Jayathilaka H.A.D.T.T.

E/16/156

CO542

Neural Networks and Fuzzy Systems

2021

Lab 05 - SOM

Exercise 1

The Pima Indians Diabetes Database contains several medical predictor variables and one target variable, 'Outcome'. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

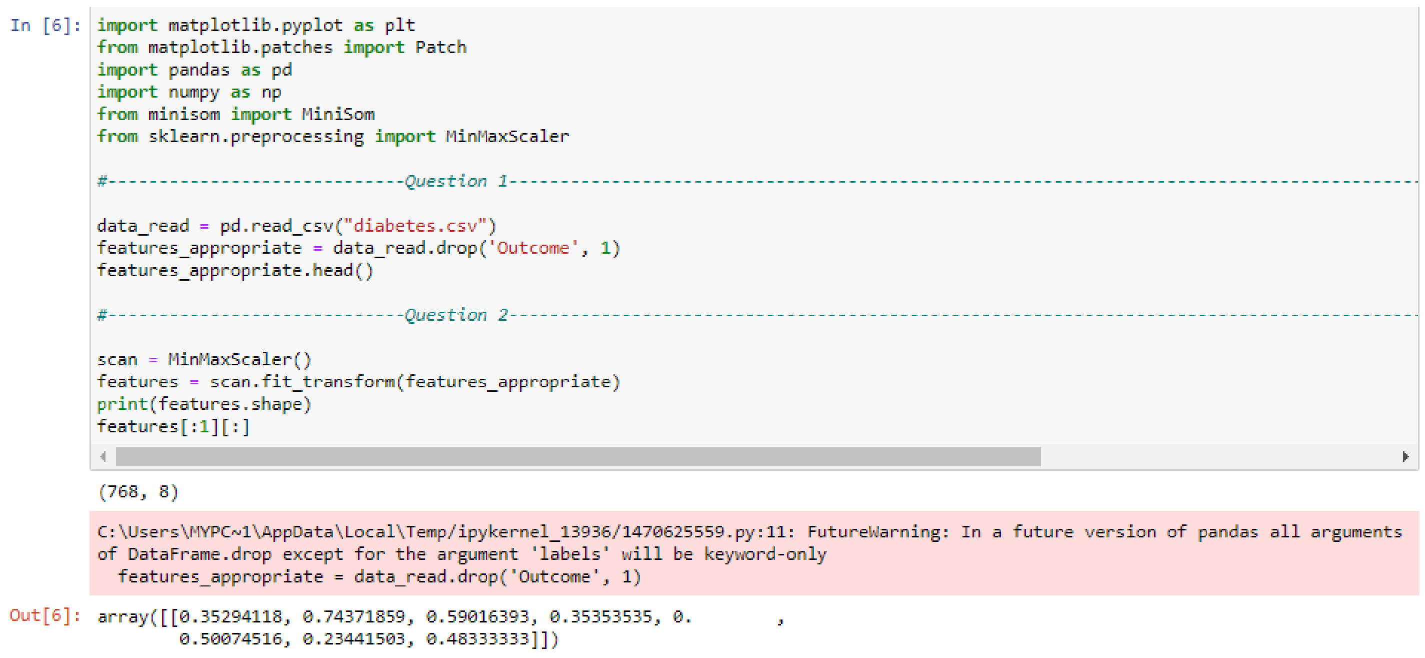
1. **Load the data set as a pandas' data frame and generate the feature matrix by selecting the appropriate features in the data set.**

```
In [1]: import matplotlib.pyplot as plt
        from matplotlib.patches import Patch
        import pandas as pd
        import numpy as np
        from minisom import MiniSom
        from sklearn.preprocessing import MinMaxScaler

        #---------------------------Question 1---------------------------

        data_read = pd.read_csv("diabetes.csv")
        features_appropriate = data_read.drop('Outcome', 1)
        features_appropriate.head()
```

Out[1]:

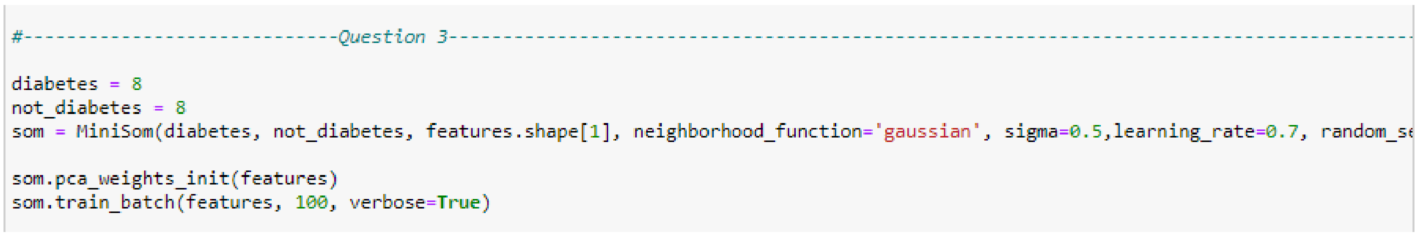
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

2. **Use sklearn.preprocessing.MinMaxScaler normalize the data between 0 and 1.**



```
In [6]: import matplotlib.pyplot as plt
        from matplotlib.patches import Patch
        import pandas as pd
        import numpy as np
        from minisom import MiniSom
        from sklearn.preprocessing import MinMaxScaler

        #----------------------------Question 1------------------------------------------------------------------------------------

        data_read = pd.read_csv("diabetes.csv")
        features_appropriate = data_read.drop('Outcome', 1)
        features_appropriate.head()

        #----------------------------Question 2------------------------------------------------------------------------------------

        scan = MinMaxScaler()
        features = scan.fit_transform(features_appropriate)
        print(features.shape)
        features[:1][:]

        (768, 8)

        C:\Users\MYPC~1\AppData\Local\Temp/ipykernel_13936/1470625559.py:11: FutureWarning: In a future version of pandas all arguments
        of DataFrame.drop except for the argument 'labels' will be keyword-only
          features_appropriate = data_read.drop('Outcome', 1)

Out[6]: array([[0.35294118, 0.74371859, 0.59016393, 0.35353535, 0.        ,
                0.50074516, 0.23441503, 0.48333333]])
```

3. **Initialize the weights using MiniSom.pca weights init function (Also identify the usage of ran▯dom weights init method and mention an advantage of using pca weights init).**



```
#----------------------------Question 3------------------------------------------------------------------------------------

diabetes = 8
not_diabetes = 8
som = MiniSom(diabetes, not_diabetes, features.shape[1], neighborhood_function='gaussian', sigma=0.5,learning_rate=0.7, random_se

som.pca_weights_init(features)
som.train_batch(features, 100, verbose=True)
```

random_weights_init : to initializes the weights of the SOM picking random samples from data

pca_weights_init : to initializes the weights to span the first two principal components

Advantages of using pca_weights_init

- Its initialization doesn't depend on random processes
- The training process converge faster

