Declaration

1. I hereby declare the entirety of the assessment was completed by myself.
2. All online materials and resources used in the completion of this assessment are cited.
3. I agree not to share the questions with others or post the questions on public forums.

Signature: OUYANG Hui

Name: OUYANG Hui

This is the link to my github repository:

https://github.com/Scarletlake/MBDS_solution

The repository contains the source codes for this test. For questions that require the explanation of the solutions, I add the explanation in the README file in that folder of that question. Also the screen shots of explanation is attached to the answer sheet.

The codes were implemented with python. The code for each question can be found in the folder of that question and it is in the file with the name “Q+Question_Number.py” (e.g. "Q1.py").

Due to the limited time, I only answered the questions that I was sure about and that were not beyond my expertise.

Programming interview

Name: OUYANG Hui

Date: 24/02/2021

Job applied for: Master of Science in Biomedical Data Science

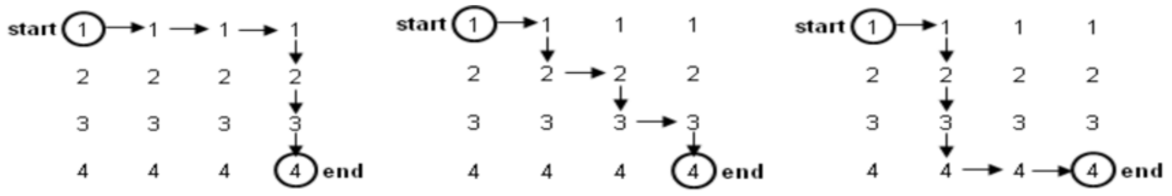
Tick the appropriate box:
o Citizen
o Permanent Resident
[✓ **o Student Pass**]
o Others, please specify: ______________________

For each question, please submit your source code and output files in the required format according to the README and example output files. In addition, for questions requiring more explanation of your method, please write your explanations in the provided space in this question sheet. If you use any library functions, please also explain how the functions work.

1 Operations for the right sum

Given a m x n matrix, we want to connect from top left corner (starting point, first row, first column) to bottom right corner (ending point, mth row, nth column). Only 2 operations are allowed: Right (R) or Down (D). Numbers that are passed through will be summed up. Given any summed number, you are required to find out the operations needed to get the number.

Example: (m=4, n=4 square matrix) with operations needed to get the desired sum.



sum=13 (RRRDDD) sum=16 (RDRDRD) sum=19 (RDDDRR)

a. For m=9, n=9 matrix, find the operations for the following summed numbers: 65, 72, 90, 110.

b. For m=90,000, n=100,000 matrix (90,000 rows, 100,000 columns), find the operations for the following summed numbers: 87127231192 and 5994891682.

[Output file: output_question_1]

2 Equivalent networks

This question does not involve coding, please write your answers on a separate piece of paper.

2.1 Background on multi-layer perceptron

A multilayer perceptron (MLP) is a network which maps some input values to produce some output. The network consists of layers of nodes connected by weights. The input at the first layer is transformed layerwise to get the output. Figure 1 shows a single layer MLP with 2 input nodes connecting to 2 output nodes. The weights between the layers and the bias at each upper layer node controls the mapping. Assuming a linear activation function, the output values in figure 1 is given by

$$\begin{aligned} a_0^{(1)} &= w_{0,0}^{(1)} a_0^{(0)} + w_{0,1}^{(1)} a_1^{(0)} + b_0^{(1)} \\ a_1^{(1)} &= w_{1,0}^{(1)} a_0^{(0)} + w_{1,1}^{(1)} a_1^{(0)} + b_1^{(1)} \end{aligned} \tag{1}$$

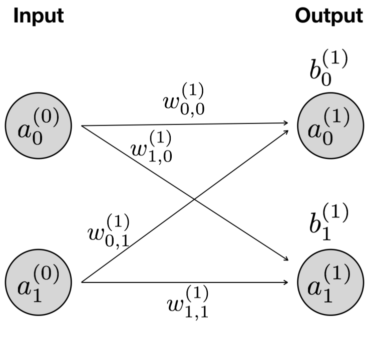
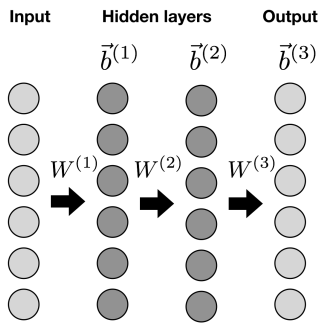


