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.

The SOM algorithm

The basic algorithm is fairly simple. For each training example:

1. Calculate the similarity between the input pattern and the weights arriving at each output node.
2. Find the most similar node; often referred to as the *winner*.
3. Select a set of output nodes which are located close to the winner *in the output grid*. This is called the *neighborhood*.
4. Update the weights of all nodes in the neighborhood such that their weights are moved closer to the input pattern.

SOM network of Task1 & Task 2

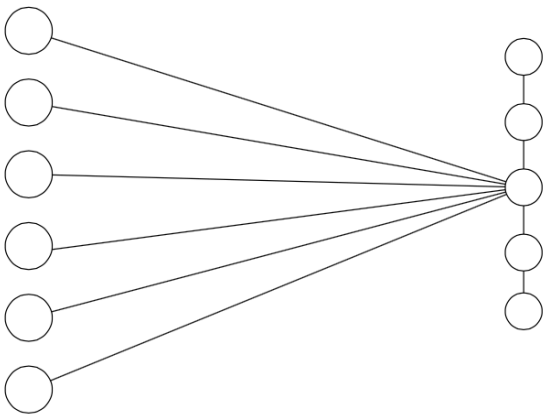


Figure 1: SOM network structure for mapping a 6-dimensional input (left) to a 1-dimensional grid (right). All input nodes are connected to all output nodes but in the figure, only connections to one particular output node are shown. The algorithm picks the output node which has the shortest distance between the input pattern and its weight vector. The weights are then updated for this winning node and for its neighbours in the output grid.

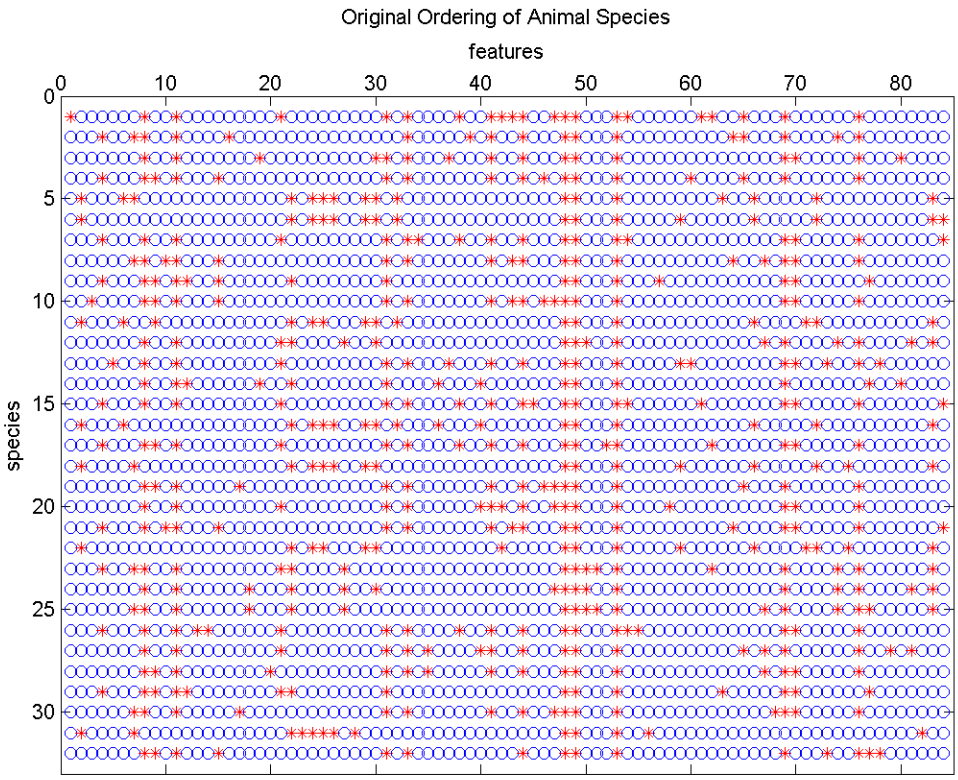
Task1 Topological Ordering of Animal Species

