1. Sequence Alignment [5 pts]

Given a string x with length n and another string y with length m, consider the following questions and justify your answers.

(a) (2.5 pts) What is the maximum length of the global alignment between x and y?

The maximum length of the global alignment between x and y is the sum of the length of both strings, **n + m**. To determine the maximum alignment, all the characters from both sequences must be included in the alignment. Gaps are included in the alignment to maximize its length where every character from one sequence aligns with a gap from the other sequence.

(b) (2.5 pts) What is the minimum length of the global alignment between x and y?

The minimum length of the global alignment between x and y would be the length of the longer string/sequence, max(n, m). The sequence would not have any additional gaps inserted into the sequence and the two sequences would be aligned end to end with the longer sequence dictating the alignment length.

2. Sequence Alignment Practice [20 pts]

Consider two DNA sequences x = ATACGATT and y = GTAGCCTATAAGTTA. In this question, we will align the two sequences using a score of +1 for a match, -1 for a mismatch, and -1 for insertion/deletion (i.e., gap). Note that in this problem, we will align sequences by maximizing the alignment score (instead of minimizing the alignment cost).

(a) (10 pts) Align the two sequences using the global alignment algorithm introduced in the lecture. You need to i) compute the final alignment score, ii) fill out the following dynamic programming table (i.e., fill in all cells with its alignment scores), and iii) highlight the path of the optimal alignment using backtrace. If there are multiple optimal solutions, you can highlight any of them.

i. Final Alignment Score: -3

ii.

		G	T	A	G	C	C	T	A	T	A	A	G	T	T	A
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15
A	-1	-1	-2	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13
T	-2	-2	0	-1	-2	-3	-4	-3	-4	-5	-6	-7	-8	-9	-10	-11
A	-3	-3	-1	1	0	-1	-2	-3	-2	-3	-4	-5	-6	-7	-8	-9
C	-4	-4	-2	0	0	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
G	-5	-3	-3	-1	1	0	0	-1	-2	-3	-4	-5	-4	-5	-6	-7
A	-6	-4	-4	-2	0	0	-1	-1	0	-1	-2	-3	-4	-5	-6	-5
T	-7	-5	-3	-3	-1	-1	-1	0	-1	1	0	-1	-2	-3	-4	-5
T	-8	-6	-4	-4	-2	-2	-2	0	-1	0	0	-1	-2	-1	-2	-3

iii. Shown in table above

(b) (10 pts) Align the two sequences using the local alignment algorithm introduced in the lecture, then compute the final alignment score, fill out the dynamic programming table, and highlight the backtrace path as in (a). If there are multiple optimal solutions, you can highlight any of them.

i. Final Alignment Score: 4

ii.

		G	T	A	G	C	C	T	A	T	A	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	1
T	0	0	1	0	0	0	0	1	0	2	1	0	0	1	1	0
A	0	0	0	2	1	0	0	0	2	1	3	2	1	0	0	2
C	0	0	0	1	1	2	1	0	1	1	2	2	1	0	0	1
G	0	1	0	0	2	1	1	0	0	0	1	1	3	2	1	0
A	0	0	0	1	1	1	0	0	1	0	1	2	2	2	1	2
T	0	0	1	0	0	0	0	1	0	2	1	1	1	3	3	2
T	0	0	1	0	0	0	0	1	0	1	1	0	0	2	4	3

iii. Shown in table above

3. Sigmoid and Softmax Functions [10 pts]

*********ON NEXT PAGE*****

3 Sigmoid and Softmax Functions [10 pts]

(a) Show that for a given constant $c \in R$, the following equality holds true:

$$\frac{\exp(z_k + c)}{\sum_{j=1}^{K} \exp(z_j + c)} = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

$$\frac{\exp(z_k + c)}{\sum_{j=1}^{K} \exp(z_j + c)}$$

$$= \frac{\exp(c) \cdot \exp(z_k)}{\exp(c) \cdot \sum_{j=1}^{K} \exp(z_j)}$$

$$= \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

(b) Using the result from (a), show that when $K = 2$, the softmax function is equivalent to the sigmoid function.

$K=1$ $\sigma(\vec{z})_1 = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$ $\qquad \sigma(z) = \frac{1}{1 + \exp(-z)}$

$K=2$ $\sigma(\vec{z})_2 = \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2)}$ $\qquad \sigma(\vec{z})_1 + \sigma(\vec{z})_2 = 1$

If $z = z_1 - z_2$:

$$\sigma(z) = \frac{1}{1 + \exp(-(z_1 - z_2))} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}$$

4 Manhattan Tourist Problem [10 pts]

In our lectures, we have explored the Manhattan Tourist Problem as a practical application of dynamic programming concepts in computational biology. This problem involves finding the path with the highest number of attractions for a tourist navigating through a grid that represents the streets of Manhattan. Consider a grid of size n × m, where n represents the number of streets (horizontal edges) and m represents the number of avenues (vertical edges). The grid is oriented such that the tourist starts at the top-left corner and aims to reach the bottom-right corner, moving only to the right or downward.