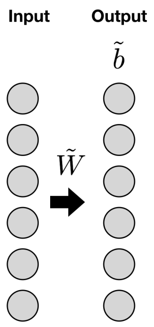
Figure 1: One layer MLP.

$a_i^{(l)}$: value of i^{th} node at layer l
$b_i^{(l)}$: bias acting on i^{th} node at layer l
$w_{i,j}^{(l)}$: weight connecting j^{th} node at layer $l-1$ to i^{th} node at layer l

Eq. 1 can be expressed more compactly using matrix and vectors, $\vec{a}^{(1)} = W^{(1)}\vec{a}^{(0)} + \vec{b}^{(1)}$, where $\vec{a}^{(0)} = (a_0^{(0)} a_1^{(0)})^T$, $\vec{a}^{(1)} = (a_0^{(1)} a_1^{(1)})^T$, $\vec{b}^{(1)} = (b_0^{(1)} b_1^{(1)})^T$, $W^{(1)} = \begin{pmatrix} w_{0,0}^{(1)} & w_{0,1}^{(1)} \\ w_{1,0}^{(1)} & w_{1,1}^{(1)} \end{pmatrix}$.



(a) Network 1: MLP with multiple hidden layers.



(b) Network 2: MLP with no hidden layers.

Figure 2

There can be any number of nodes in each layer. We can also have multiple hidden layers in between the input and output layers, where we use the same transformation equations. For the network in Figure 2a with two hidden layers, the output at the second and third layers are given by

$$\vec{a}^{(2)} = W^{(2)}\vec{a}^{(1)} + \vec{b}^{(2)} \tag{2}$$

$$\vec{a}^{(3)} = W^{(3)}\vec{a}^{(2)} + \vec{b}^{(3)} \tag{3}$$

2.2 Question: Formulating equivalent networks

Two networks are said to be equivalent if, they have the same number of input and output nodes, and for all inputs, the output of both networks are identical. Two neural networks are shown in Figure 2a and 2b with different number of hidden layers. Given Network 1's weights and biases values, find out Network 2's weights ($\tilde{W}$) and bias ($\tilde{b}$) such that the two networks are equivalent.

The input of one layer is the output from the former layer

So for Network 1: $\vec{a}^{(1)} = \vec{w}^{(1)} \vec{a}^{(0)} + \vec{b}^{(1)}$

$\vec{a}^{(2)} = \vec{w}^{(2)} (\vec{w}^{(1)} \vec{a}^{(0)} + \vec{b}^{(1)}) + \vec{b}^{(2)} = \vec{W}^{(2)} \vec{W}^{(1)} \cdot \vec{a}^{(0)} + \vec{W}^{(2)} \vec{b}^{(1)} + \vec{b}^{(2)}$

$\vec{a}^{(3)} = \vec{W}^{(3)} (\vec{W}^{(2)} \vec{W}^{(1)} \cdot \vec{a}^{(0)} + \vec{W}^{(2)} \vec{b}^{(1)} + \vec{b}^{(2)}) + \vec{b}^{(3)} = \vec{W}^{(3)} \vec{W}^{(2)} \vec{W}^{(1)} \vec{a}^{(0)} + \vec{W}^{(3)} \vec{W}^{(2)} \vec{b}^{(1)} + \vec{W}^{(3)} \vec{b}^{(2)} + \vec{b}^{(3)}$

For Network 2: $\tilde{a} = \tilde{w} \vec{a}^{(0)} + \tilde{b}$

Since they are equivalent: $\tilde{w} = \vec{W}^{(3)} \vec{W}^{(2)} \vec{W}^{(1)}$

$\tilde{b} = \vec{W}^{(3)} \vec{W}^{(2)} \vec{b}^{(1)} + \vec{W}^{(3)} \vec{b}^{(2)} + \vec{b}^{(3)}$

3 Multilayer perceptron for regression

Regression analysis is a method that is used for analysing the relationship between a set of independent variables and a dependent variable.