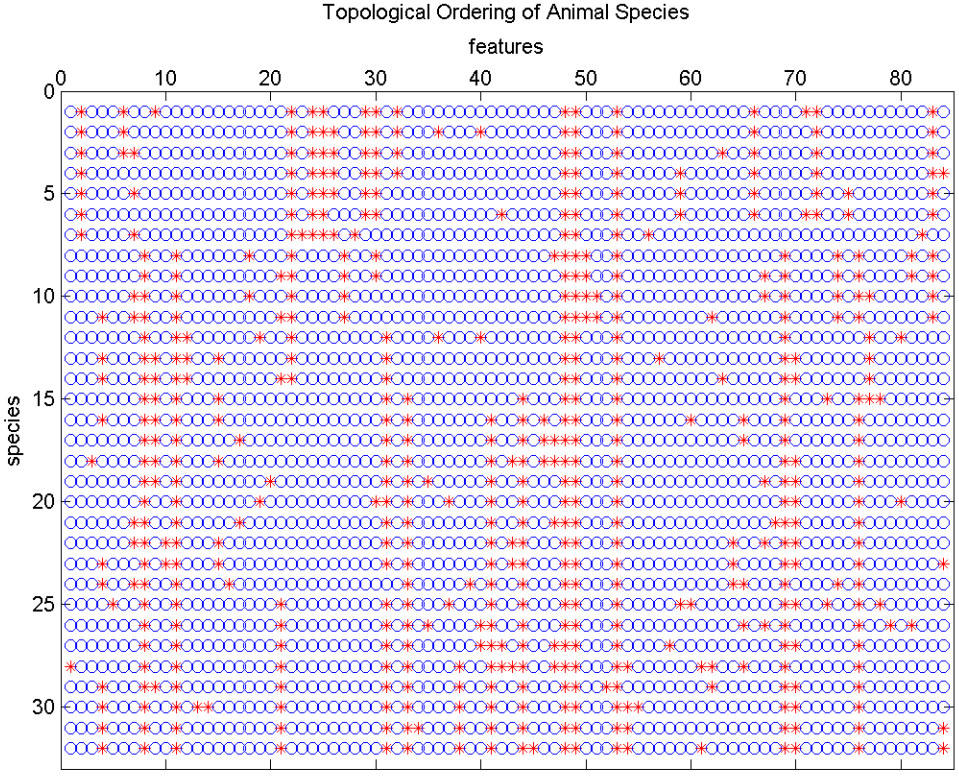
Results:

'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.


Original Ordering of Animal Species
features
species
0
10
20
30
40
50
60
70
80
5
15
25


Topological Ordering of Animal Species
features
species
0
10
20
30
40
50
60
70
80
5
15
25

Task2 Cyclic Tour

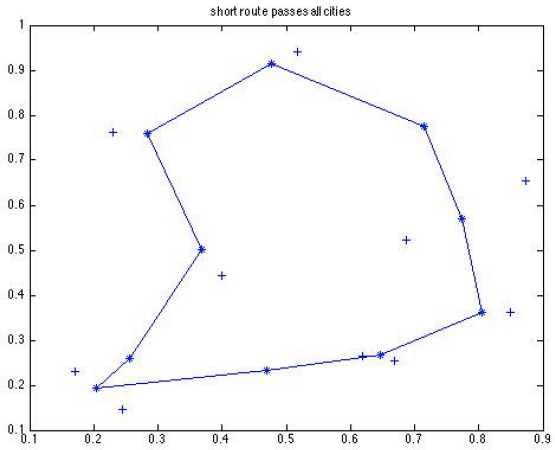
Difference with "Topological Ordering of Animal Species":

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.

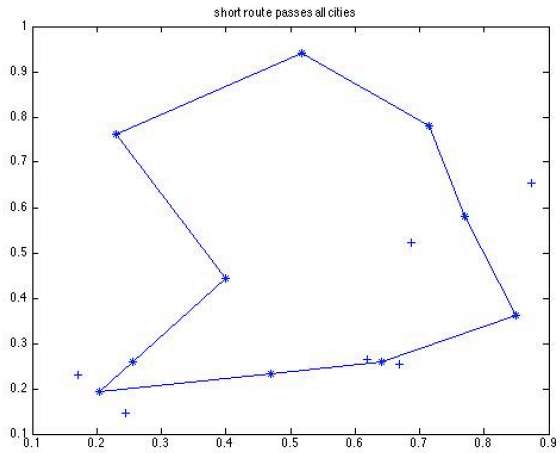
Results:

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

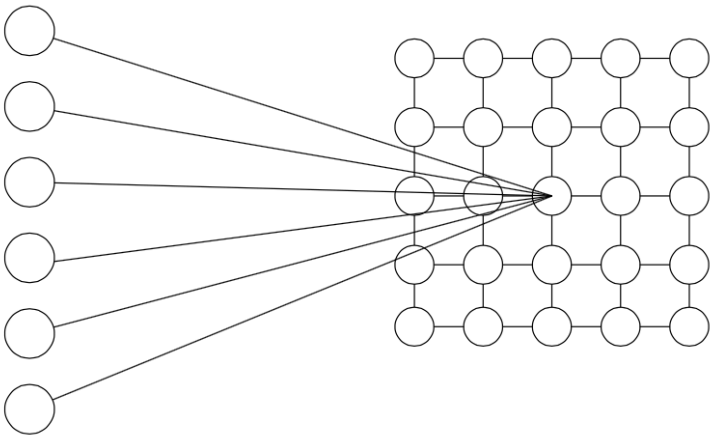


Figure 2: SOM network structure for mapping a 6-dimensional input (left) to a 2-dimensional grid (right). Like in figure 1, only the connections to one of the output nodes are shown.

Task3 Data Clustering: Votes of MPs

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

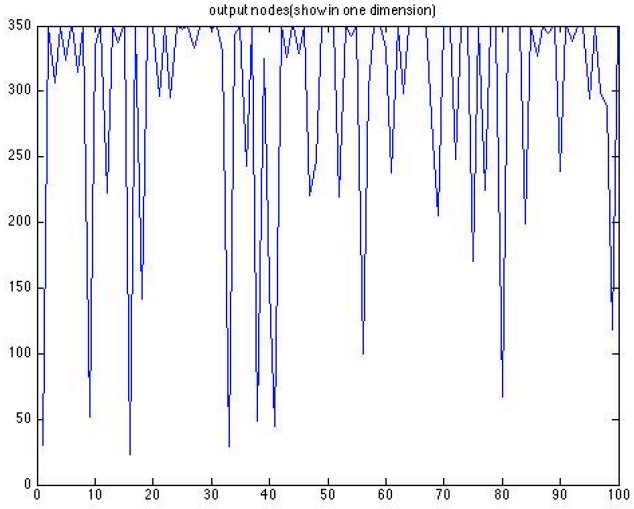
Difference with Task 1 & Task2:

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:

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	30	350	296	350	45	350	238	350	350	350
2	350	223	350	330	350	220	350	248	350	338
3	306	350	295	29	326	350	298	350	350	350
4	350	337	350	343	350	342	350	350	199	350
5	324	350	347	350	329	350	350	170	350	294
6	350	23	350	243	350	100	350	350	327	350
7	315	350	333	350	221	302	350	225	350	299
8	350	142	350	49	247	350	263	350	344	289
9	52	350	350	325	350	350	205	349	350	118
10	335	350	348	152	350	332	350	67	239	350

Most of the index numbers are 350. They mean nodes which never win. In our algorithm, we only use 4-nearest neighbourhood.

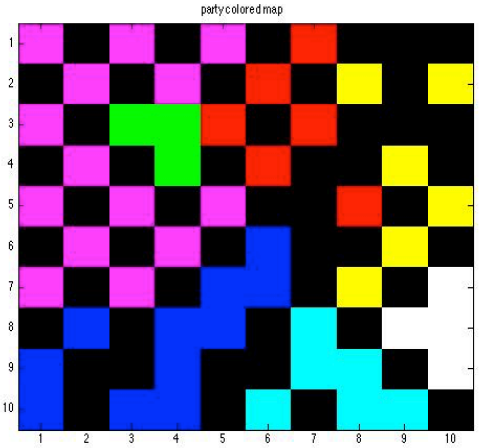
1) Different parties end up in the map

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

The resulting 2D grid map:

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	3	0	3	0	3	0	4	0	0	0
2	0	3	0	3	0	4	0	7	0	7
3	3	0	5	5	4	0	4	0	0	0
4	0	3	0	5	0	4	0	0	7	0
5	3	0	3	0	3	0	0	4	0	7
6	0	3	0	3	0	1	0	0	7	0
7	3	0	3	0	1	1	0	7	0	6
8	0	1	0	1	1	0	2	0	6	6
9	1	0	0	1	0	0	2	2	0	6
10	1	0	1	1	0	2	0	2	2	0



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.

