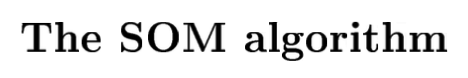
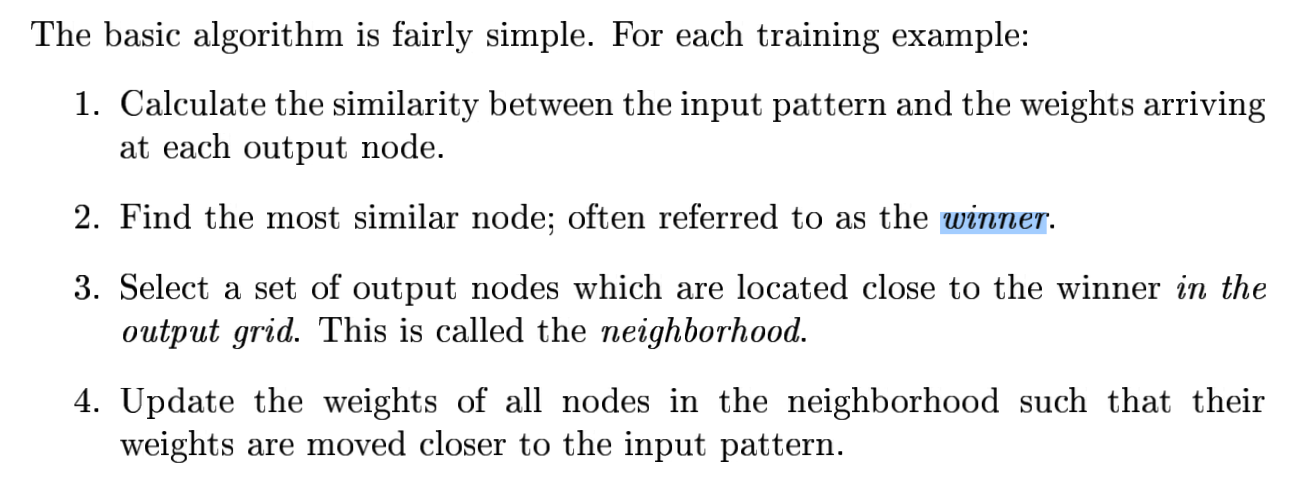
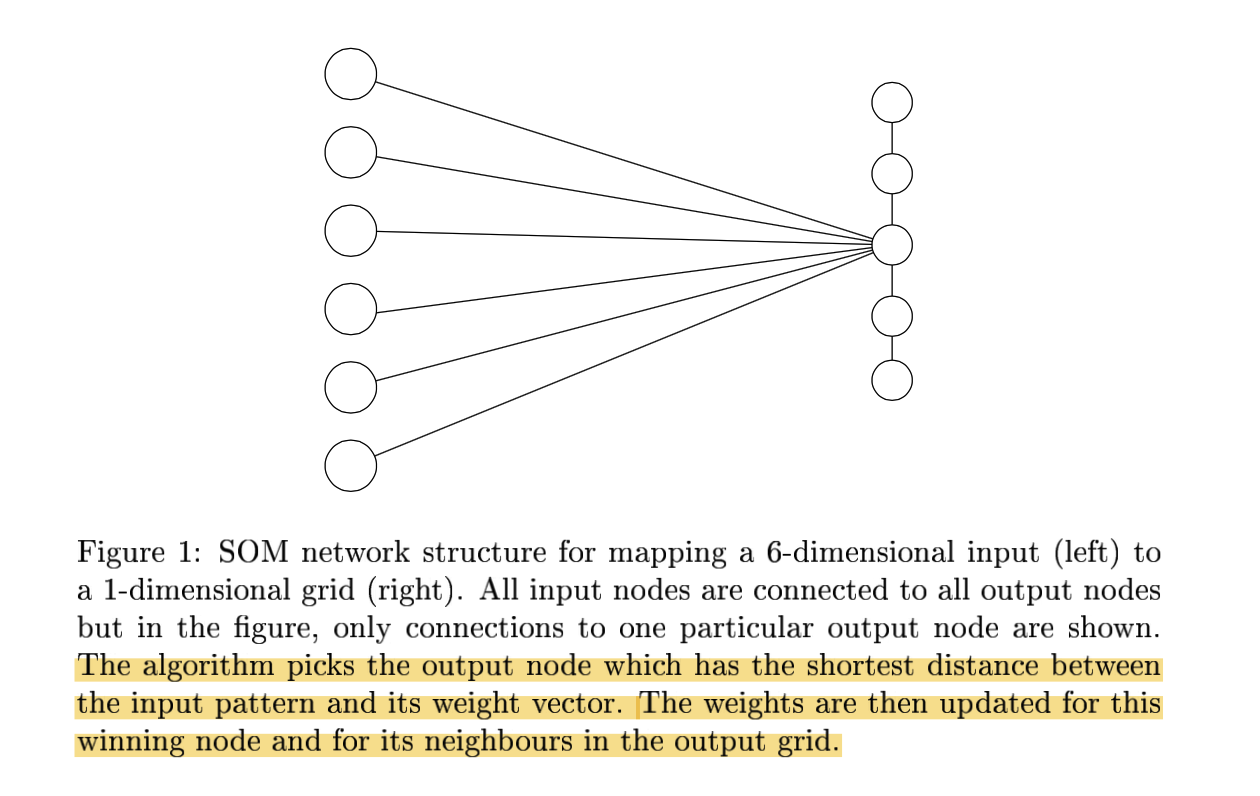
**Lab3 Self Organizing Map**

The algorithm is supposed to find a low-dimensional representation that preserves topology of the higher-dimensional data. Topology preservation means that points which are close in the input space should also be close in the output space.





