TI301 – Algorithms and Data Structures 2

Computer Science and Mathematics Project

This project is written in collaboration with the Mathematics Department

Study of Markov Graphs – PART 3

It is now time to look at the properties of these graphs for probabilities.

The concept of distribution

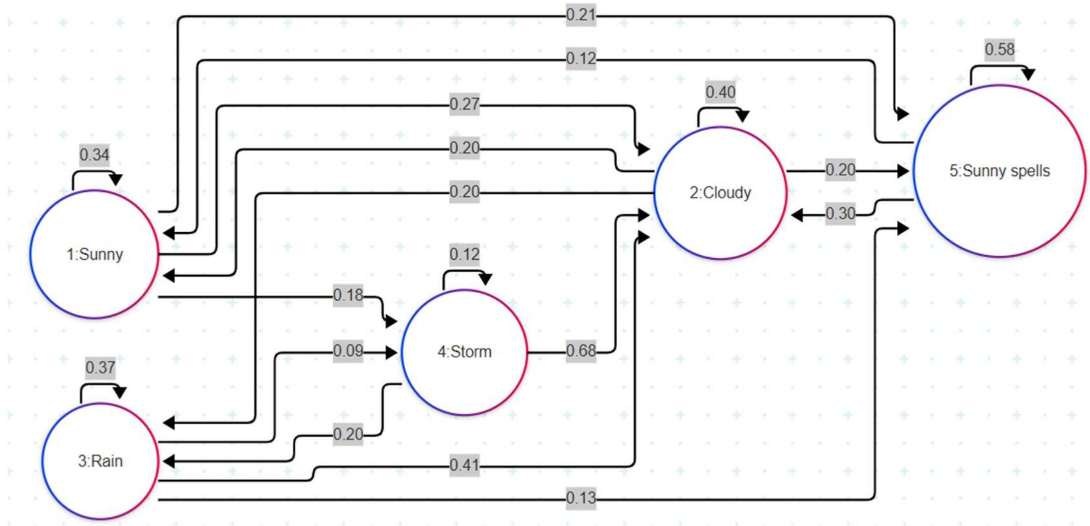
So far, we have associated probabilities with the edges, indicating the probability of moving from one state to another.

At a given moment, a system should be in a state defined according to a 'starting' state. However, as transitions are probabilistic, it is not possible to say that the system will be in 'such and such a state' after 'a certain number' of transitions.

On the other hand, we can indicate the probabilities that the system will be in a given state at a given time t , based on a starting 'state'.

Illustrations and examples with a Markov graph for the weather

Consider the following Markov graph, illustrating (in a completely arbitrary manner) the changes in the weather from day to day: each state represents the weather for one day, and each transition indicates the probability of moving from one state to another. The 'discrete time' used is one day: we 'calculate' the changes from one day D to another day $D+1$, then from $D+1$ to $D+2,$ and so on.



A distribution represents, on this graph, the probabilities of being in a given state at a given moment. A distribution is the distribution of probabilities across all states: its sum must be equal to 1. It is a row vector, where each coordinate indicates the probability of each state of the graph.

The standard notation for a distribution is : Π

Example

Today it is "Sunny": indicates that the probability of being in state 1 is equal to 1, with all others equal to 0.

The associated distribution is then $\Pi = (1 \quad 0 \quad 0 \quad 0 \quad 0)$

Here is a small table to help you interpret this distribution

State number	1	2	3	4	5
Related weather	Sunny	Cloudy	Rain	Storm	Sunny spells
Probability	1	0	0	0	0

Calculating the discrete-time evolution of a distribution

By studying distributions, Markov graphs enable us to answer questions such as:

1. "What is the probability that the weather will be **cloudy** in three days if it is **sunny** today?"
2. "What are the probabilities that the weather will be in such a state in a week (7 days) knowing that it is raining (**Rain**) today?"
3. "Do we reach probabilities independent of the initial distribution after a certain amount of time?" (in mathematical terms, is there a stationary distribution?)