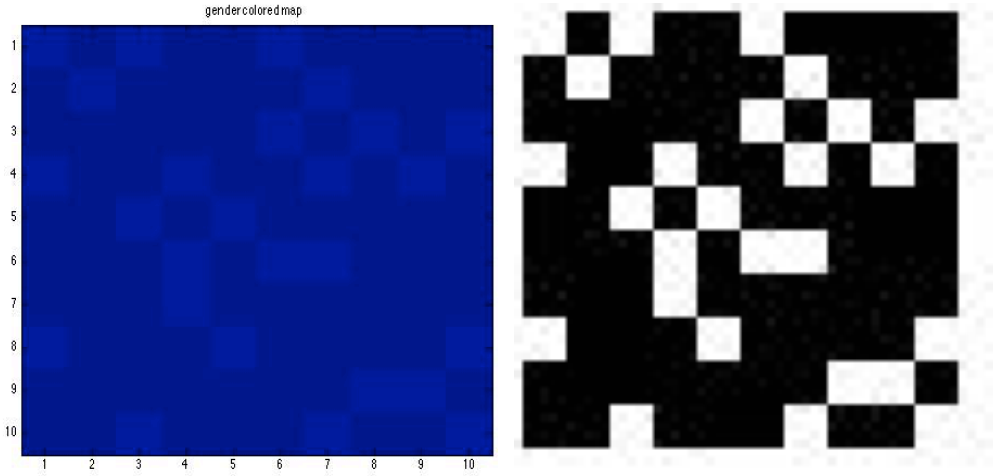
2) Different genders end up in the map

Coding: Male 0, Female 1

The resulting 2D grid map:

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	1	0	1	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1	0	0	0
3	0	0	0	0	0	1	0	1	0	1
4	1	0	0	1	0	0	1	0	1	0
5	0	0	1	0	1	0	0	0	0	0
6	0	0	0	1	0	1	1	0	0	0
7	0	0	0	1	0	0	0	0	0	0
8	1	0	0	0	1	0	0	0	0	1
9	0	0	0	0	0	0	0	1	1	0
10	0	0	1	0	0	0	1	0	0	1



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

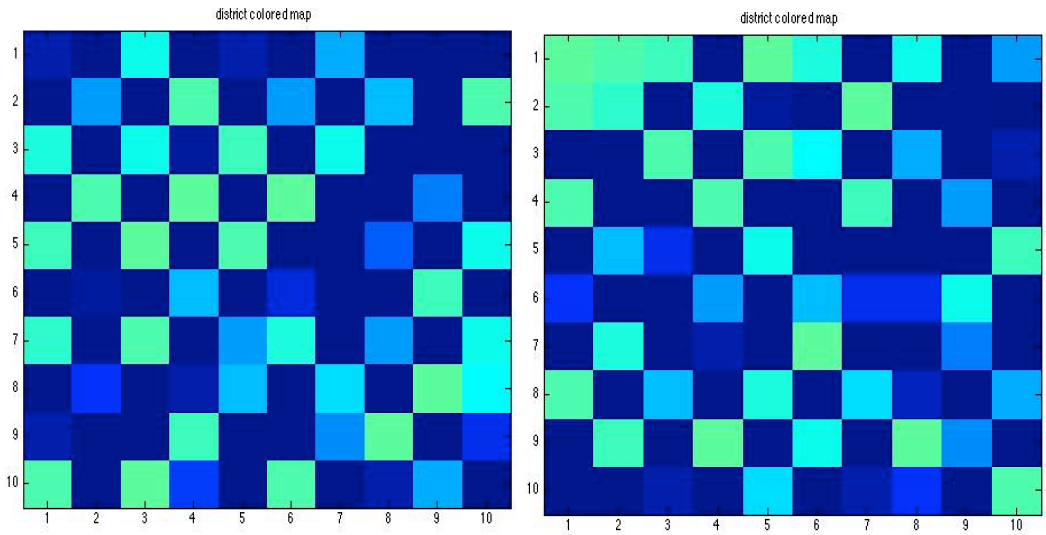
By looking at the distribution of female and male MPs, we can see that there is no obvious tendency that MPs tend to vote differently depending on their gender.

3) Different districts end up in the map

The resulting 2D grid map:

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	29	28	27	0	29	25	0	24	0	17
2	28	26	0	25	1	0	29	0	0	0
3	0	0	28	0	28	23	0	18	0	2
4	28	0	0	28	0	0	27	0	17	0
5	0	19	6	0	24	0	0	0	0	27
6	10	0	0	17	0	19	6	6	24	0
7	0	25	0	2	0	29	0	0	15	0
8	28	0	19	0	25	0	21	3	0	18
9	0	27	0	29	0	24	0	29	16	0
10	0	0	2	0	21	0	2	7	0	28



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.