**SOM network of Task1 & Task 2**

****

**Task1** 

**Results:**

***'dragonfly'***

***'grasshopper'***

***'beetle'***

***'butterfly'***

***'housefly'***

***'moskito'***

***'spider'***

'pelican'

'duck'

'penguin'

'ostrich'

'frog'

'crocodile'

'seaturtle'

'walrus'

'bear'

'hyena'

'dog'

'rat'

'bat'

'skunk'

'cat'

'lion'

'ape'

'elephant'

'rabbit'

'kangaroo'

'antelop'

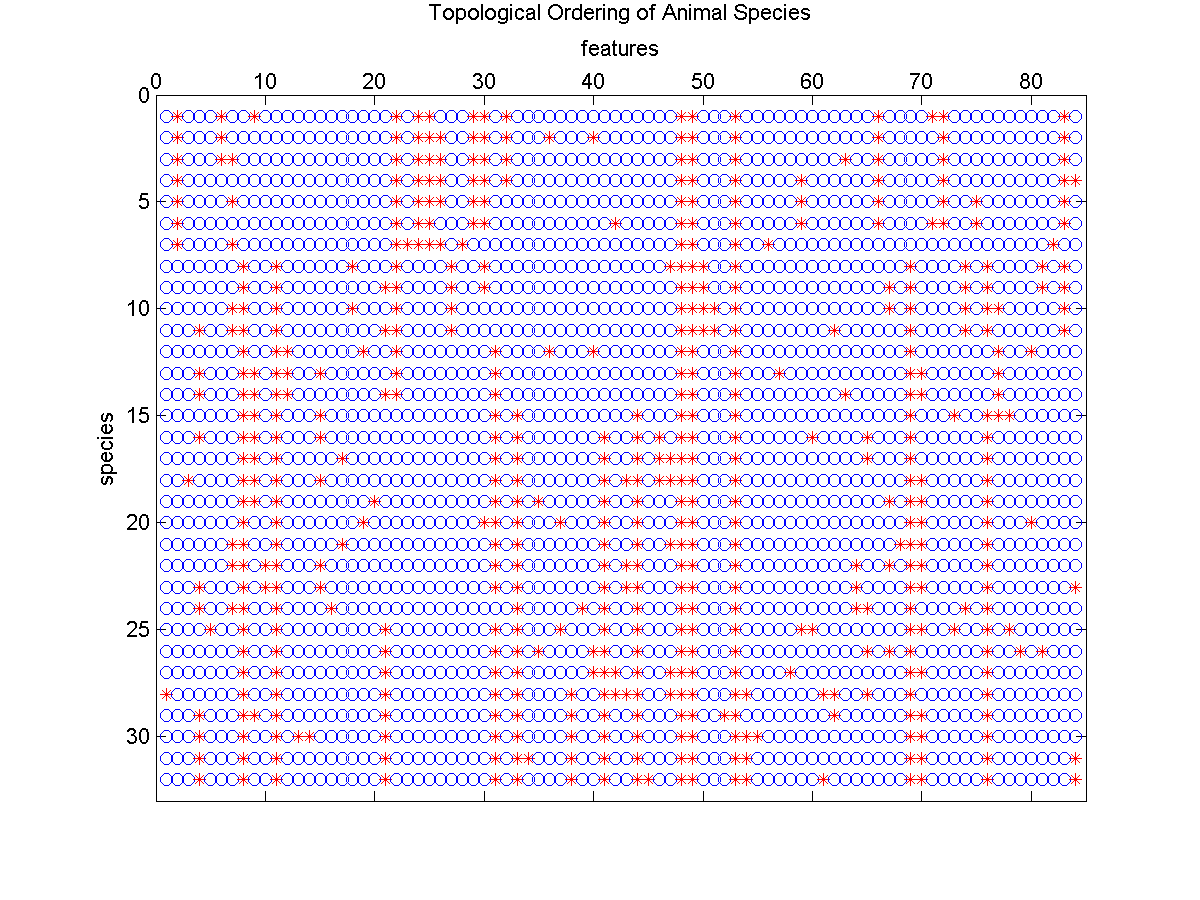
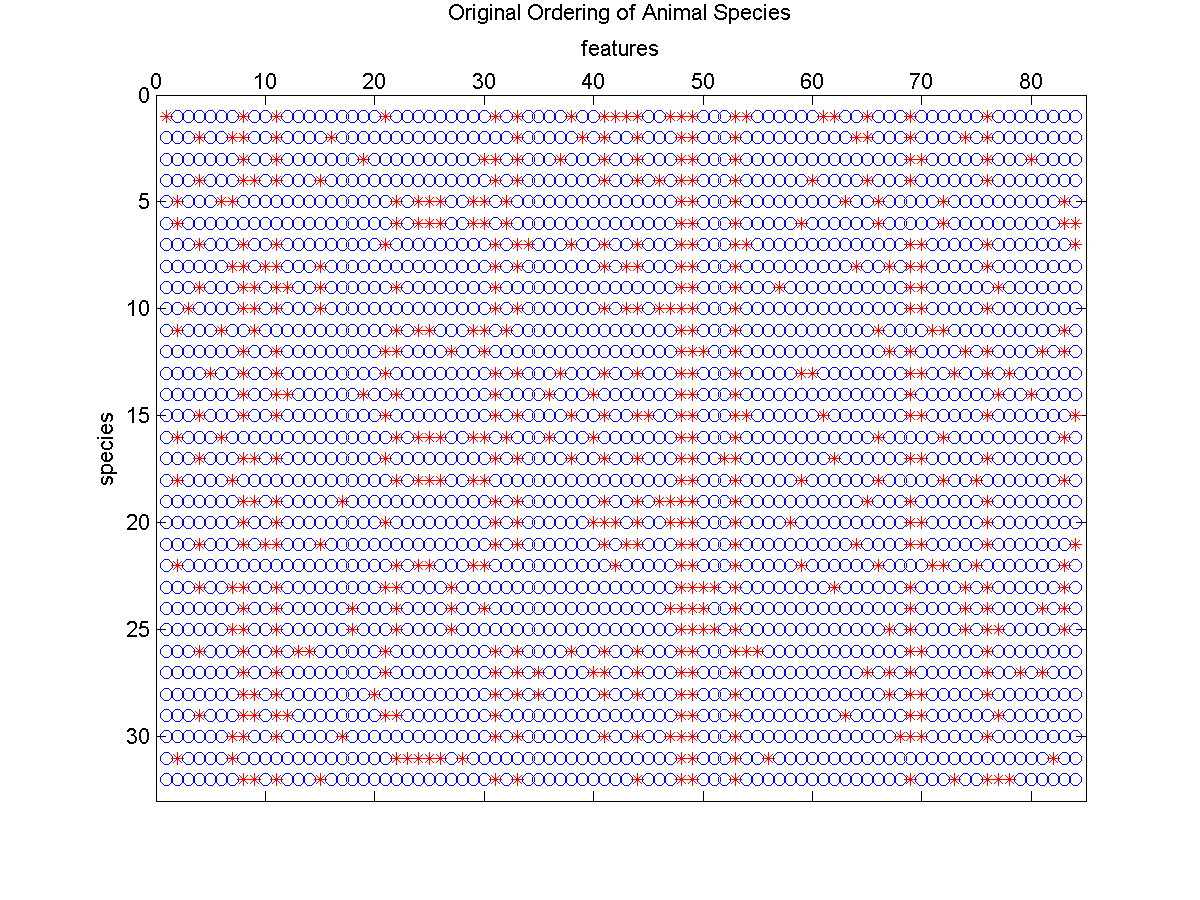
'horse'