In this question, you are given a training dataset that consists of three independent variables x_1, x_2, x_3 and a dependent variable y=f(x_1, x_2, x_3). You are required to construct the Multi Layer Perceptron (MLP) model shown in Figure 3 to predict the dependent variable y by using the values of independent variables x_1, x_2 and x_3. After constructing and training the MLP model with training dataset, you will predict the values of y for the test dataset.

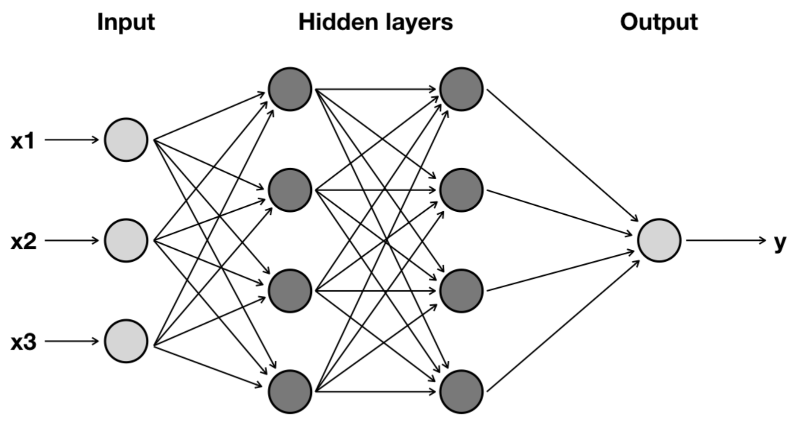


Figure 3: MLP Model: 3-input, 4x4 hidden layers and 1-output

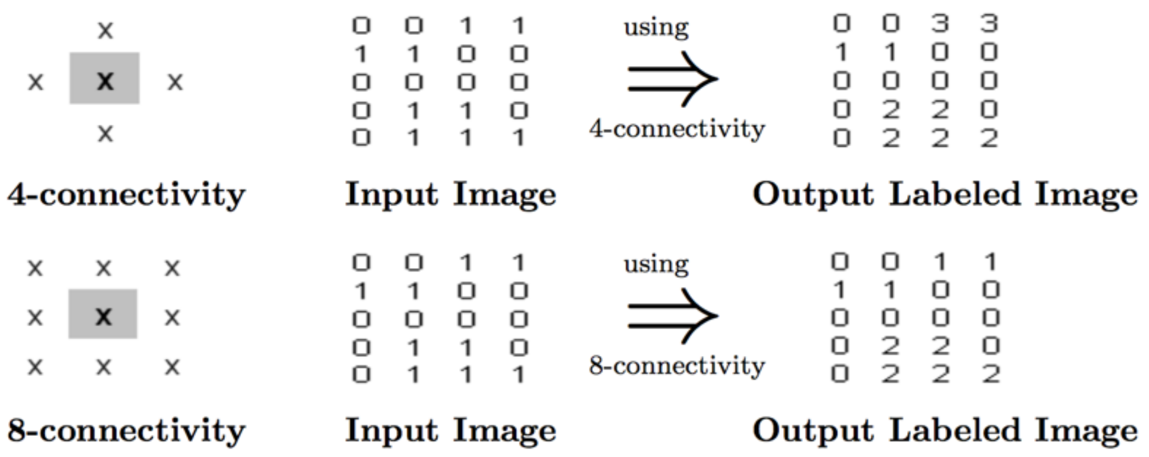
[Input file1: train_data.txt (tab-seperated file: each column corresponds to an independent variable: x_1, x_2 and x_3, respectively)]
[Input file2: train_truth.txt (ground-truth values of dependent variable: y)]
[Input file3: test_data.txt (tab-seperated file: each column corresponds to an independent variable: x_1, x_2 and x_3, respectively)]
[Output file1: test_predicted.txt (write the predicted y values)]

Note: You can use Tensorflow, Keras and the other frameworks to construct your model.