Calculation principles

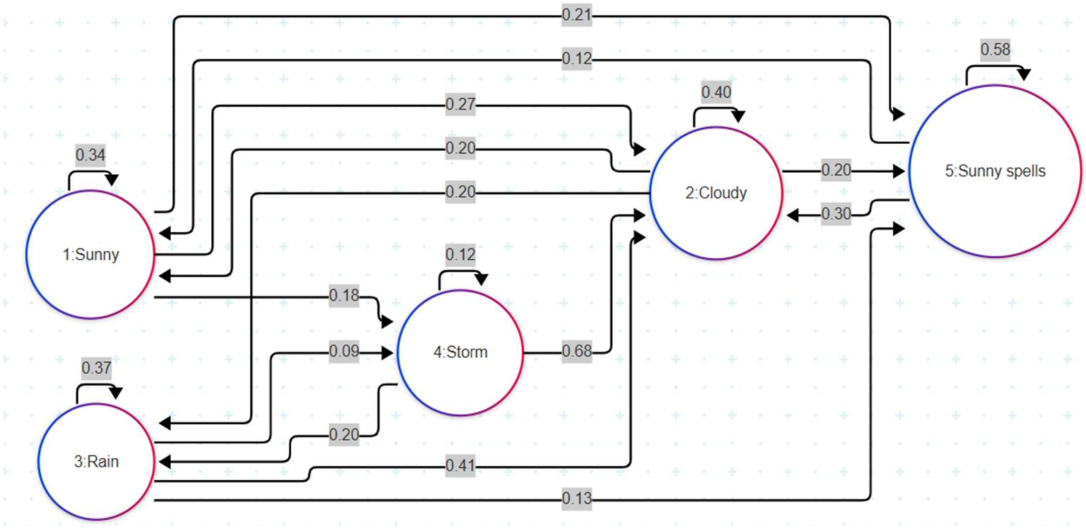
Since the Markov graph is represented by a matrix M , and given an initial distribution Π_0 , we calculate the evolution of the distribution $\Pi_1 = \Pi_0 . M$

We can then start again and calculate: $\Pi_2 = \Pi_1 . M = (\Pi_0 . M) . M = \Pi_0 . M^2$

Step by step, we can therefore calculate the evolution of the initial distribution Π_0 after *n* steps: $\Pi_n = \Pi_0 . M^n$

Illustration with the 'Weather' graph

The matrix M indicates the probabilities of transition from one state to another – here is the matrix representation of the graph used as an example.



$$M = \begin{pmatrix} 0.34 & 0.27 & 0 & 0.18 & 0.21 \\ 0.20 & 0.40 & 0.20 & 0 & 0.20 \\ 0 & 0.41 & 0.37 & 0.09 & 0.13 \\ 0 & 0.68 & 0.20 & 0.12 & 0 \\ 0.12 & 0.30 & 0 & 0 & 0.58 \end{pmatrix}$$

What calculations do we need to perform to answer the questions asked?

1. "What is the probability that the weather will be **cloudy** in 3 days if it is **sunny** today?"
2. "What are the probabilities that the weather will be in a certain state in a week (7 days) given that it is raining (**Rain**) today?"
3. "Do we reach probabilities independent of the initial distribution after a certain amount of time?" (in mathematical terms, is there a stationary distribution?)

<u>For question 1</u>: the initial distribution is $\Pi_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$, we are trying to calculate:

$\Pi_3 = \Pi_0 . M^3$, (3 for 'in 3 days') and we get :

$$M^3 = \begin{pmatrix} 0.17 & 0.37 & 0.13 & 0.05 & 0.27 \\ 0.16 & 0.37 & 0.14 & 0.05 & 0.28 \\ 0.14 & 0.38 & 0.18 & 0.04 & 0.25 \\ 0.15 & 0.38 & 0.18 & 0.05 & 0.24 \\ 0.17 & 0.34 & 0.09 & 0.04 & 0.35 \end{pmatrix}$$

Since $\Pi_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$, $\quad \Pi_3 = (0.17 \quad 0.37 \quad 0.13 \quad 0.05 \quad 0.27)$

Status number	1	2	3	4	5
Related weather	Sunny	Cloudy	Rain	Storm	Sunny spells
Probability after 3 days	0.17	0.37	0.13	0.05	0.27

So, if it is **sunny** today, the probability that it will be **cloudy** in 3 days is 0.37 = 37%.