The brute-force approach to solving this problem would involve enumerating all possible paths from the start to the end point and selecting the path that maximizes the number of attractions. However, this method is computationally inefficient because there can be many valid paths to enumerate.

1. (a) (8 pts) Calculate the total number of valid paths from the top-left corner to the bottom-right corner in an n × m grid. Provide a mathematical expression for your answer and a brief explanation of your reasoning.

$$Total\ Paths = \binom{n+m}{m} = \frac{(n+m)!}{m! \cdot n!}$$

$$\binom{n+m}{m} = \binom{n+m}{n}$$

The equation represents the number of ways to arrange the number of right steps, m among $n+m$ or the number of down steps, n among $n+m$ steps. In the Manhattan Tourist problem, every path from the top-left to the bottom-right corner consists of a total of $n+m$ steps, comprising exactly n downward movements and m rightward movements. So, if you want to move down a n number of times, the rest of the steps must be rightward movements. This works the other way too, where if you move to the right a certain number of times (m) out of the $n + m$ total steps, the remaining steps will be the downward steps, n. This makes it a combination problem where you need to only use one binomial coefficient; both expressions are symmetric in terms of combination properties.

2. (b) (2 pts) Discuss the implications of your findings on the feasibility (computational efficiency) of using a brute-force strategy (as compared to the dynamic programming algorithm) for solving the Manhattan Tourist Problem, especially as the size of the grid increases.

The brute-force strategy is not computationally efficient and gets less and less efficient as the grid size increases. If you look at the expression in part A, you can see that the rapidly growing factorial function is included. Hence, the number of paths increases exponentially with the size of the grid. Large grids make the brute force strategy computationally infeasible and impractical due to the time and resources required to carry out the evaluation of all paths. The dynamic programming algorithm approach is more practical and much more computational efficient than the brute force strategy. The dynamic programming approach solves the problem in polynomial time. It will solve smaller subproblems and reuse solutions to solve the bigger problem as a whole, $O(n\ x\ m)$. With the use of dynamic programming, redundant calculations are avoided and much more feasible when dealing with larger grids.

5 Precision and Recall [5 pts]

Using the COVID-19 testing scenario as a case study, we can gain a more intuitive understanding of many metrics used for evaluating the performance of classification models. In this scenario, we categorize the infection status of individuals (ground truth) as either infected or non-infected, and the test outcomes (model predictions) as positive or negative. This setup allows us to explore the definitions of the evaluation metrics in a concrete manner. For example, we showed that the specificity metric could be interpreted as "the fraction of all non-infected people who got a negative test result". Please review the definition of precision and recall in our slides and interpret the meanings of these two metrics in the context of the COVID-19 test in a similar way.

In the context of COVID-19, we can look at the performance of a test in identifying infected individuals. Precision is the positive predictive value in the context of COVID-19 and can be analyzed as the percentage/fraction of individuals who tested positive that are truly infected. Precision looks at the accuracy of the test when it gives a positive result. Recall is also known as the sensitivity and is the true positive rate. In terms of COVID-19 context, recall is the percentage/fraction of truly infected individuals who are correctly identified by the test. Recall looks at the tests ability to detect all the positive cases.

Instructions on Jupyter notebook

In the next two questions, you will need to work with the Jupyter notebooks we provided. You can run the Jupyter notebook on your local computer/laptop or Google Colab. We recommend running the notebook on Google Colab because in this way you do not have to install software on your local computer/laptop. Detailed steps are given below (also in lecture slides):

Access the notebooks from the link we provided in the following questions. Complete all questions mentioned in the notebook.

After completing all subquestions, you can export your finished notebook by clicking "File - Download - Download .ipynb".

6. Regression [25 pts]

In this problem, you will implement linear regression using gradient descent and apply it to real-world data to predict the quantitative measure of disease progression. Please see the Jupyter notebook hw1 reg.ipynb for instructions. You can access the notebook file at:

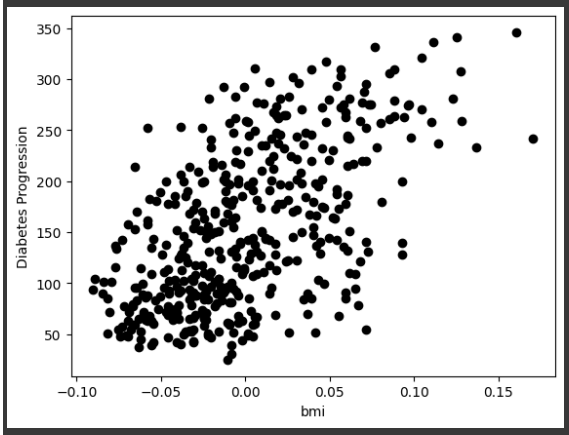
https://colab.research.google.com/drive/1LVlt4Y9BSmuLnylsp5cVOPQ3MGMTx3DN?usp=sharing

