Homework 1: Optimization Methods

Problem 1: First-order condition for optimality (30 points)

Use the first-order condition to find all stationary points of g (calculations should be done by hand). Then plot g and label the point(s) you find, and determine "by eye" whether each stationary point is a minimum, maximum, or saddle point.

(a) $g(w) = w \log(w) + (1 - w) \log(1 - w)$ where w lies between 0 and 1

(b) $g(w) = \log(1 + e^{w})$

(c) $g(w) = w \tanh(w)$

(d) $g(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{C}\mathbf{w} + \mathbf{b}^T\mathbf{w}$ where $\mathbf{C} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

(a) 8 pts (b) 5 pts (c) 8 pts (d) 9 pts

Problem 2: Try out gradient descent (30 points)

Run gradient descent to minimize the function

$$g(w) = \frac{1}{50}\left(w^4 + w^2 + 10w\right)$$

with an initial point w0 = 2 and 1000 iterations. Make three separate runs using each of the steplength values $\alpha = 1$, $\alpha = 10^{-1}$, and $\alpha = 10^{-2}$. Compute the derivative of this function by hand, and implement it (as well as the function itself) in Python using **NumPy**.

Plot the resulting cost function history plot of each run in a single figure to compare their performance. Which steplength value works best for this particular function and initial point?

Problem 3: Compare fixed and diminishing steplength values (40 points)

Compare fixed and diminishing steplength values for a simple example. Repeat the comparison experiment described below, producing the plots as shown in the below figure.

Experiment
In the below figure, we illustrate the comparison of a fixed steplength scheme and a diminishing one to minimize the function

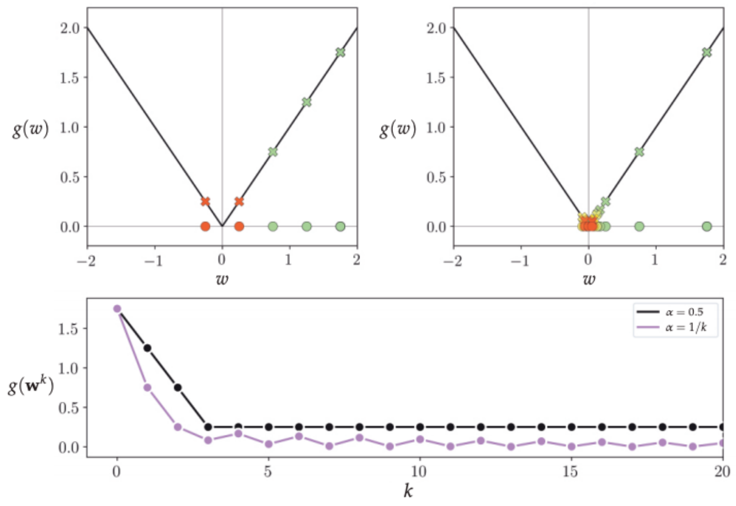
$$g(w) = |w|.$$

Notice that this function has a single global minimum at w = 0. We make two runs of 20 steps of gradient descent, each initialized at the point w0 = 1.75, the first with a fixed steplength rule of α = 0.5 (shown in the top-left panel) for each and every step, and the second using the diminishing steplength rule α = 1/k (shown in the top-right panel).

Here we can see that the fixed steplength run gets stuck, unable to descend towards the minimum of the function, while the diminishing steplength run settles down nicely to the global minimum (which is also reflected in the cost function history plot shown in the bottom panel of the figure). This is because the derivative of this function (defined everywhere but at w = 0) takes the form

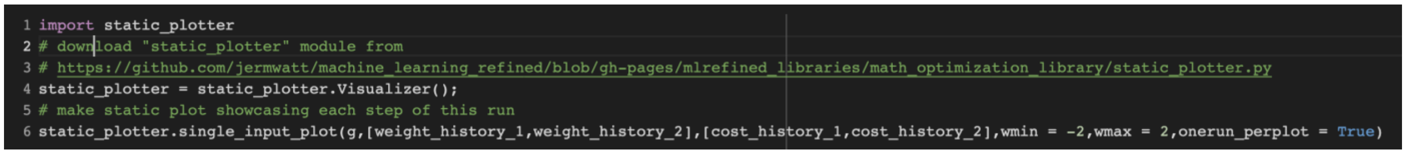
$$\frac{\mathrm{d}}{\mathrm{d}w} g(w) = \begin{cases} +1 & \text{if } w > 0 \\ -1 & \text{if } w < 0 \end{cases}$$

which makes the use of any fixed steplength scheme problematic since the algorithm will always move at a fixed distance at each step. We face this potential issue with all local optimization methods



Hint: You may use the below code for plotting the top two figures (g(w) vs. w).

Download the module “static_plotter.py” using this link.


```
import static_plotter
# download "static_plotter" module from
# https://github.com/jermwatt/machine_learning_refined/blob/gh-pages/mlrefined_libraries/math_optimization_library/static_plotter.py
static_plotter = static_plotter.Visualizer();
# make static plot showcasing each step of this run
static_plotter.single_input_plot(g,[weight_history_1,weight_history_2],[cost_history_1,cost_history_2],wmin = -2,wmax = 2,onerun_perplot = True)
```