For question 2: the initial distribution is $\Pi_0 = (0 \quad 0 \quad 1 \quad 0 \quad 0)$, (it is raining, the state is **'Rain'**) we want to calculate : $\Pi_7 = \Pi_0 . M^7$, (7 to calculate 'in 7 days') and we obtain:

$$M^7 = \begin{pmatrix} 0.16 & 0.36 & 0.13 & 0.05 & 0.29 \\ 0.16 & 0.36 & 0.13 & 0.05 & 0.29 \\ 0.16 & 0.36 & 0.13 & 0.05 & 0.29 \\ 0.16 & 0.36 & 0.13 & 0.05 & 0.29 \\ 0.16 & 0.36 & 0.13 & 0.05 & 0.30 \end{pmatrix}$$

Since $\Pi_0 = (0 \quad 0 \quad 1 \quad 0 \quad 0)$, $\Pi_7 = (0.16 \quad 0.36 \quad 0.13 \quad 0.05 \quad 0.29)$

So, if it rains today: **in a week**, there is a 16% chance of sunny weather, a 36% chance of cloudy weather, a 13% chance of rain, a 5% chance of storms, and a 29% chance of sunny spells.

For question 3: we can already see that, in the matrix M^7, all the rows are equal (modulo rounding), and so, regardless of the initial state (= today's weather), we reach the same result:

Regardless of today's weather, in a week's time: there is a 16% chance of sunny weather, a 36% chance of cloudy weather, a 13% chance of rain, a 5% chance of storms, and a 29% chance of sunny spells. This is also due to the very simplified representation of weather patterns.

This distribution, called $\Pi^* = (0.16 \quad 0.36 \quad 0.13 \quad 0.05 \quad 0.29)$, is said to be stationary. There will be no further change in probabilities, so we have: $\Pi^* . M = \Pi^*$

Now it's your turn

You can use AI tools for this part, but you will need to explain:

- ✓ **If you used such tools;**
- ✓ **The prompt you have used;**
- ✓ **The code obtained.**

These questions will be asked during tour project presentation

Step 1: Matrix calculations

Using the **`matrix.c`** / **`matrix.h`** files, define the following functions:

- ✓ A function that, from an adjacency list for a graph with n states, creates a $n\ x\ n$ matrix filled with the transition probabilities between states;
- ✓ A function that creates a $n\ x\ n$ matrix filled with the value 0;
- ✓ A function that copies the values from one matrix to another of the same size;
- ✓ A function for multiplying two $n\ x\ n$ matrices;
- ✓ A function that calculates the 'difference' between two matrices M and N : $diff(M,N) = \sum_i \sum_j |m_{ij} - n_{ij}|$ – sum of the absolute values of the differences between the coefficients of the matrices.

Validation of step 1

- ✓ Display of the matrix M associated with the weather example (**`exemple_meteo.txt`**);
- ✓ Calculate M^3 , you should obtain the same result as the one shown in the example;
- ✓ Calculate M^7 ; you should obtain the same result as the one shown in the example.
- ✓ Using the example files: calculate M^n , for which the difference between M^n and M^{n-1} is less than $\varepsilon = 0.01$. (Please note that this criterion may not work for some examples: indicate which examples these are).