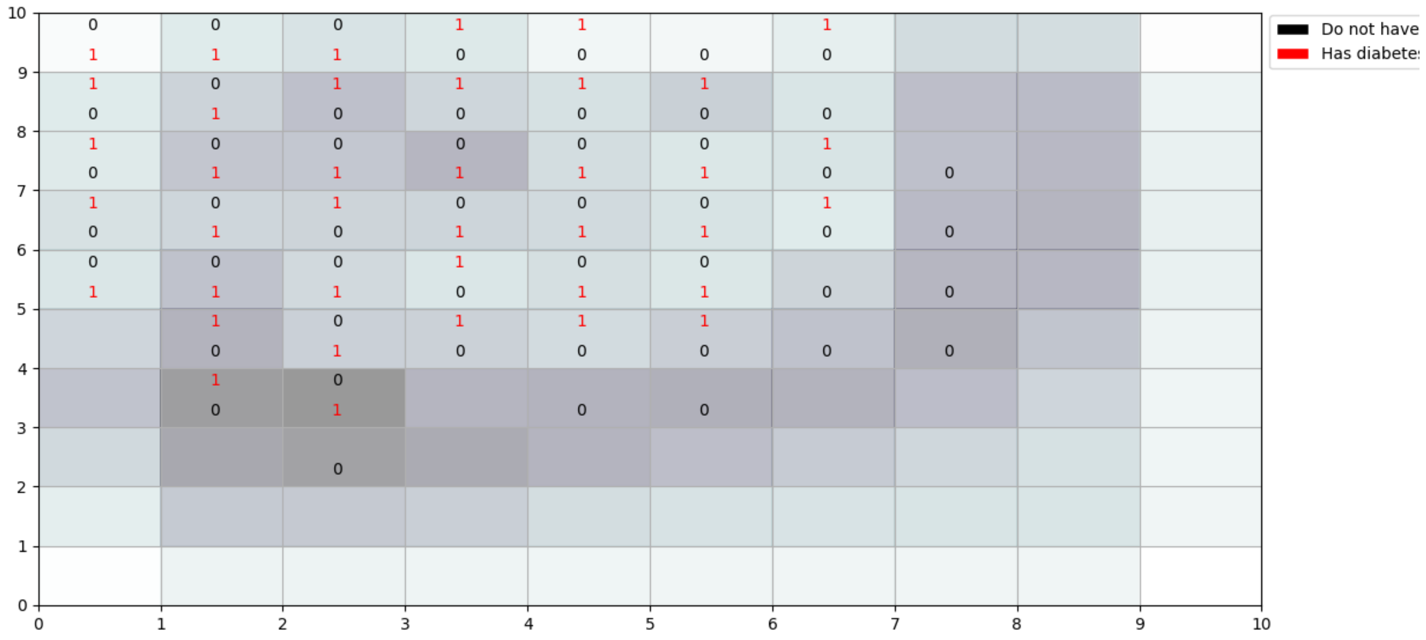
4. **Identify the difference between Minisom.train batch and Minisom.train random methods available minisom and use one of the models to train your self-organizing map.**

