1.1 *(8 points) Parallel Subgradient Computation via 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 \( n = 5, m = 4 \).

```python
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 differentiation (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) 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

1.3 (6 points) Subdifferential sets. For each of the following convex functions, determine the subdifferential set at the specified point.

(a) \( f(x) = \text{ReLU}(x) \triangleq \max(x, 0) \) at \( x = 0 \)
(b) \( f(x) = \max(x, 0)^2 \) at \( x = 0 \)
(c) \( f(x_1, x_2, x_3) = |x_1| + 2|x_2| + 3|x_3| \) at \( (x_1, x_2, x_3) = (0, 0, 1) \).
(d) \( f(x_1, x_2, x_3) = \max\{|x_1|, |x_2|, |x_3|\} \) at \( (x_1, x_2, x_3) = (0, 0, 0) \).
(e) \( f(x) = e^{\left| x \right|} \) at \( x = 0 \) (\( x \) is a scalar).

\footnote{You can run your python script online on a Google Colaboratory notebook easily: \url{colab.research.google.com}}
(f) \( f(x_1, x_2) = \max\{x_1 + x_2 - 1, x_1 - x_2 + 1\} \) at \((x_1, x_2) = (1, 1)\).

1.4 (7 points) **Weak subgradient calculus.** For each of the following convex functions, explain how to calculate a subgradient at a given \( x \).

(a) \( f(x) = \text{ReLU}(a^T x + b) \), where \( x, a \in \mathbb{R}^n, b \in \mathbb{R} \), and \( \text{ReLU}(x) \triangleq \max(x, 0) \).

(b) \( f(x) = \max_{i=1, \ldots, m} \text{ReLU}(a_i^T x + b_i) \).

(c) \( f(x) = \max_{0 \leq t \leq 1} p(t) \), where \( p(t) = x_1 + x_2 t + \cdots + x_n t^{n-1} \).

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

(e) \( f(x) = \min_{Ay \preceq b} \|x - y\|^2 \), i.e., the square of the distance of \( x \) to the polyhedron defined by \( Ay \preceq b \). You may assume that the inequalities \( Ay \preceq b \) are strictly feasible. *(Hint: You may use duality, and then use subgradient the rule for pointwise maximum)*

(f) \( f(x) = \max_{Ay \preceq b} y^T x \), i.e., the optimal value of an LP as a function of the cost vector. *(You can assume that the polyhedron defined by \( Ay \preceq b \) is bounded.)* *(Hint: You may use the subgradient rule for pointwise maximum)*

1.5 (6 points + 4 extra credit points) **Computing gradients of expectations.** Suppose \( z \in \mathbb{R} \) is a random variable with distribution \( P \) and let \( F : \mathbb{R}^{d \times 1} \) be a collection of random functions indexed by \( z \). Define the expected function to be \( f(x) := \mathbb{E}_z[F(x, z)] \). We assume that \( x \mapsto F(x, z) \) is convex for all \( z \) and that \( \mathbb{E}_z[|F(x, z)|] < \infty \) for all \( x \). Additionally, \( f \) and \( x \mapsto F(x, z) \) are differentiable in \( x \).

(a) (3 points) Verify that \( f \) is also a convex function.

(b) (3 points) Use the subgradient inequality to show that \( \nabla_x f(x) = \mathbb{E}_z[\nabla_x F(x, z)] \).

That is, we can exchange the order of differentiation and integration for convex functions.

(c) (2 points extra credit) Assume \( n = 1 \). Abusing notation, the derivative of \( f \) is
\[
\nabla_x f(x) = \lim_{t \to 0} \frac{f(x + t) - f(x)}{t}.
\]

Part (b) shows that we can exchange the limit in this equation with the expectation operator when \( f \) is convex (convinve yourself that this is true). However, the order of limits and integrals may not be exchanged in general.

Let \( \{Y_k\} \) be a sequence of random variables with limit \( \bar{Y} \). The Dominated Convergence Theorem\(^2\) says that
\[
\lim_{k \to \infty} \mathbb{E}[Y_k] = \mathbb{E}[\bar{Y}]
\]
if \( |Y_k| \leq |Z| \) for all \( k \) and a random variable \( Z \) satisfying \( \mathbb{E}[|Z|] < \infty \). Show that Part (b) is consistent with measure-theoretic probability by proving that \( \nabla_x f(x) = \mathbb{E}_z[\nabla_x F(x, z)] \) using the Dominated Convergence Theorem.

\(^2\)See, for example, Cinlar, 2011 \[Cin11\].
(d) (1 point extra credit) Now suppose that $n > 1$. Use Part (c) to prove
\[ \nabla x f(x) = E_z[\nabla x F(x, z)] \]
by restricting $f$ to a scalar function along $e_i$, a vector from the
cardinal basis.

References