Step 2: Properties of Markov graphs

Markov graphs have the following properties (without proof)

- ✓ An irreducible graph (a single class) has a unique limiting distribution

- ✓ For a non-irreducible graph (multiple classes):
  - **Transient** classes have a zero limit distribution (all probabilities are equal to 0);
  - **Persistent** classes each have a limiting distribution;

In the **matrix.c** / **matrix.h** files, add the following function (using the elements from part 2):

```
/**
 * @brief Extracts a submatrix corresponding to a specific
component of a graph partition.
 *
 * @param matrix The original adjacency matrix of the graph.
 * @param part The partition of the graph into strongly
connected components.
 * @param compo_index The index of the component to extract.
 * @return t_matrix The submatrix corresponding to the
specified component.
 */
t_matrix subMatrix(t_matrix matrix, t_partition part, int
compo_index);
```

In this 'submatrix', only the rows and columns of the vertices belonging to a given component are kept.

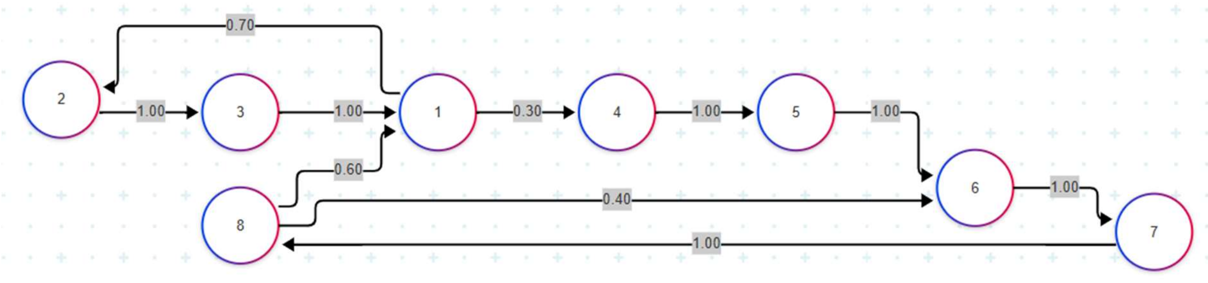
This will allow you to create a 'submatrix' for a given class: by calculating the powers of these submatrices, you will obtain the stationary distributions by class.

Validation of step 2

You obtain the **stationary** distributions **for each of the classes in a graph** (test on the sample files provided).

Step 3 (bonus challenge)

Periodicity of classes: some classes do not have a limit distribution, but **several** stationary (and periodic) distributions, because the probabilities evolve in a 'cyclical' manner. Here is an example of a periodic graph:



This graph is irreducible (a single class – check it with your programme) but has a 'period'. Starting from any vertex, you can return to it after 3 steps (but not after 1 or 2 steps), or after 6 steps.

So here is the challenge – I am providing you with the raw code for calculating the period for a class (code to be adapted according to your data structures and the functions you have written).

```
int gcd(int *vals, int nbvals) {
    if (nbvals == 0) return 0;
    int result = vals[0];
    for (int i = 1; i < nbvals; i++) {
        int a = result;
        int b = vals[i];
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        result = a;
    }
    return result;
}

int getPeriod(t_matrix sub_matrix)
{
    int n = sub_matrix.rows;
    int *periods = (int *)malloc(n * sizeof(int));
    int period_count = 0;
    int cpt = 1;
    t_matrix power_matrix = createEmptyMatrix(n);
    t_matrix result_matrix = createEmptyMatrix(n);
    copyMatrix(power_matrix, sub_matrix);

    for (cpt = 1; cpt <= n; cpt++)
    {
        int diag_nonzero = 0;
        for (int i = 0; i < n; i++)
        {
            if (power_matrix.data[i][i] > 0.0f)
            {
                diag_nonzero = 1;
            }
        }
        if (diag_nonzero) {
            periods[period_count] = cpt;
            period_count++;
        }
        multiplyMatrices(power_matrix, sub_matrix, result_matrix);
        copyMatrix(power_matrix, result_matrix);
    }

    return gcd(periods, period_count);
}
```

Challenge number 1

Comment on and explain this code, then integrate it into your programme.

Challenge number 2

Calculate the periods of the classes in the graph (the period will then be the same for all vertices belonging to that class), then find the associated stationary distributions.