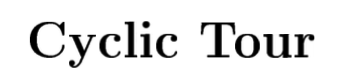
'pig'

'camel'

'giraffe'

Animals next to each other in the listing have some similarity between them. ***Insects*** should typically be grouped together, separate from the different cats, for example.



**Task2** 

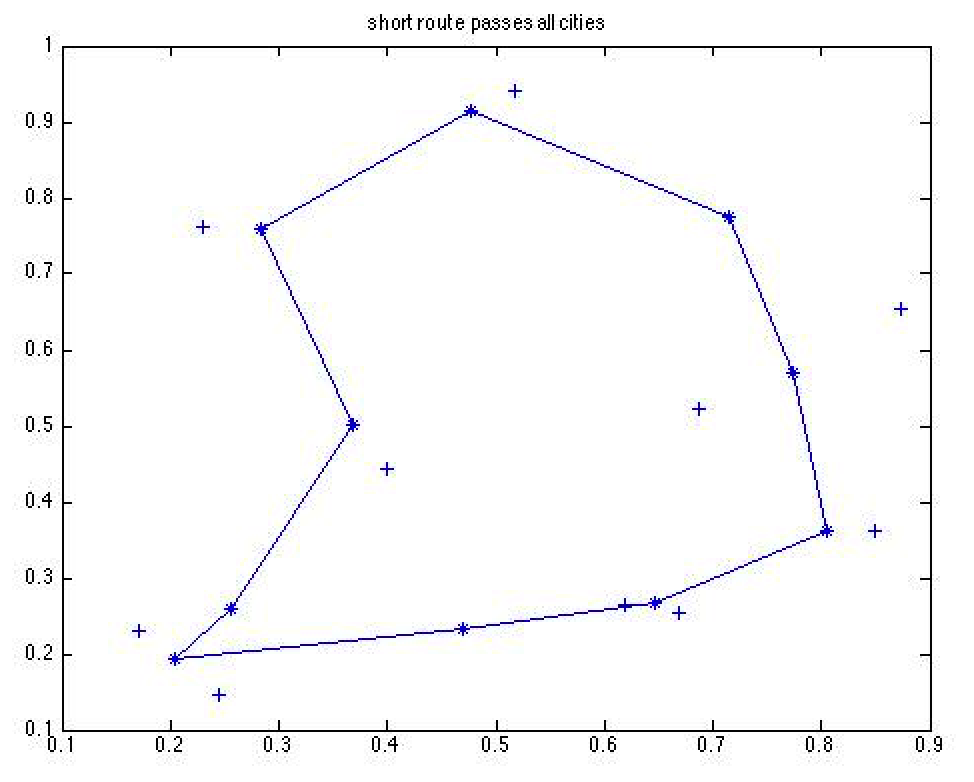
**Difference with “Topological Ordering of Animal Species”:**

The neighbourhood should be circular since we are looking for a circular tour. When calculating the neighbours you have to make sure that the first and the last output node are treated as next neighbours.

