EE364b Spring 2022 Homework 1

Due Sunday 4/16 at 11:59pm via Gradescope

1.0 *(0 points) Course forum data opt-out form.* The course staff is hoping to train a language model on the discussions on public conversations in Ed. We hope that this will help future EE364a/b students by enabling fast automated help and new CVXPY debugging tools in future quarters. **No private Ed posts will be included in the dataset under any circumstance. Public conversations will be stripped of personally identifiable information before contributing to the model's training.**

If you would like to opt-out or change your consent status at any point in the quarter please fill out this form: https://forms.gle/2aqa9w2xdPAdz1wp8

1.1 *(8 points) Parallel Subgradient Computation using Dask.* In this question, you will implement subgradient computations using the parallel computing library Dask.

(a) (0 points) Read the example showing how to modify Python code to utilize data parallelism via Dask here. Install Python and Dask as described on this page.

(b) (2 points) Suppose that $x \in \mathbb{R}^n$ is fixed and given. Compute a subgradient of the function

$$f(x) := \max_{j \in \{1,\ldots,m\}} a_j^T x$$

by using the Python code template below (also available in Canvas/Files/Homeworks). Set $n = 100 \times 10^6$ and $m = 4$. Generate $x \in \mathbb{R}^n$ and $a_1, \ldots, a_m \in \mathbb{R}^n$ randomly and independently from a standard normal distribution. Your code should return a valid subgradient of $f(x)$ at x. Repeat the data generation and subgradient calculation for 100 trials and plot a histogram of the total computation time of the subgradient using the serial approach.

(c) (3 points) Implement the same subgradient computation in (b) using Dask to see the benefit of data parallelism in terms of the total computation time. You only have to modify your code in (b) using `dask.delayed` as shown in this tutorial. Repeat the data generation and subgradient calculation for 100 trials and plot a histogram of the total computation time of the subgradient with parallelization.

(d) (2 points) Implement the same subgradient calculation in part (b) using `numpy.matmul()` and `numpy.argmax()`. Repeat the data generation and subgradient calculation for 100 trials and plot a histogram of the total computation time of the subgradient with Numpy.

(e) (1 points) Visualize the computation graph for your Dask based implementation using the function `visualize()` for $n = 5$, $m = 4$.

```
from time import time
import dask
import numpy as np
def inprod(x, y):
    return np.dot(x,y)
n, m = 100000000, 4
data = np.random.randn(m,n)
start = time()
output = []
x= np.random.randn(n)
for i in range(data.shape[0]):
    output.append(inprod(data[i,:],x))
index = np.argmax(output)
print("Time spent for the computation without parallelization:",time()-start)
```

1.2 (3 points) *Does autodiff work?* Calculate a ‘gradient’ of the following functions using an automatic differentation (autodiff) method at the specified points. Check whether the result is a valid subgradient and give an explanation if there is a mismatch. You may use any programming language and any autodiff package.

(a) $f(x) = \max(x, 0)^2$ at $x = 0$

(b) $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 0$

(c) $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 10^{-50}$

(d) $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 10^{-30}$

(e) $f(x) = \min(|x|, x)$ at $x = 0$

(f) $f(x) = \min(x, |x|)$ at $x = 0$

Hint: You can use Pytorch and Google Colab for autodiff (recommended)[1]*. Please see the following example which calculates the gradient of* $ReLU(x) = \max(x, 0)$ *at* $x = 0$.

```
import torch
x = torch.tensor([0.], requires_grad=True)
zero = torch.tensor([0.])
f = torch.max(x,zero)
f.backward()
print(x.grad) #prints the gradient of f with respect to x at its current value
```

[1] You can run your python script remotely using a Google Colab notebook: `colab.research.google.com`

1.3 (2 points) *Subgradients in ChatGPT.* Large-language models like ChatGPT and GPT-4 have been making waves in the machine learning community. In this problem, we investigate the mathematical capabilities of ChatGPT through the calculation of subgradients.

(a) Suppose we are interested in a subgradient of $f(x) = \min(x, 0) + \max(x, 0)$ at $x = 0$. Start with the following query in ChatGPT:

Find a subgradient of the function min(x, 0) + max(x, 0) at x = 0.

Does ChatGPT give the correct answer? If not, where did ChatGPT go wrong in its calculation? Attach a screenshot of both the query and output with your submission.

(b) If ChatGPT gave the wrong answer in part (a), develop your own sequence of queries that guide ChatGPT to the correct answer. What was your strategy to guide ChatGPT to the correct answer?

If ChatGPT gave the correct answer in part (a), try the query from part (a) again in several (at least 2) new chats. Does ChatGPT still give the correct answer?

In either case, attach a screenshot of both the queries and outputs with your submission.

1.4 (6 points) *Subdifferential sets.* For each of the following convex functions, determine the subdifferential set at the specified point.

(a) $f(x) = \mathrm{ELU}_\alpha(x) \triangleq \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$ with $0 \leq \alpha < 1$, at $x = 0$.[2]

(b) $f(x) = \mathrm{ELU}_1(x)$ at $x = 0$.

(c) $f(x) = \frac{1}{2}\|x\|_2^2 + \|x\|_1$ at $x = (-2, 0, 1)$.

(d) $f(x) = \|Ax\|_\infty$ with $A = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ 1 & 2 & 1 \end{pmatrix}$, at $x = (1, -1, 1)$.

(e) $f(x_1, x_2) = \log(e^{|x_1|} + e^{|x_2|})$ at $(x_1, x_2) = (0, 0)$.

(f) $f(x_1, x_2) = \max\{x_1 + x_2 + 1, x_1^2 + x_2^2, (x_1 + 2x_2)^2 - 6\}$ at $(x_1, x_2) = (1, 1)$.

1.5 (6 points) *Weak subgradient calculus.* For each of the following convex functions, explain how to calculate a subgradient at a given x.

(a) $f(x) = -\log(\min\{a^T x + b, 1\})$, where $a \in \mathbb{R}^n, b \in \mathbb{R}, x \in \mathcal{H}_+ = \{w : a^T w + b > 0\}$.

(b) $f(x) = \max_{i=1,\ldots,m} -\log(\min\{a_i^T x + b_i, 1\})$.

[2] The Exponential Linear Unit (ELU) is a generalization of the Rectified Linear Unit (ReLU) activation function used in training deep neural networks (Clevert et al., 2016).

(c) $f(x) = \max_{t \in \Delta^{n-2}} p(x,t) = x_1 + x_2 t_1 + \cdots + x_n t_{n-1}$, where $\Delta^{n-1} = \{z : z \geq 0, 1^\top z = 1\}$ is the probability simplex.

(d) $f(x) = x_{[1]} + \cdots + x_{[k]}$, where $x_{[i]}$ denotes the ith largest element of the vector x.

(e) $f(x) = \min_{y \succeq 0} \|x - y\|_A^2$, *i.e.*, the square of the distance of x in the matrix norm $\|z\|_A^2 = z^\top A z$ to the non-negative orthant. You may assume that A is positive definite. (*Hint: You may use duality, and then use subgradient the rule for pointwise maximum*)

(f) $f(x) = \max_{\|y\|_\infty \leq b} y^T B x$, *i.e.*, the partial maximization of a multilinear problem over the infinity ball. (*Hint: You may use the subgradient rule for pointwise maximum*)