Rohan Singh

Objective: To help you review concepts you already should know (from course prerequisites) and to help you identify any gaps in your knowledge.

Grading: Unlike other assignments in this class, your grade for this assignment will be based on effort. You will receive a "complete" if you complete the assignment and an "incomplete" if you do not complete the assignment.

***Deviation from Policy in Other Assignments*: Late submissions will NOT be accepted.**
Submission Instructions: Submit your solutions as a single PDF document electronically in Canvas. Scans of handwritten work are acceptable if they are legible, but typed solutions are preferred. Please name your document **FIRSTNAME_LASTNAME.pdf**, where FIRSTNAME and LASTNAME are your first (given) and last (family) names as they appear in Canvas.

Problems

Problem 1: Find the derivative of the following functions.

$$f(x) = a_0 + a_1x + a_2x^2 = 2a_2x$$

$$f(x) = \frac{1}{11 + e^{-x}} = (11 + e^{-x})^{-1} = -(11 + e^{-x})^{-2}(-e^{-x}) = \frac{e^{-x}}{(11 + e^{-x})^2}$$

Problem 2: Find the gradient of the following function at coordinates

$$x = 0, y = 1, f(x, y) = 2x^2 - 2y^2 + 4xy - 3$$

$$\delta f(x, y) = f_x(x, y)i + f_y(x, y)j = (4x + 4y)i + (-4y + 4x)j$$

$$\frac{\delta f}{\delta x}(0,1) = 4(0) + 4(1) = 4, \frac{\delta f}{\delta y}(0,1) = -4(1) + 4(0) = -4$$

$$\therefore \nabla \mathrm{f}(0,1) = (4, -4)$$

Problem 3: Please answer the following as True or False.

1) If during an iterative optimization algorithm all elements of the gradient vector are zero, your search has reached a locally optimal solution. True
2) For some objective function f(x), there is only one value for x at which f(x) is maximized. False
3) Maximizing function f(x) is the same as minimizing function g(x)=−f(x) True
4) The gradient vector of an objective function always points in the direction of an optimum. False

Problem 4:
Let:

- A be an N×N matrix
- x be an N×1 vector
- b be an N×1 vector
- y be an M×1 vector
- c be an M×1 vector

Assume N ≠ M. Determine the dimensionality of the following expressions. If the expression is not a valid mathematical operation, state "not valid" as your answer.

$a)$ $x^T x$ is a 1×1 scalar.
$b)$ Ax is a *N*×1 vector.
$c)$ yx^{-1} Not valid unless *x* is a non-singular square matrix.
$d)$ $(Ax)^T b$ is a 1×1 scalar.
$e)$ $c^T Ax$ is a 1×1 scalar.

Problem 5: A programmer from a rival university created the following script in Python:

```
import numpy as np
u = np.array([1, 2, 3], dtype='float')
v = np.array([4, 5, 6], dtype='float')
w = u.T * v
```

The programmer intended the script to define the following two vectors, $\mathrm{u} = [1.0, 2.0, 3.0]^\mathrm{T}$ and $\mathrm{v} = [4.0, 5.0, 6.0]^T$ and compute the product $\mathrm{w} = \mathrm{u}^\mathrm{T}\mathrm{v}$. Unfortunately, the programmer goofed up. The code does not do what they intended. Determine the following:
What result should the code produce?

The code should define two column vectors **u** and **v**, and then compute their outer product (also known as the tensor product), resulting in a matrix. The outer product of two vectors **u** and **v** is denoted as **u^T v** and results in a matrix where each element **w[i][j]** is the product of the **i**-th element of **u** and the **j**-th element of **v**. So, the expected result is a 3x3 matrix, and **w** should be:

```
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
```

What result does the code produce?

The code as written does not produce the intended result. Instead, it performs element-wise multiplication between **u** and **v** because the * operator with NumPy arrays is element-wise multiplication. Therefore, **w** will be a NumPy array containing the element-wise products of **u** and **v**.

How can the code be changed to operate as intended?

To achieve the intended behavior, which is the outer product of **u** and **v**, you can use the **np.outer()** function in NumPy. Here's the modified code:

```
import numpy as np

u = np.array([1, 2, 3], dtype='float')
v = np.array([4, 5, 6], dtype='float')
w = np.outer(u, v)
```

Problem 6: The following calculation is run in Python on a 64-bit Windows machine: 58.0-0.58*100.0. One would expect the answer to be zero, but it is not (the result is 7.105427357601002e-15). Explain why.

The result we are seeing, 7.105427357601002e-15, is due to the limitations of floating-point precision in computers. Computers use finite binary representations for floating-point numbers, and this can lead to tiny rounding errors in calculations involving real numbers. It's essentially a very small error that occurs due to the way computers store and handle floating-point values.

Problem 7: Why did you enroll in this course? Please briefly state your reasons and what you hope to gain by taking it. (The instructor is curious & this type of information can help improve the course.)

I enrolled in this course because I'm currently a **software engineer** @Tesla and I love software development so naturally I am interested in machine learning. I know what you are thinking… yes, I am studying mechanical engineering but I discovered last year through my co-op that I really enjoy software engineering so I've been self-teaching myself as many concepts as I can. I'm excited to see what I can learn in this class and hopefully utilize in the future.