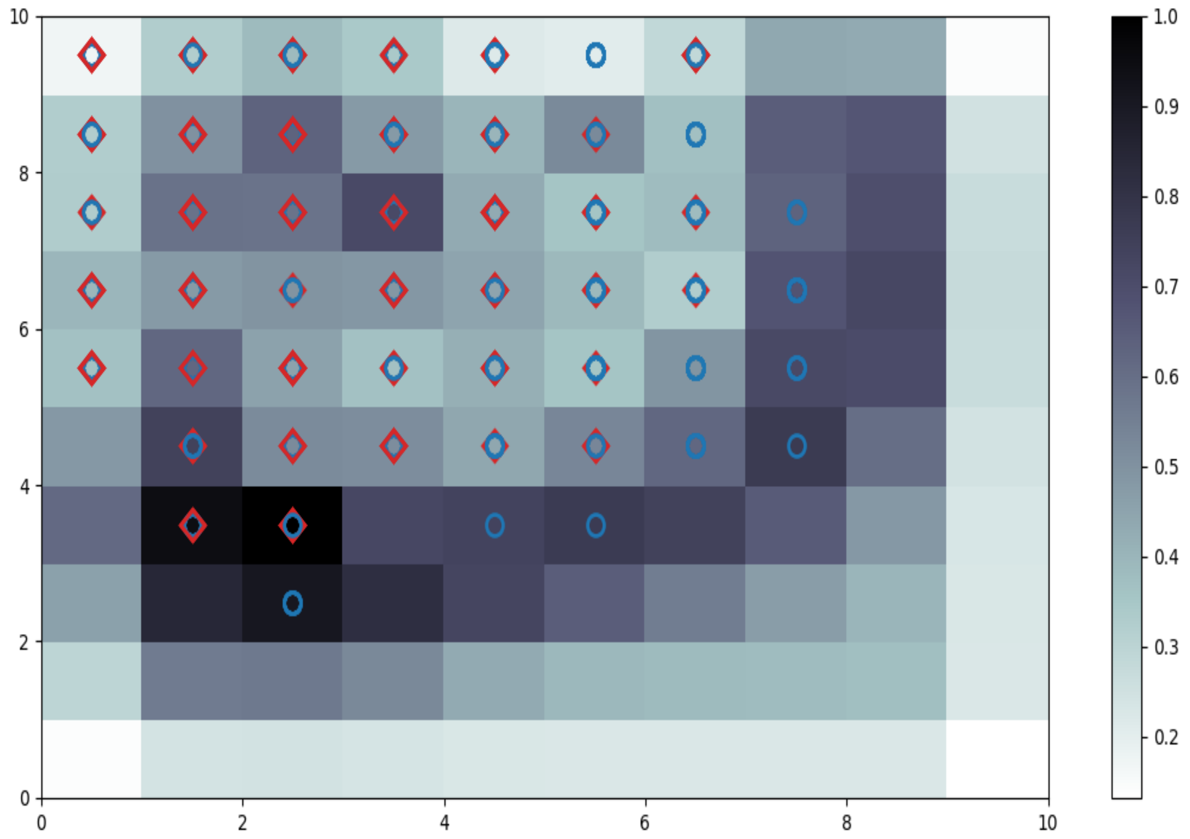
train_random : Train the model by picking random samples from data.
train_batch : The samples are picked in the order that they are stored.

5. **Study about the distance map and mention what does it indicate related to the self-organizing maps.**

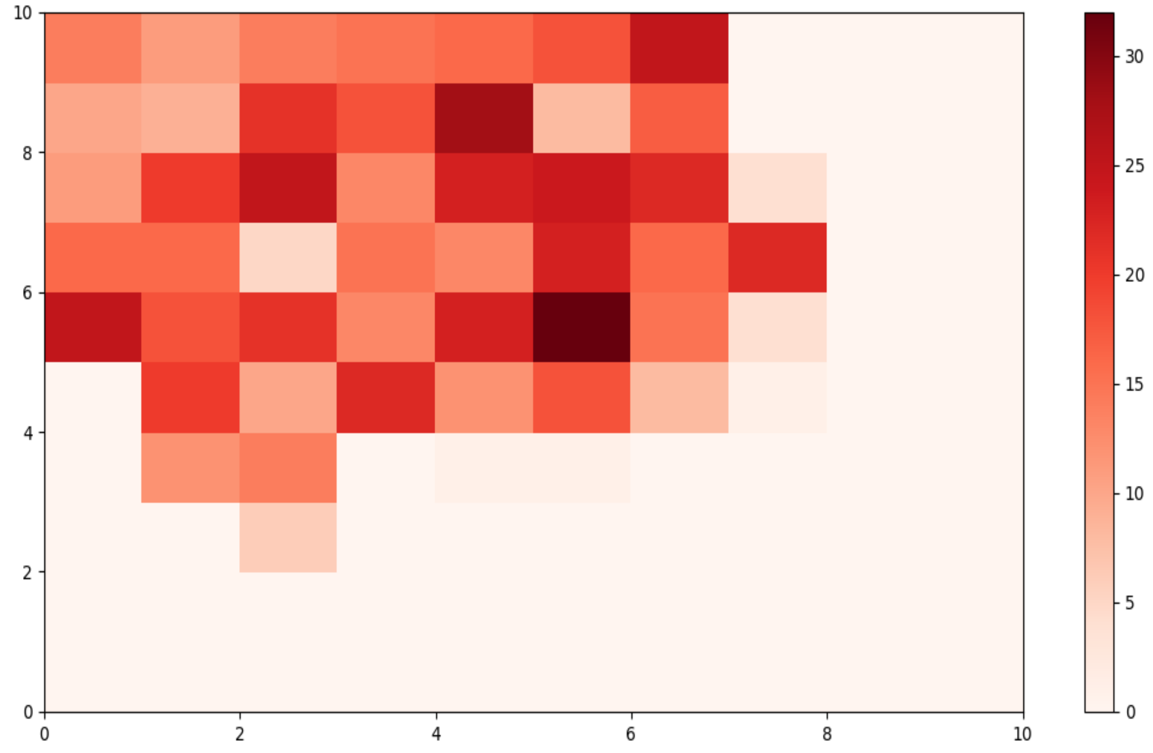
It returns the weights' distance map, with each cell representing the normalized sum of the distances between neurons.