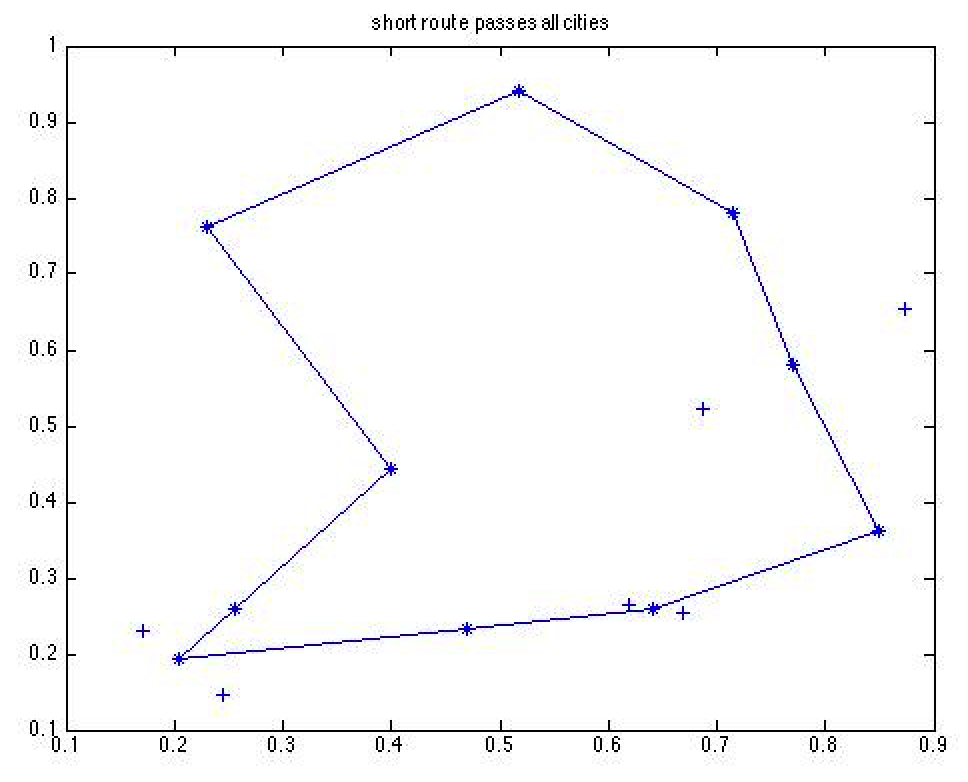
**Results:**

With some luck, the SOM algorithm will be able to find a fairly short route which passes all cities.

20 epochs

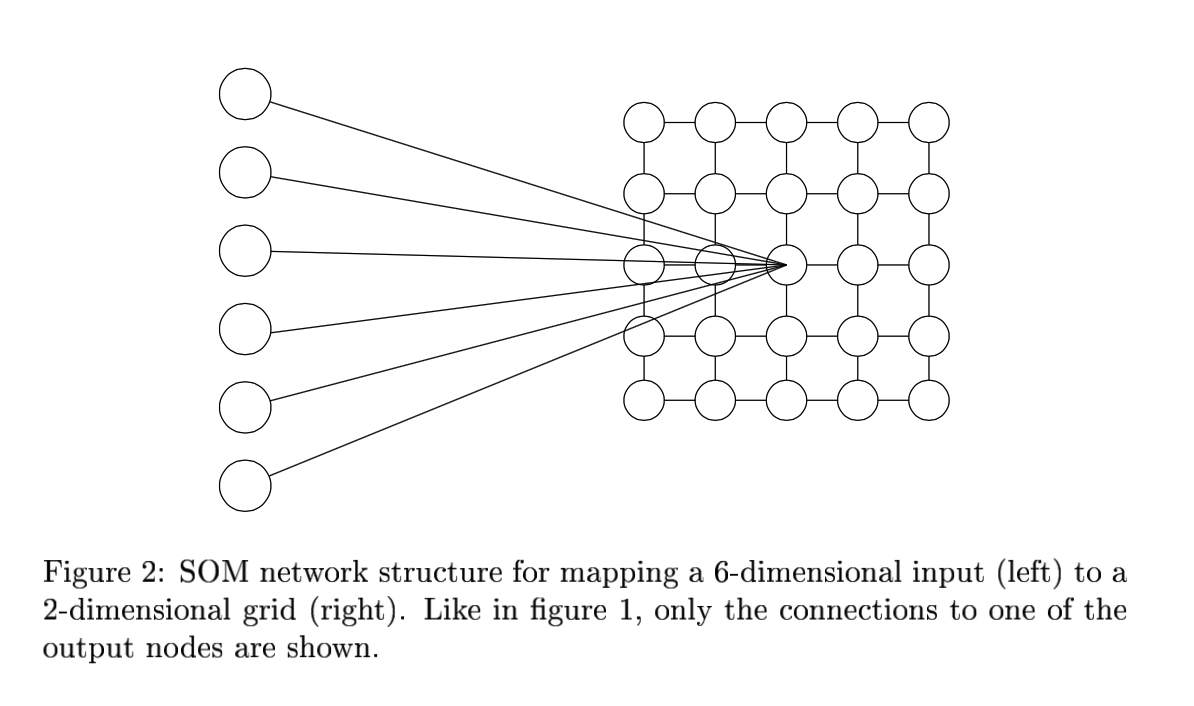


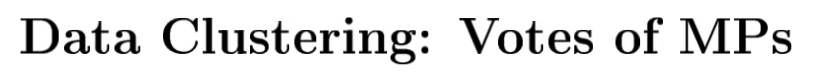
5000 epochs



After large amount of epochs, some weights end up to exactly the input nodes.

**SOM network of Task3**



**Task3** 

You should use the SOM algorithm to find a topological mapping from the 31-dimensional input space to a 10 × 10 output grid.

