Mahmoud Ahmed Mohamed Ali

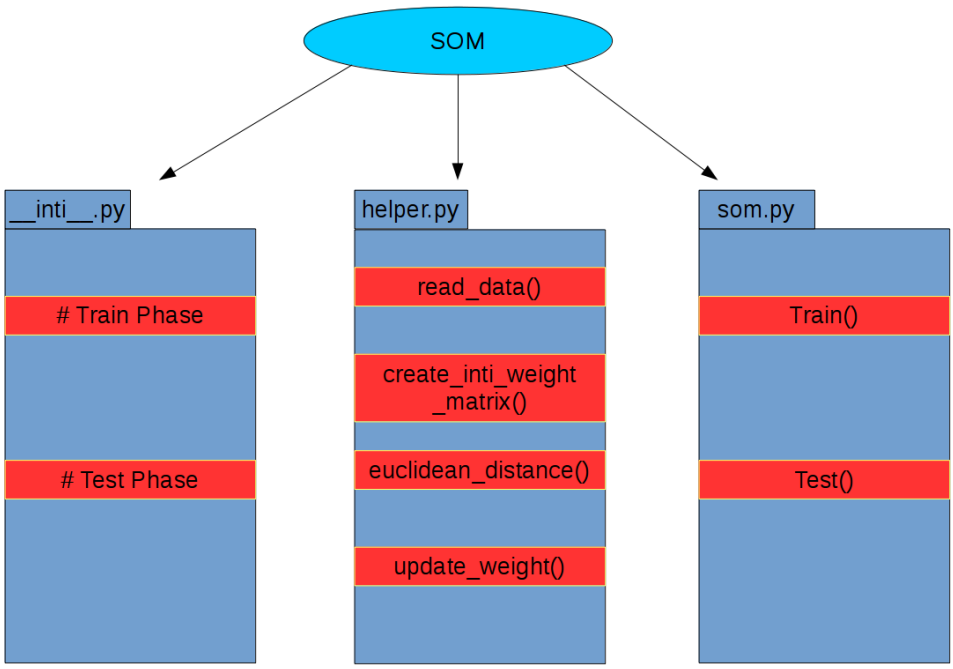
Self Organizing Map (SOM)

Kohonen network (SOM) is introduced around 1982 by Teuvo Kohonen. The Kohonen neural network differs considerably from the feed forward back propagation neural network. The Kohonen neural network differs both in how it is trained and how it recalls a pattern. The Kohohen neural network does not use any sort of activation function. Further, the Kohonen neural network does not use any sort of a bias weight. *The principle goal* of the self-organizing map is to transform an incoming signal pattern of arbitrary dimension into a one- or two dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion.

Details of the algorithm to compute SOM:

1. Load the train data.
2. Create initial weight matrix randomly.
3. Train model (for # of epochs).
   3.1 calculate euclidean distance for each feature in the data.
   3.2 get the min distance of the features.
   3.3 update the wight of this min feature.
4. Load the test data.
5. Test model.
   5.1 calculate euclidean distance using the final saved wight from training.
   5.2 get the min distance for the features.
   5.3 set the data to the class based on min distance.

The structure of the program (SOM):



Start processes:

- I create this example to test the program and explain

1- **Load the train data.**

(1) start to load the data from the files.
(2) we have two files (Patient , control) with size 10 X 650.

2- **initial weight matrix**

(1) create initial weight matrix randomly with value (0-1)
(2) the size of this matrix depend on the input and the output
(3) the input (20 X 650) and the output 2 so the weight matrix size is (2 X 650).

3- **Train model (for # of epochs)**

(3.1) calculate euclidean distance for each feature in the data.

- Each input, X, compared to all the W ij for all output neurons vectors sing the Euclidean distance as follows:

$$D_{ij} = |X - W_{ij}| = \sqrt{(x_1 - w_{ij1})^2 + \cdots + (x_n - w_{ijn})^2}$$

(3.2)get the min distance of the features.

- the neuron whose weights vector is the most similar to the input vector, winner neuron determined based on the minimal Euclidean distance

$$D_{k_1 k_2} = \arg\min D_{ij}$$

(3.3) update the wight of this min feature.

- Only update weight associated with winning output unit at each iteration based on the min distance.

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\ \ h_{ij}(t)\ \ (x(t) - w_{ij}(t))$$

4- Load the test data

(1) start to load the data from the file to predict the class (patient, Control)
(2) we have files with size 4 X 650.

5- Test model

(5.1) calculate euclidean distance using the final saved wight from training.
(5.2) get the min distance for the features.

(5.3) Set the data have min distance to class

- the winner neuron is selected on the basis of minimum distance from the input pattern.

important information

(1) Final Result

1- Test #1 (Class 1 and 2)

- **['class_1', 'class_1', 'class_2', 'class_2']**

2- Test #2 (Patient and Control)

- **['Control', 'Patient', 'Control', 'Patient']**

(2) Machine information

- By using machine have this information:

Information Computer	
Processor	Up to 8th Gen Intel® Core™ i7 Processor
Graphics	- Intel Integrated Graphics - NVIDIA® GeForce® 940MX(4GB)
Memory Ram	16 GB
Thread(s) per core	2
Core(s) per socket	4
Socket(s)	1
CPU(s) = Thread(s) per core X Core(s) per socket X Socket(s)	8

(3) programing language

Python 3	
Library	**version**
numpy	1.18.1
random	---
argparse	---

(3) OS: Ubuntu 16.04