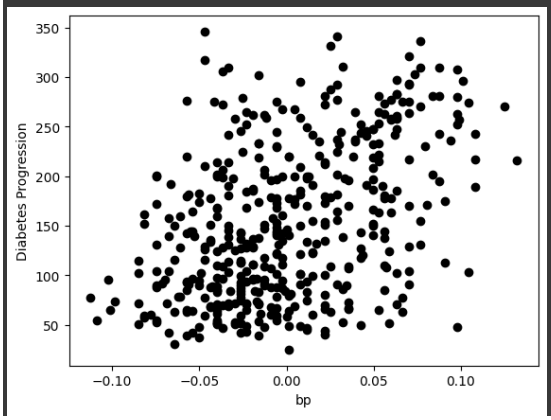
a. [10 pts] Task1 - Exploring the features.
b. [15 pts] Task2 - Gradient descent for linear regression.

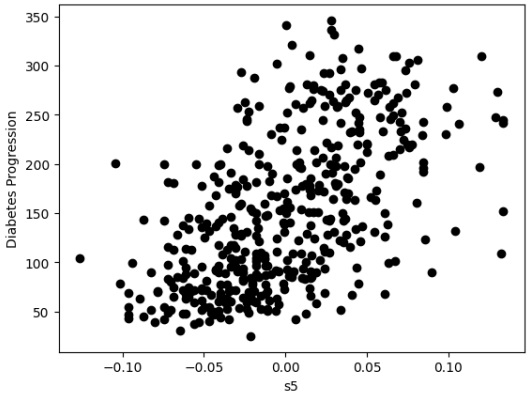
Task 1: Exploring the features:

Execute the code below
Answer the questions below and report it in your assignment PDF file.
(1) What are the three features you will want to keep if you can only use three features to predict disease progression?

You will want to keep the BMI , BP, and s5 as they have the strongest positive correlations out of all the features.
Explain the reason in one sentence. (Hint: look at the output from pearson's r) <5 points>
(2) Use the function, PlotDataset, to visualize the relationship between your chosen features and the disease progression (Include your plots in your assignment PDF). <5 points>







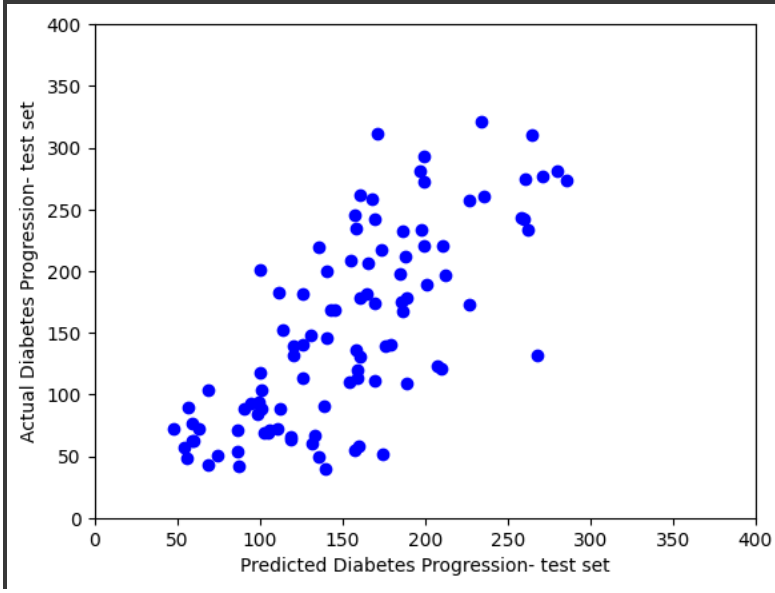
Task 2: Gradient descent for linear regression:

Implement the gradient descent algorithm for linear regression. (Complete the code for gradient descent).

Answer the questions below and report it in your assignment pdf file:
(1) What is the MSE in your final iteration on your training set? What is the MSE on your test set? Note: Please use the formula $MSE=12n\sum ni=1(f\theta(xi)-yi)2$ <8 points >

MSE on final iteration of the training set: MSE: 1463.661919
MSE on test set: test set MSE: 1386.871429506086

(2) Use the code below to plot the predictions and the actual diabetes progression, and include the figure in your assignment PDF. <7 points>



7 PyTorch Warm-ups [25 pts]

This is a programming assignment. Please see the Jupyter notebook hw1 torch intro.ipynb for de- tails. Please upload the zip file including the hw1 torch intro.ipynb in this problem and the notebook hw1 reg.ipynb to Gradescope. You can access the notebook file at:

https://colab.research.google.com/drive/19xByhBf1mIqr1Z-ZpMnNyLLAQgDTtaMo?usp=sharing