Lab 8

Matrices

Sup'Biotech 3

Python

Pierre Parutto

November 23, 2016

Preamble

Document Property

Authors	Pierre Parutto
Version	1.0
Number of pages	5

Contact

Contact the assistant team at: supbiotech-bioinfo-bt3@googlegroups.com

Copyright

Contents

1 Introduction

In this lab we will manipulate matrices using the library numpy.

2 Characteristics of Matrices

2.1 Only Negative Values?

Write a function `all_neg(m: array) -> bool` that returns `True` if the matrix `m` contains only strictly negative values and `False` otherwise. If the matrix is empty, the function must return `True`.

Example

```
>>> from numpy import array
>>> all_neg(array([[-1,-2,-3],[-4,-5,6]]))
False
>>> all_neg(array([[-1,-2,-3], [-3,-2,-1]]))
True
```

2.2 Has negative value?

Write a function `has_neg(m: array) -> bool` that returns `True` if the matrix `m` possesses at least one strictly negative value and `False` otherwise. If `m` is empty the function must return `False`.

Example

```
>>> from numpy import array
>>> has_neg(array([[1,2,3], [4,5,5]]))
False
>>> has_neg(array([[-1,3],[4,5]]))
True
```

2.3 Maximum Value

Write a function `max_mat(M: array) -> int` that returns the greatest value in the matrix `M`. If `m` is empty your function must return `None`.

Example

```
>>> from numpy import array
>>> max_mat(array([[1,9], [2,3]]))
9
>>> max_mat(array([[1,2,5], [12,-2,0]]))
12
```

2.4 Common Values

Write a function `common_values(A: array, B: array) -> list` that returns the list of the common values between the two matrices `A` and `B`. The matrices `A` and `B` have the same size.

Example

```
>>> from numpy import array
>>> common_values(array([[3,-5],[4,8]]), array([[1,2], [3,8]]))
[3,8]
>>> common_values(array([[1,2,3],[-1,4,9]]), array([[3,1,4], [9,8,2]]))
[1,2,3,4,9]
```

2.5 Positions Of Maximum

Write a function `pos_max(M: array) -> list` that returns the positions of the maximum value in the matrix `M`. The returned value is a list of tuples `[(i,j), ...]` where `i` is the line number and `j` is a column number.

Example

```
>>> from numpy import array
>>> pos_max(array([[1,3],[4,6]]))
[(1,1)]
>>> pos_max(array([[1,2,3],[3,-4,3]]))
[(0,2), (1,0), (1,2)]
```

3 Computations on Matrices

3.1 Matrix Transposition

Transposition is the operation that transforms lines into columns and vice versa. For a matrix $M = m_{i,j}$ of size $m \times n$, its transpose is written as: ${}^tM$, has size $n \times m$ and follows the formula:

$$ {}^tm_{i,j} = m_{j,i} $$

Write a function `transpose(M: array) -> array` that returns the transpose of the matrix `M`.

Example

```
>>> from numpy import array
>>> transpose(array([[1,2],[3,4],[5,6]]))
array([[1,3,5], [2,4,6]])
>>> transpose(array([[1,2,3], [4,5,6]]))
array([[1,4], [2,5], [3,6]])
```

3.2 Matrix Multiplication

The multiplication between two compatible matrices $A = a_{i,j}$ of size $m \times n$ and $B = b_{i,j}$ of size $n \times k$ produces a matrix $C = c_{i,j}$ of size $m \times k$ such that $\forall 0 < i < n$ and $0 < j < k$:

$$ c_{i,j} = \sum_{i=0}^{n} a_{i,k} b_{k,j} $$

Write a function `mat_mult(A: array, B: array) -> array` that returns the multiplication between the two matrices `A` and `B`.

Example

```
>>> from numpy import array
>>> mat_mult(array([[1,2],[4,5]]), array([[1,0], [0,1]]))
array([[1,2], [4,5]])
>>> mat_mult(array([[-1,-2], [-4,-5], [-3, -6]]), array([[1,2,3],[4,5,6]]))
array([[-9,-12,-15], [-24,-33,-42],[-27,-36,-45]])
```

4 Creating Matrices

4.1 Count Of Successive Bases

Consider a DNA sequence, we want to count at which a base pair $b1$ is followed by a base pair $b2$. We will create a matrix where the lines correspond to the base $b1$ and the columns to the base $b2$. As there are 4 bases, `A,T,G,C`, the matrix will be of size 4×4, where we consider the mapping:

- `A` $\rightarrow 0$
- `T` $\rightarrow 1$
- `G` $\rightarrow 2$
- `C` $\rightarrow 3$

Write a function `count_succ(s: str) -> array` that returns the matrix presented above given the string `s`.

Example

```
>>> count_succ("ATTGTGACT")
array([[0,1,0,1], [0,1,2,0], [1,1,0,0], [0,1,0,0]])
>>> count_succ("AAAAATGAGTA")
array([[4,1,1,0], [1,0,1,0], [1,1,0,0], [0,0,0,0]])
```