive dataset containing one hundred-million road sign images, $\{x_1, \dots x_{1e8}\}$, each image x_i accompanied by a label $y_i : y_i \in \{$"`STOP SIGN`", "`YIELD SIGN`", "`SLIPPERY ROAD SIGN`"$, \dots\}$.

Part (d): Watching a Rocket Launch. The famed rocket company GreenOrigin is launching their brand new rocket from a launchpad in the suburbs of New York City. You are planning to watch the launch. Ideally, you would like to see the rocket in air for as long as possible, and you are willing to go at most

0.1 degree north or south in latitude, as well as at most 0.1 degree east or west in longitude from the geographical coordinates of your dorm to do so. Lucky for you, GreenOrigin has released their visibility model for the launch; indeed, given a pair (x, y) of latitude and longitude coordinates in New York City, the rocket is estimated to be visible for $t(x, y) = |x - 40.7| \cdot |(x - 40.7) + (y + 73.9)^4 + \sin^2(x/40.7)|$ minutes. You would like to determine the exact coordinates satisfying your constraints at which the rocket is visible for the maximal amount of time.

Part (e): Plant Watering. A friend gifts you a mysterious house plant. You're not sure how often and how much it needs to be watered. You decide to launch a long-term experiment using an optimization mechanism discussed in class to find a near-optimal weekly watering strategy, $\pi : \{1, \ldots, 7\} \mapsto \{0, \frac{1}{2}, 1\}$, mapping each day of the week to one of three quantities of water: no cups, half a cup, or a full cup. At the end of each week, you will take a photo of the plant, record how healthy it looks on the scale of 1 to 10, and possibly adjust your strategy for the following week.

Question 2: GD and SGD (10 points per part)

Respond to each of the stated questions in parts (a) and (b). You may keep your answers concise.

Hint: Try completing problem 2 of the coding portion of this homework before attempting this problem.

Part (a): Learning Rate. What is the role of learning rate in gradient descent? What happens if the learning rate is set too low? What happens if it is set too high?

Part (b): Batch Size. What is the role of batch size in stochastic gradient descent? In what situation might we need to lower the batch size? Increase it?