Problem 1 (10pts)
Consider the following scalar-valued function

\[ f(x, y, z) = x^2 y + \sin(z + 6y), \]

where \( x, y, z \in \mathbb{R} \).

(A) Compute partial derivatives with respect to \( x, y, \) and \( z \).

(B) We can consider \( f \) to take a vector \( \theta \in \mathbb{R}^3 \) as input where \( \theta = [x, y, z]^T \). Show the gradient \( \nabla_{\theta} f \) as a vector and evaluate it at \( \theta = [3, \frac{\pi}{2}, 0]^T \).
Problem 2 (10pts)
The purpose of this problem is to demonstrate Clairaut’s Theorem, which states that in general, the order in which you partial differentiate does not matter. Consider the following scalar-valued function,

\[ f(x, y, z) = x \sin(xy), \]

where \( x, y \in \mathbb{R} \).

(A) Compute \( \frac{\partial}{\partial x} \frac{\partial}{\partial y} f(x, y) \). This means we first compute the partial derivative of \( f \) with respect to \( y \), then compute the partial derivative of the resulting function with respect to \( x \). This is sometimes denoted \( \partial_{xy} f \).

(B) Compute \( \frac{\partial}{\partial y} \frac{\partial}{\partial x} f(x, y) \).

You should have gotten the same answer for parts (A) and (B), which demonstrates Clairaut’s Theorem. This holds more generally for \( n \) variables as well, that is, if \( f(x_1, x_2, \ldots, x_n) \) is a function of \( n \) variables. This result can be useful when differentiating functions, since it’s possible that it’s more convenient computationally to differentiate in a particular order.
Problem 3 (24pts)
Consider the following vector function from $\mathbb{R}^3$ to $\mathbb{R}^3$:

$$
f(x) = \begin{bmatrix}
\sin(x_1 x_2 x_3) \\
\cos(x_2 + x_3) \\
\exp(-\frac{1}{2}(x_3^2))
\end{bmatrix}
$$

(A) What is the Jacobian matrix of $f(x)$?

(B) Write the determinant of this Jacobian matrix as a function of $x$.

(C) Is the Jacobian a full rank matrix for all $x \in \mathbb{R}^3$? Explain your reasoning.
**Problem 4 (10pts)**
Compute the gradients for the following expressions. (You can use identities, but show your work.)

(A) \( \nabla_x \frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \)  
Assume \( x, \mu \in \mathbb{R}^n \) and invertible symmetric \( \Sigma \in \mathbb{R}^{n \times n} \).

(B) \( \nabla_x (c + Ax)^T (c - Bx) \)  
Assume \( x \in \mathbb{R}^n, c \in \mathbb{R}^m \) and \( A, B \in \mathbb{R}^{m \times n} \).
**Problem 5 (12pts)**

(A) The sigmoid function \( f : \mathbb{R} \rightarrow \mathbb{R} \) (also called the *logistic function*) is defined to be:

\[
f(z) = \frac{1}{1 + e^{-z}}
\]

Compute the derivative of the sigmoid function, i.e., \( f'(z) \). And verify that \( f'(z) = f(z) (1 - f(z)) \).

(B) The cost function of a very popular machine learning model logistic regression has the following form:

\[
c(\theta, x, y) = -y \log \left( \frac{1}{1 + e^{-\theta^T x}} \right) - (1 - y) \log \left( 1 - \frac{1}{1 + e^{-\theta^T x}} \right)
\]

with \( \theta \in \mathbb{R}^d, x \in \mathbb{R}^d, y \in \mathbb{R} \). Compute the partial derivative with regards to \( \theta \), i.e., \( \frac{\partial c(\theta, x, y)}{\partial \theta} \). And verify that \( \frac{\partial c(\theta, x, y)}{\partial \theta} = (f(\theta^T x) - y) x^T \).
Problem 6 (32pts)

In gradient descent, we attempt to minimize some function $f(x)$ by altering parameters $x \in \mathbb{R}^n$ according to the following formula:

$$x_{t+1} = x_t - \lambda (\nabla_x f(x_t))^T$$

for some small $\lambda \geq 0$ known as the learning rate or step size. We adjust $x$ so as to move in a direction proportional to the negative gradient. We will discuss gradient descent in more detail later in the course.

Consider the simple function $f(x) = x^T A x$ for a constant matrix $A \in \mathbb{R}^{n \times n}$.

(A) Implement a function $f(A, x)$ that takes as input an $n \times n$ numpy array $A$ and a 1D array $x$ of length $n$ and returns the value of $f$ as defined above.

(B) Implement a function $\text{grad}_f(A, x)$ that takes the same two arguments as above but returns $\nabla_x f(x)$ evaluated at $x$.

(C) Now implement a third and final function $\text{grad\_descent}(A, x_0, \lambda, \text{num\_iters})$ that takes the additional arguments $\lambda$, representing the learning rate $\lambda$ above, and $\text{num\_iters}$ indicating the total number of iterations of gradient descent to perform. The function should print the values of $x_t$ and $f(x_t)$ after each iteration of gradient descent.

(D) We will perform gradient descent on $f$ with $A = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$. Set the initial $x_0$ to be np.array([10, 10]). Run gradient descent for 50 iterations with learning rates of 1, 0.25, 0.1, and 0.01. What do you notice? Does $x_t$ always converge to the same value? Does our gradient descent algorithm work every time?
Problem 7 (2pts)
Approximately how many hours did this assignment take you to complete?

My notebook URL: https://colab.research.google.com/xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Changelog

- 23 March 2020 – Initial version.
- 26 March 2020 – Clarifications for problem 3.