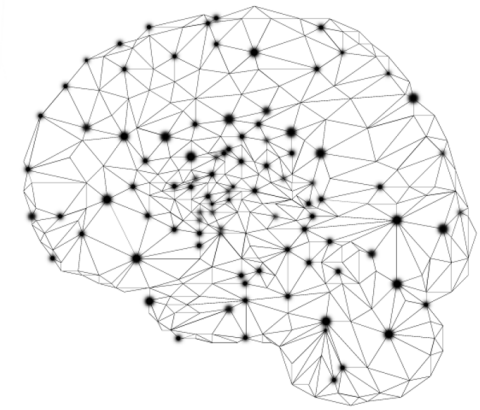

Neural Networks and Data Science

Lab #5

08.11.2023
Deadline: 15.11.2023, 12:10

Dr. Marcel Völschow

HAW HAMBURG

Problem 1

In the lecture part we have introduced stochastic matrices which are a special form of transition matrices used to model various processes. In this task, we will take a deeper look into the political future (and past?) of Neurostan.

a) Explain why the inverse matrix M^{-1} can be used to calculate $\vec{v}_{-1}$, the hypothetical state vector one week before the revolution. Use `np.linalg.inv(M)` to calculate the inverse matrix and $\vec{v}_{-1}$. What's the problem with this state?

b) Make the constructor of `Markov` more robust by including a dimension check of `v0` and `M` using the `len` function

c) Verify that the total number of voters remains constant by writing a method that returns a list of the sum of voters for every timestep.

d) Explain why $\vec{v}_{10}$ can be calculated via $M^{10} \cdot \vec{v}_0$. Write a new method `batchEvolve` that allows you to calculate the next `n` timesteps via a single call.

e) Implement a method that plots the number of voters for every party as a function of time. Calculate the first 100 timesteps and create a plot. Describe your findings.

f) Why does the eigenvector $v_{\lambda=1}$ associated with the eigenvalue $\lambda = 1$ represent the final state of the system v_∞? Use `np.linalg.eig(M)` to calculate $v_{\lambda=1}$.

Problem 2

Python has no built-in datatype to deal with mathematical fractions. In this task, you will implement a new class `Fraction` to fix that issue and carry out basic arithmetical operations by overloading fundamental operators. In Python, this can be done by implementing *magic methods* with a certain name.

a) Create a class `frac` with a constructor `__init__(self,p,q)` that takes two integer arguments `p` and `q` representing the numerator and the denominator. Make sure that the letter is not zero.

b) To print a `frac` object, we need to specify a string representation of it. For that, write a method `__str__(self)` that returns a string `p/q`. Verify that the `print` function now accepts `frac` objects and displays a proper fraction.

c) To use the regular plus operator `+` to add two fractions, write a method `__add__(self,b)` that takes a second `frac` object and returns the sum as a `frac`.

d) After adding two fractions, one usually has to clear up the sum by shortening it, i.e. dividing both numerator and denominator by the greatest common divisor. Write a method `short(self)` that performs that task. Hint: You can use this little code snippet to find the greatest common divisor:

```
def greComDiv(p,q):
    if (p==0):
        return abs(p)
    if (q==0):
        return abs(q)
    while (q!=0):
        h = p % q
        p = q
        q = h
    return abs(p)
```

e) Optional task: If you have too much spare time left, feel free to overload more basic operators using magic methods such as `sub` for subtractions, `mul` for multiplications, `truediv` for divisions or `pow` for the power function. Likewise, you can implement logical operators such as `LT`, `GT`, `LE`, `GE`, `EQ` and `NE` (`<,>,<=,>=,==,!=`).