4 Connected components

Write a code to find out connected components for a given image. When a group of pixels in the image is “connected” to each other, they are said to form a connected cluster and we refer to this cluster as connected component. In imaging, pixels can be connected in 4 neighbors (4-connectivity) or 8 neighbors (8-connectivity). In the example given below, the input image will result in 3 connected components if using 4-connectivity or 2 connected components if using 8-connectivity. You can implement either 4-connectivity or 8-connectivity connected components.

*Remember to explain the workings of your code. If you use library functions, please explain how they work.



[Input file: input_question_4; Output file: output_question_4]

For the purpose of this question, functions to find clusters by 4-connectivity is implemented.

The image is stored as a 2d array, each element is a list `[cluster_num, labled]` contains the cluser number of the pixel and if it is labeld. labled == 0 means not labled, labled == 1 means it is labled.

To find the cluster the pixel belongs to, function `find_cluster` is called recursively. It will terminate if if it not connect to a pixel or it's neighbors is allready labled.

```python
def find_cluster(loc, input_image, row, col):
    input_image[loc[0]][loc[1]][1] = 1        # Mark it as labeld

    # 4 neighbors: up, right, down, left pixels
    neighbors = [(loc[0]-1, loc[1]), (loc[0], loc[1]-1), (loc[0]+1, loc[1]), (loc[0], loc[1]+1)]

    for neighbor_loc in neighbors:

        # location out of boundary
        if (neighbor_loc[0] < 0 or neighbor_loc[0] >= row or
            neighbor_loc[1] < 0 or neighbor_loc[1] >= col ):
            pass

        # terminate if it not connect to a new pixel
        elif (input_image[neighbor_loc[0]][neighbor_loc[1]][0] == 0 or
              input_image[neighbor_loc[0]][neighbor_loc[1]][1] == 1 ):
            pass

        # if it connects new clusters
        # label this pixel and final new neighbours by recursively call this function
        else:
            input_image[neighbor_loc[0]][neighbor_loc[1]][0] = input_image[loc[0]][loc[1]][0] # Lable the cluster r
            find_cluster(neighbor_loc, input_image, row, col);
```

5 Coloring

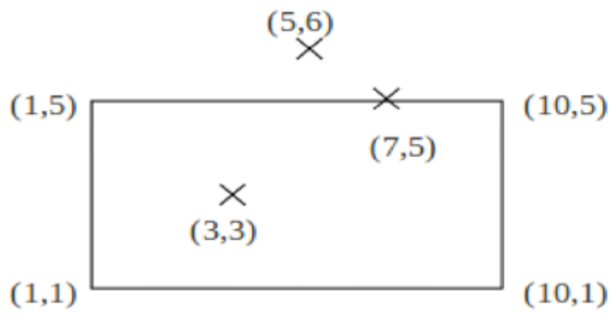
Given a L by L square grid, consider 4 neighbors connections. Given L^2 beads of different colors. Put all the beads onto the grid and the penalty for putting any two beads of the same color as neighbor is one. Total penalty is the sum of all panelties. Your task is to find a way to put the beads onto the square grid with least penalty. Perform this tasks with

1. L=5 with 12 red beads (R) and 13 blue beads (B)
2. L=64 with 139 red beads (R), 1451 blue beads (B), 977 green beads (G), 1072 white beads (W), 457 yellow beads (Y)

[Output file1: output_question_5_1 (write the grid configuration (bead placement) for part 1)]
[Output file2: output_question_5_2 (write the grid configuration (bead placement) for part 2)]

6 Points inside/outside polygon

Given a sequence of points that form a polygon, you are required to tell if a list of points are either inside or outside the polygon.



Example: Given a sequence of points for polygon: (1,1),(1,5),(10,5),(10,1). The following is the outcome of points tested.