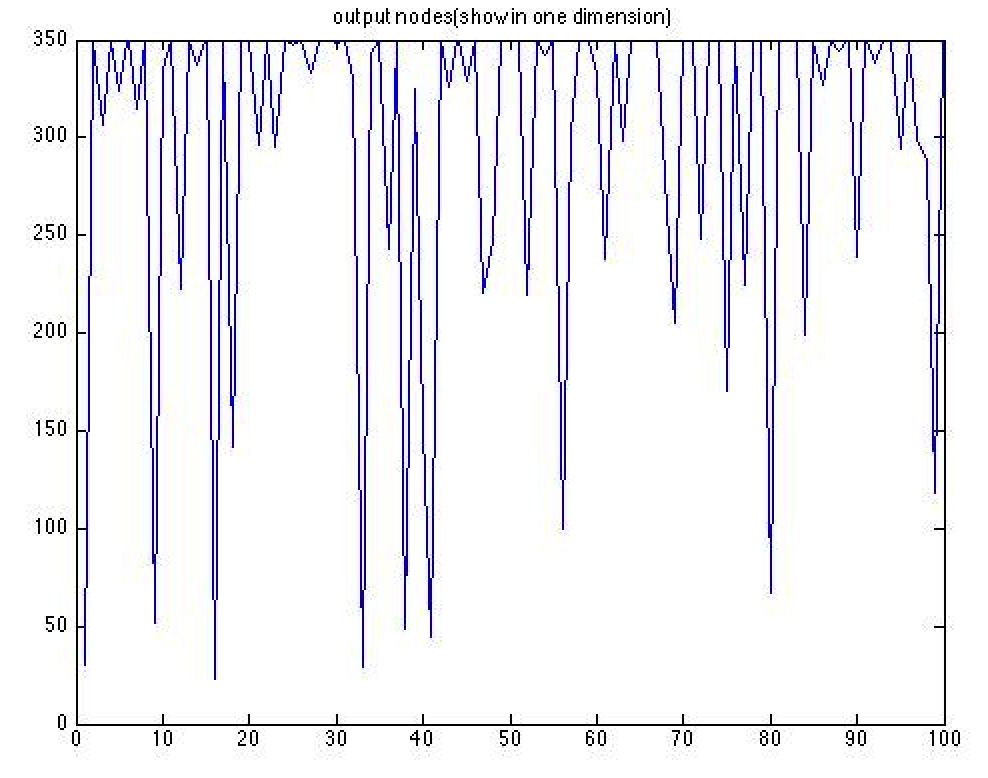
**Difference with Task 1 & Task2:**

1. Output space is a 2D grid map
2. Update of neighbor is using 4-nearest neighbours.

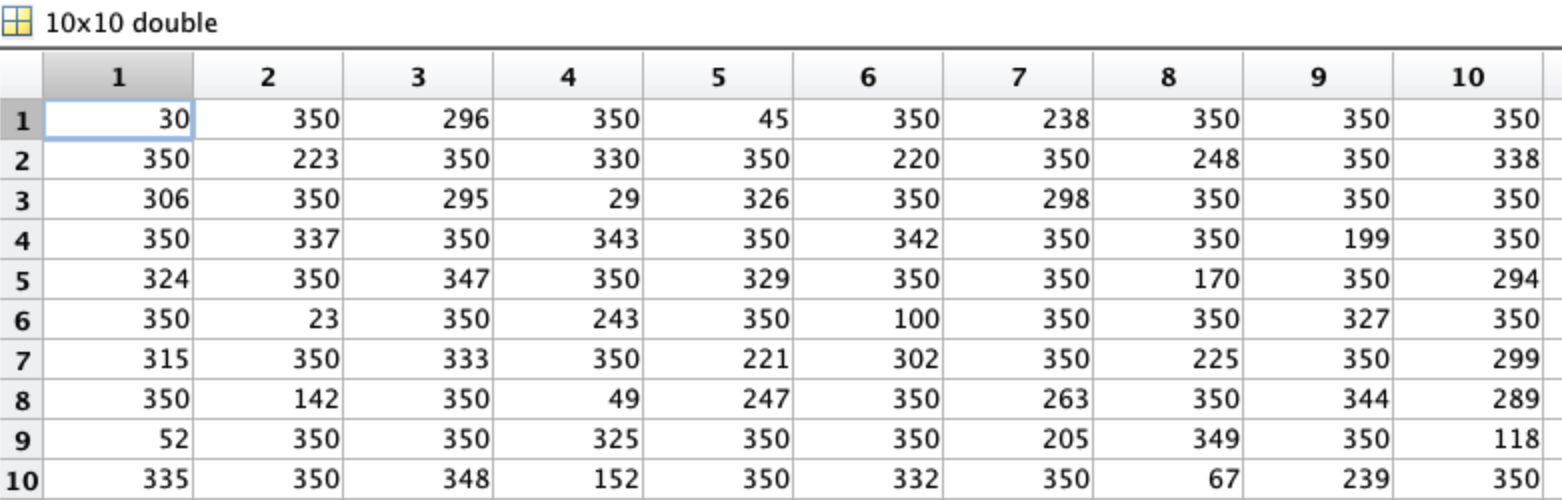
**Results:**

Output is stored in a (a 1by100 vector, which represent 100 nodes). Several MPs may end up at the same output node and the variable a will simply keep the index of the last one.

Plot this 1-D vector:

****

**Show the resulting 2D grid map:**

****

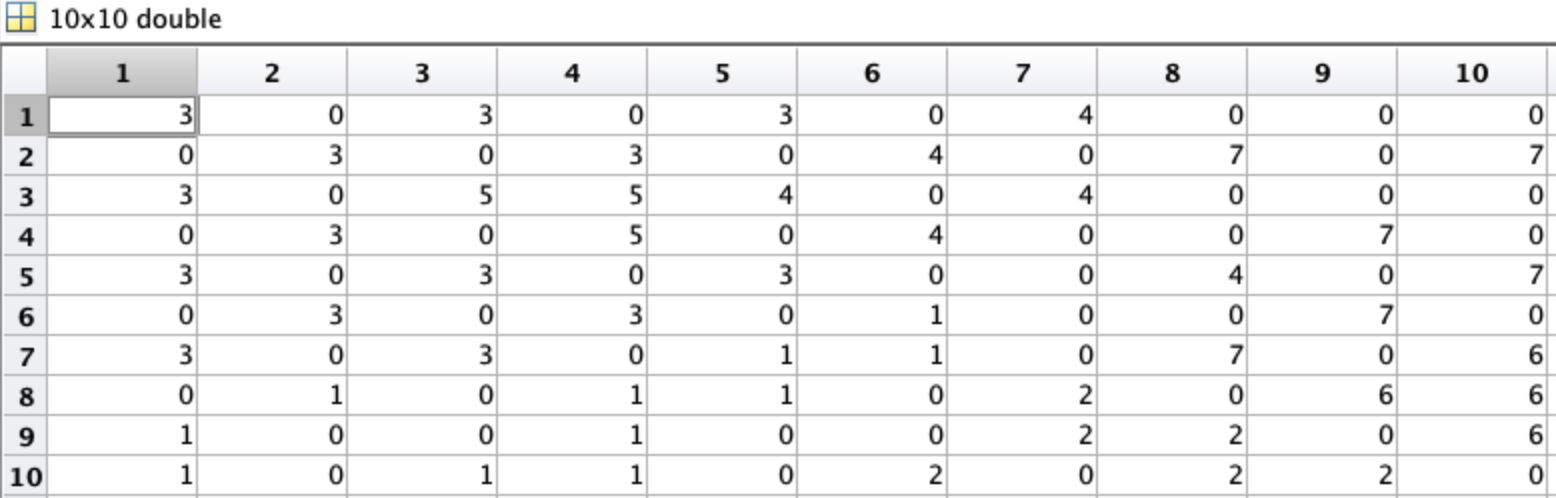
**Most of the index numbers are 350. They mean nodes which never win.**

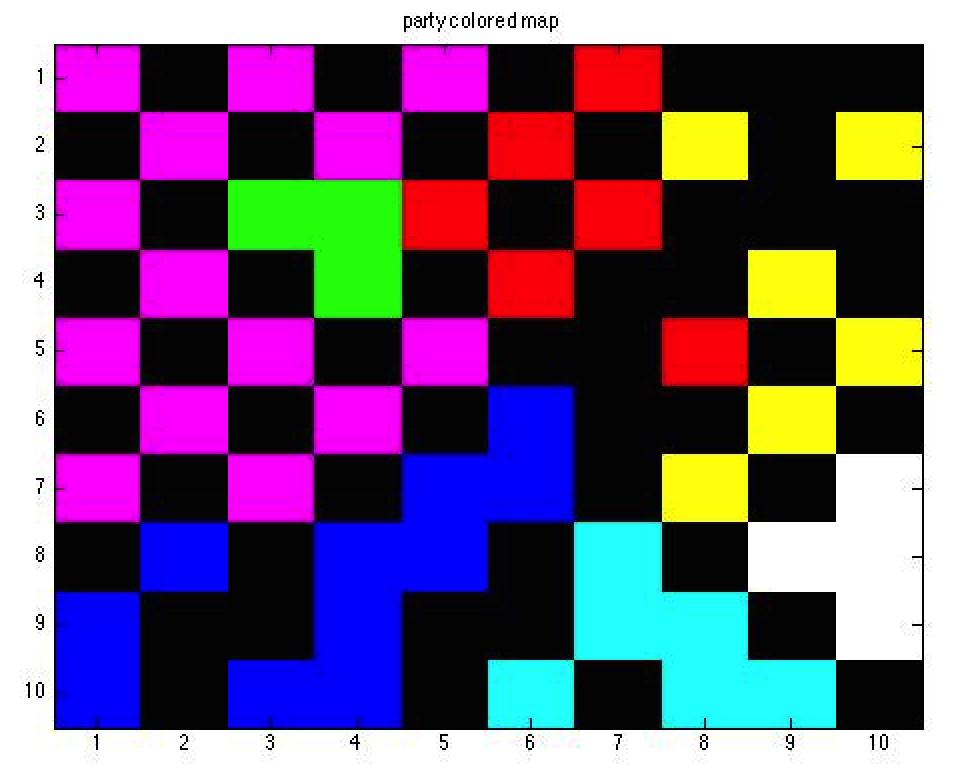
**In our algorithm, we only use 4-nearest neighbourhood.**

1. **Different parties end up in the map**

Coding: 0=no party, 1='m', 2='fp', 3='s', 4='v', 5='mp', 6='kd', 7='c'

The resulting 2D grid map:



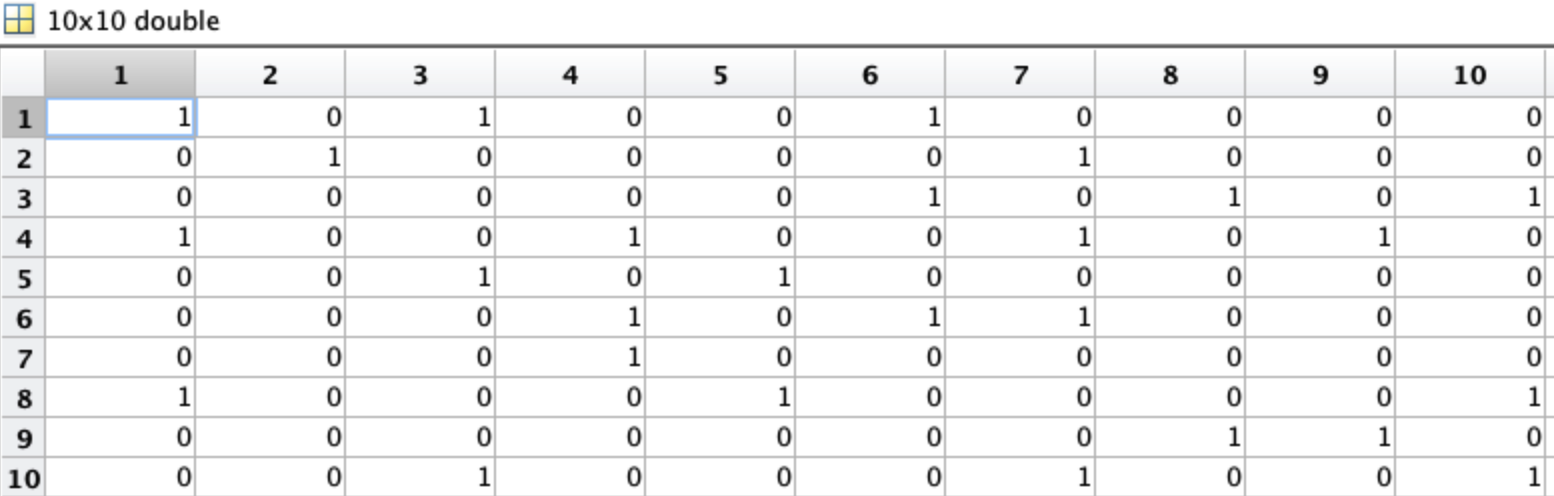


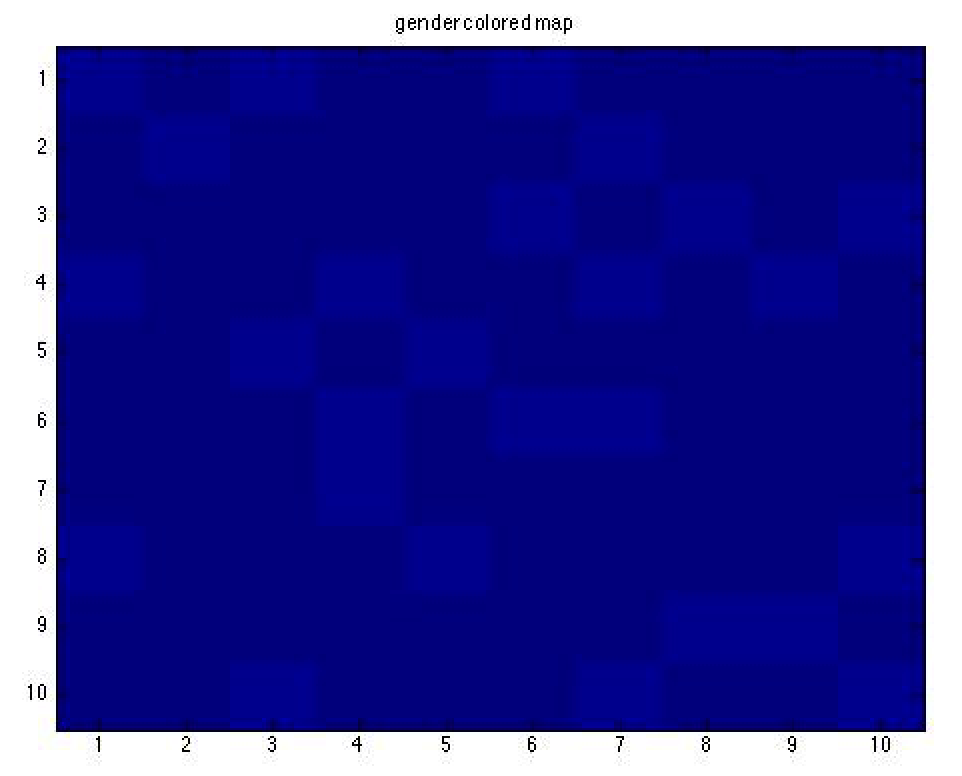
There are some different color clusters, which show the different voting area for different parties. The pink and blue ones have more votes compared with other ones.

1. **Different genders end up in the map**

Coding: Male 0, Female 1

The resulting 2D grid map:



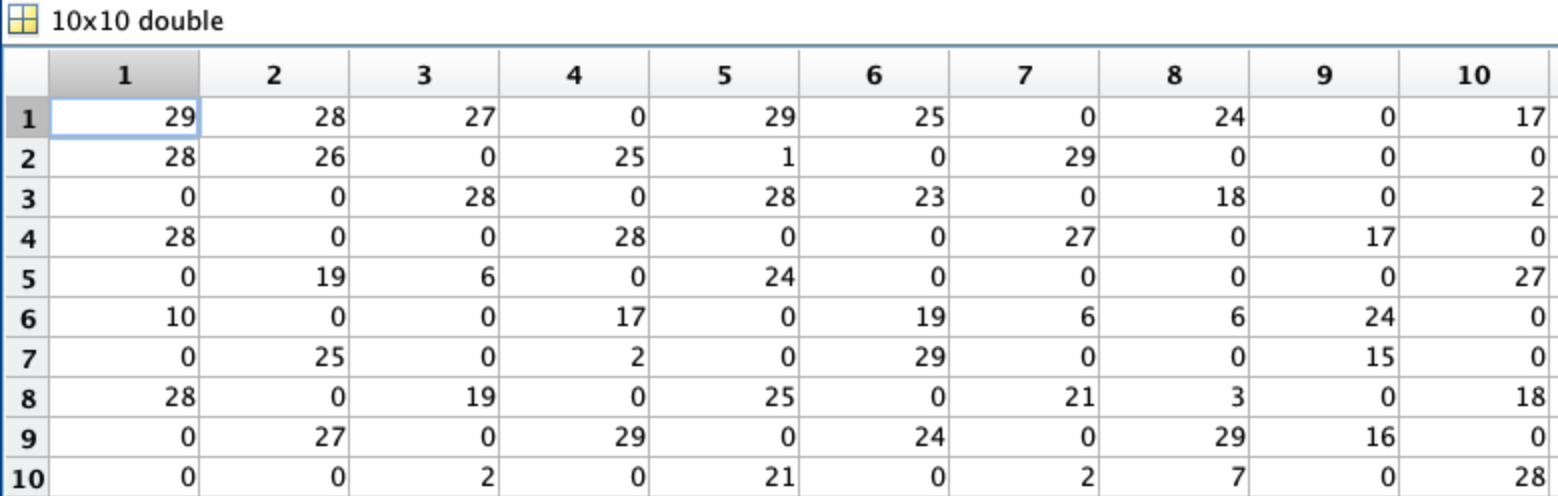


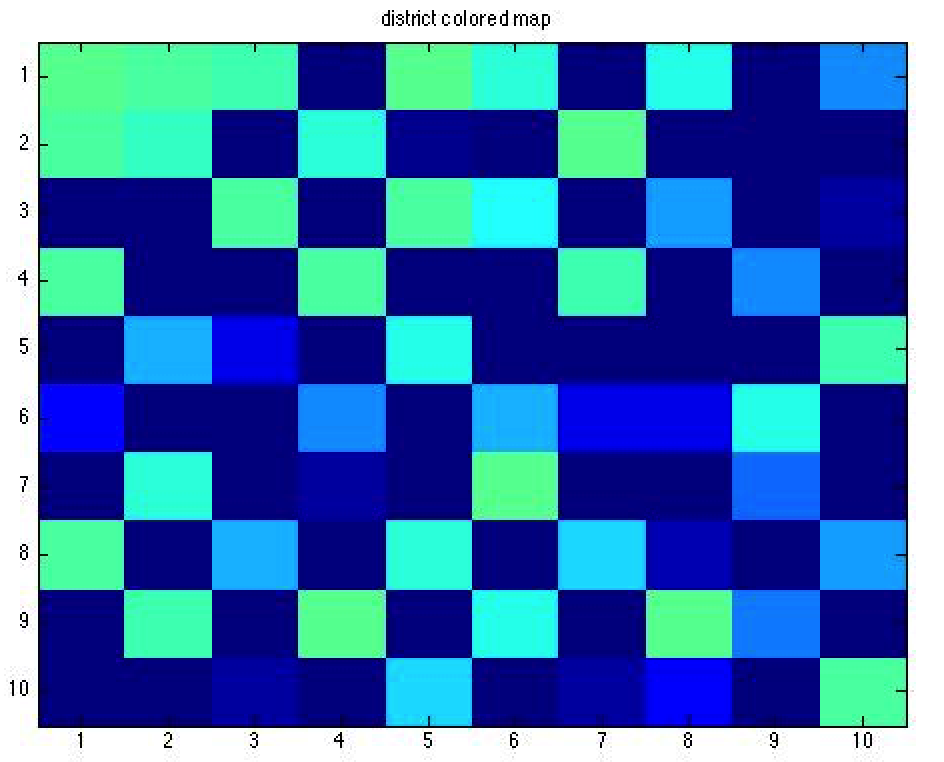
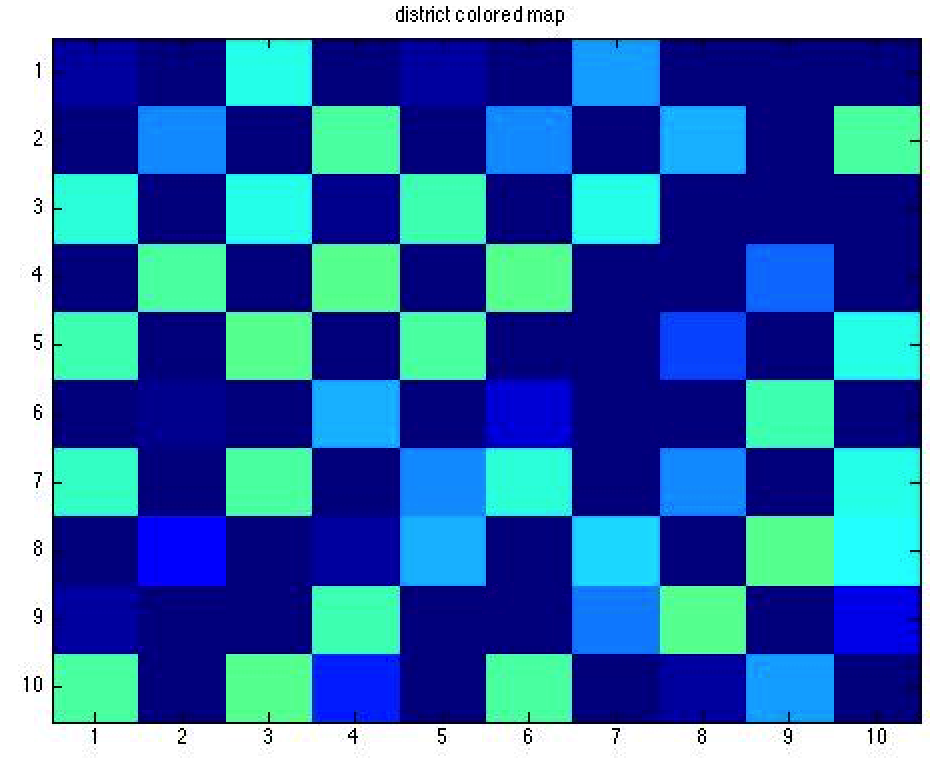
The lighter blue (white) means Female while the darker blue (black) means Male.

By looking at the distribution of female and male MPs, we can see that there is no obvious tendency that MPs tend to vote differently depending on their gender.

1. **Different districts end up in the map**

The resulting 2D grid map:





The matrix tells that different index numbers of districts are distributed evenly, i.e. there is no obvious cluster.

Comparing two results of two 2D grid maps of different districts, we see that different color areas in both images are distributed evenly.

Therefore, we conclude that there is no obvious tendency for MPs from different districts to vote systematically different.