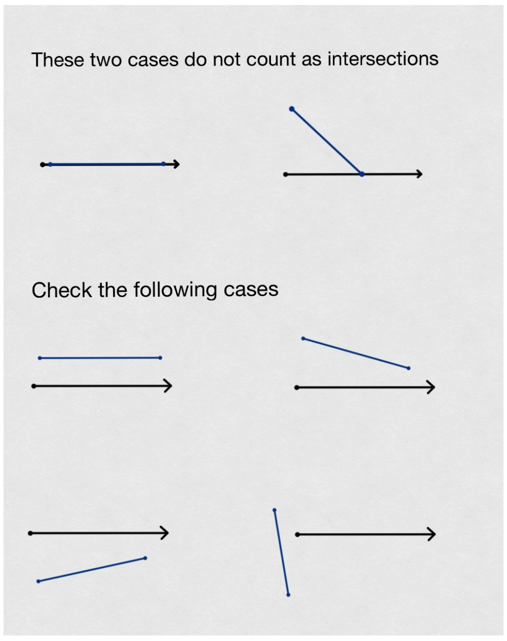
Point	State
(3,3)	Inside
(7,5)	Inside
(5,6)	Outside

*Remember to explain the workings of your code. If you use library functions, please explain how they work. [Input file: input_question_6_polygon, input_question_6_points; Output file: output_question_6]

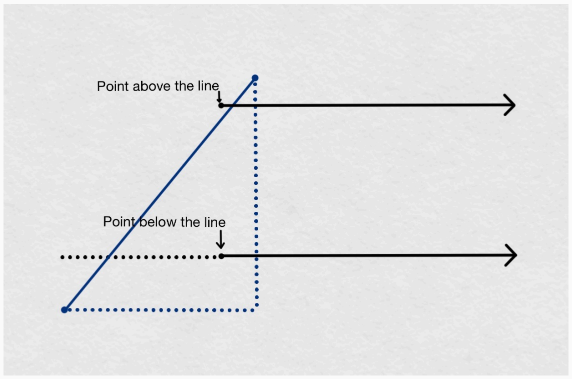
To check if a point is inside a polygon, a ray is drawn from the point from left to right, if this ray intersects with edges of polygon for odd number of times, then it is inside the polygon.

The following situations should not be considered as intersection.

- The ray overlaps the edge
- The ray pass through one of the edge and it is below the edge


These two cases do not count as intersections
Check the following cases

To test if the point of intersection is at the left or right of side of the ray, the edge is considered as a linear function where the vertices are the points on that line. If the testing point is above the line, then the ray does not intersect with the edge.



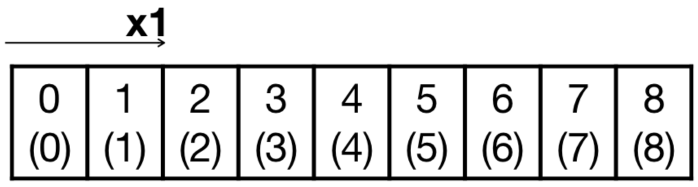
```
# The point of intersection is at the left of the ray
    # Consider the edge as a function, if the point is above the
    # line, then the ray does not intersect with the edge.
    # slop = (v1_x-v2_x)/(v1_y-v2_y)
    elif ((polygon_v1[0]-polygon_v2[0])/(polygon_v1[1]-polygon_v2[1])*(point[0]-polygon_v1[0]) >
          (point[1]-polygon_v1[1])):
        return False

    else:
        return True
```

7 Coordinates-to-index & Index-to-coordinates

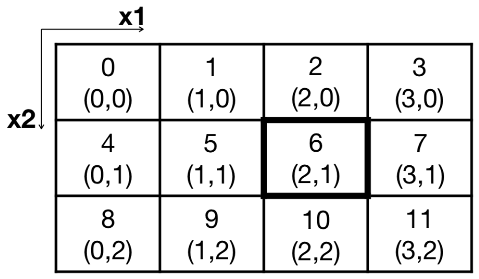
This question involves both deriving mathematical equations and coding, please write your derivations on a separate piece of paper. **For this question, do not use build-in library functions, code all the implementations.**

7.1 2-dimension

For one dimensional grid, indexing the cells is just going along one direction (1-dimensional raster scan) as shown in Figure 4a. For two dimensional grid, indexing the cells is done by 2-dimensional raster scan as shown in Figure 4b. For example, a 2-dimensional grid with sizes (L_1, L_2)=(4, 3) is shown in Figure 4b. In this grid, coordinates (x_1, x_2)=(2, 1) corresponds to index $I = 6$, and vice versa.



(a) 1D raster scan



(b) 2D raster scan

Figure 4

a) Derive a mathematical equation converting coordinates to index and derive the inverse equations converting index into coordinates in 2-dimensional grid.

b) Given 2-dimensional grid with sizes (L_1, L_2) = (50, 57), write a code by yourself to do this **(do not use library functions, code all the implementations)**:

- Write a code to convert given coordinates to index
  (i.e. given x_1 and x_2, find index I)

  [Input file1: input_coordinates_7_1.txt (tab-seperated file: each column corresponds to a dimension)]
  [Output file1: output_index_7_1.txt (write the calculated index values)]

- Write a code to convert given index to coordinates
  (i.e. given I, find coordinates x_1 and x_2)

  [Input file2: input_index_7_1.txt]
  [Output file2: output_coordinates_7_1.txt (write the calculated coordinates)]

a) Let the grid has sizes (L_1, L_2). I for index

- Coverting coordinates to index :

  $\text{Index}(x_1, x_2) = x_1 + L_1 \cdot x_2$

- Coverting index to coordinates :

  $x_2 = I \,//\, L_1$ (the quotient of I / L_1)

  $x_1 = I \,\%\, L_1$ (the remainder of I / L_1)

7.2 d-dimension

For a grid of d-dimension with sizes $L_1, L_2, L_3, \cdots, L_d$, indexing is done by d-dimensional raster scan.

For example, a 3-dimensional grid with sizes (L_1, L_2, L_3)=(4, 3, 2) is shown in Figure 5. In this grid, coordinates (x_1, x_2, x_3)=(2, 0, 1) corresponds to index $I = 14$, and vice versa.

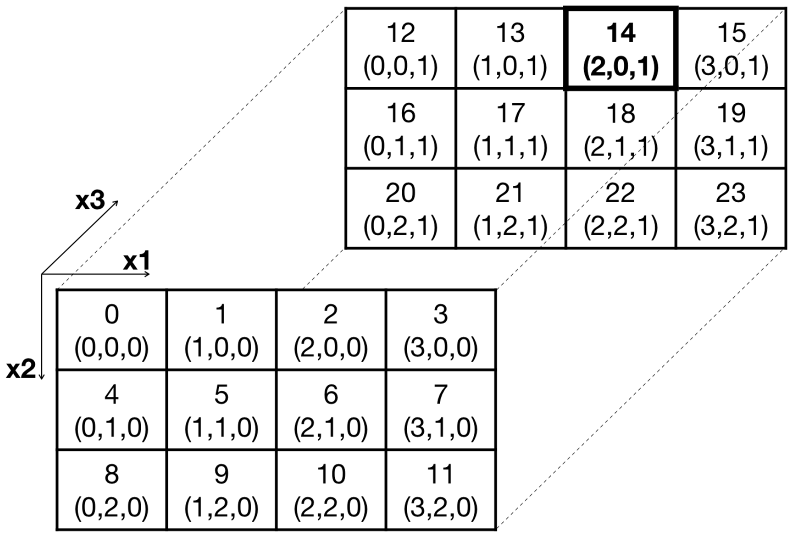


Figure 5: A 3-dimensional grid with sizes (L_1, L_2, L_3)=(4, 3, 2).

a) Derive a mathematical equation converting coordinates to index and derive the inverse equations converting index into coordinates in d-dimensional grid.

b) Given 6-dimensional grid with sizes (L_1, L_2, L_3, L_4, L_5, L_6)=(4, 8, 5, 9, 6, 7), write a code by yourself to do this **(do not use library functions, code all the implementations)**:

- Write a code to convert given coordinates to index
  (i.e. given x_1, x_2, x_3, x_4, x_5, x_6, find index I)

  [Input file1: input_coordinates_7_2.txt (tab-seperated file: each column corresponds to a dimension)]
  [Output file1: output_index_7_2.txt (write the calculated index values)]

- Write a code to convert given index to coordinates
  (i.e. given I, find coordinates x_1, x_2, x_3, x_4, x_5, x_6)

  [Input file2: input_index_7_2.txt]
  [Output file2: output_coordinates_7_2.txt (write the calculated coordinates)]

a) Let the grid has sizes $(L_1, L_2, \dots\dots L_n)$, I for index

- Coverting coordinates to index :

  $I(x_1, x_2 \cdots x_n) = f_1 x_1 + f_2 x_2 + \cdots f_n x_n$

  where $f_1 = 1$, $f_2 = L_1$, $f_3 = L_1 \times L_2$, $f_4 = L_1 \times L_2 \times L_3, \cdots$ $f_n = L_1 \times L_2 \times L_3 \cdots L_{n-1}$

  That is $f_1 = 1$, $f_2 = L_1$, $f_3 = f_2 \cdot L_2$, $f_4 = f_3 \cdot L_3, \dots$ $f_n = f_{n-1} \cdot L_{n-1}$

- Coverting index to coordinates :

  $x_n = I // f_n$

  $x_{n-1} = (I \% f_n) // f_{n-1}$

  $f_{n-2} = ((I \% f_n) \% f_{n-1}) // f_{n-2}$

  $\vdots$

  $x_1 = ((\cdots(I \% f_n) \% f_{n-1}) \cdots \% f_3) \% f_2$

8. Enzyme Kinetics

Enzymes are catalysts that help convert molecules that we will call substrates into other molecules that we will products. They themselves are not changed by the reaction. Within cells, enzymes are typically proteins. They can speed up biological reactions, sometimes by up to millions of times. They are also regulated by a very complex set of positive and negative feedback systems. Computational biologists are painstakingly mapping out this complex set of reactions. In this problem, we will model and simulate a simplified enzyme reaction.

An enzyme *E* converts the substrate *S* into the product *P* through a two-step process. First, *E* forms a complex with *S* to form an intermediate species *ES* in a reversible manner at the forward rate k_1 and reverse rate k_2. The intermediate *ES* then breaks down into the product *P* at a rate k_3, thereby releasing *E*. Schematically, we write

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \overset{k_3}{\rightarrow} E + P$$

8.1. Using the law of mass action, write down four equations for the rate of changes of the four species, *E*, *S*, *ES*, and *P*.

8.2. Write a code to numerically solve these four equations using the fourth-order Runge-Kutta method. For this exercise, assume that the initial concentration of *E* is 1 μM, the initial concentration of *S* is 10 μM, and the initial concentrations of *ES* and *P* are both 0. The rate constants are: k_1=100/μM/min, k_2=600/min, k_3=150/min.

8.3. We define the velocity, *V*, of the enzymatic reaction to be the rate of change of the product *P*. Plot the velocity *V* as a function of the concentration of the substrate *S*. You should find that, when the concentrations of *S* are small, the velocity *V* increases approximately linearly. At large concentrations of *S*, however, the velocity *V* saturates to a maximum value, V_m. Find this value V_m from your plot.

8.1 There are two reactions : $E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES$

$ES \xrightarrow{k_3} E + P$

According to the law of mass action :

$$\frac{k_2}{k_1} = \frac{[S][E]}{[ES]},$$

$$\begin{cases} V_E = k_2[ES] - k_1[S]\cdot[E] + k_3[ES] \\ V_S = k_1[S]\cdot[E] - k_2[ES] \\ V_{ES} = k_1[S][E] - k_2[ES] - k_3[ES] = -V_E \\ V_P = k_3\cdot[ES] \end{cases}$$

9. Read the following (https://www.nature.com/articles/s42256-019-0048-x).

Explain in no more than 2 sides of a page (Arial, font size 12, single spacing) whether the Rashomon set is realistic and can be used to meaningfully capture explainable models. You may include references (these do not count to the 2-page limit). Please avoid the use of excessive generics in your response.