6. **Use Minisom.distance map function to visualize the results obtained from the training process and use appropriate markers to indicate individual samples matched into each cell (according to the classes; patient who has diabetes/ do not have diabetes define two markers).**

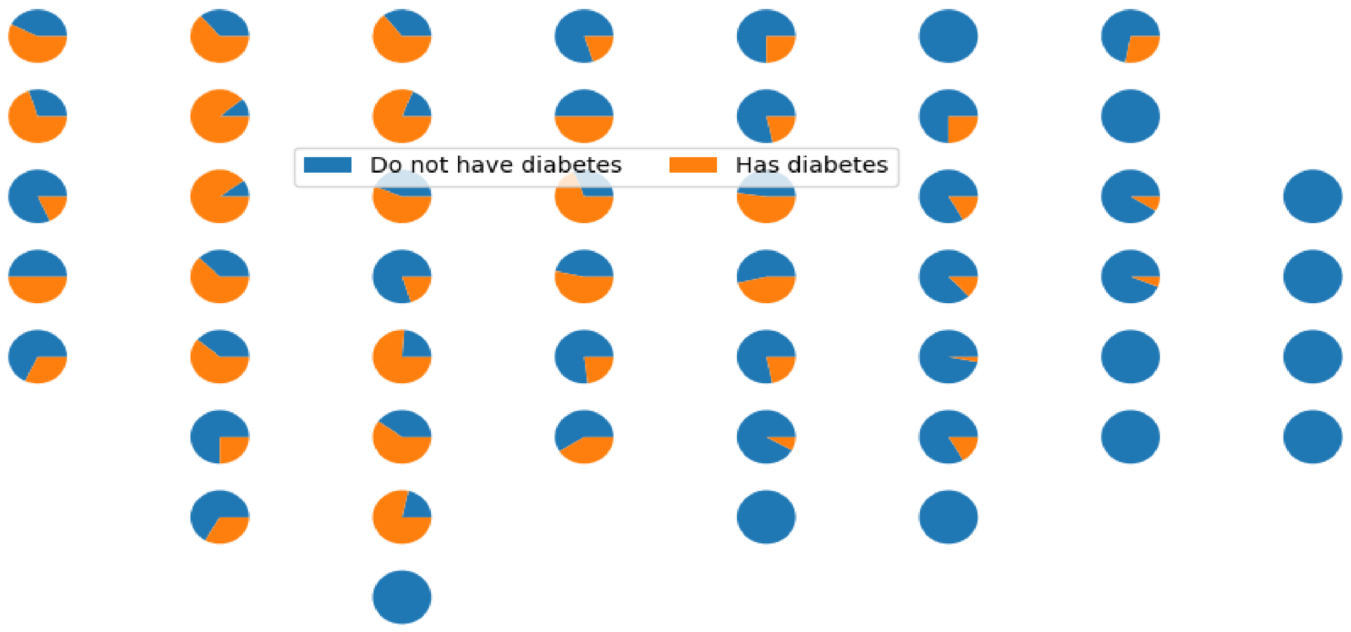


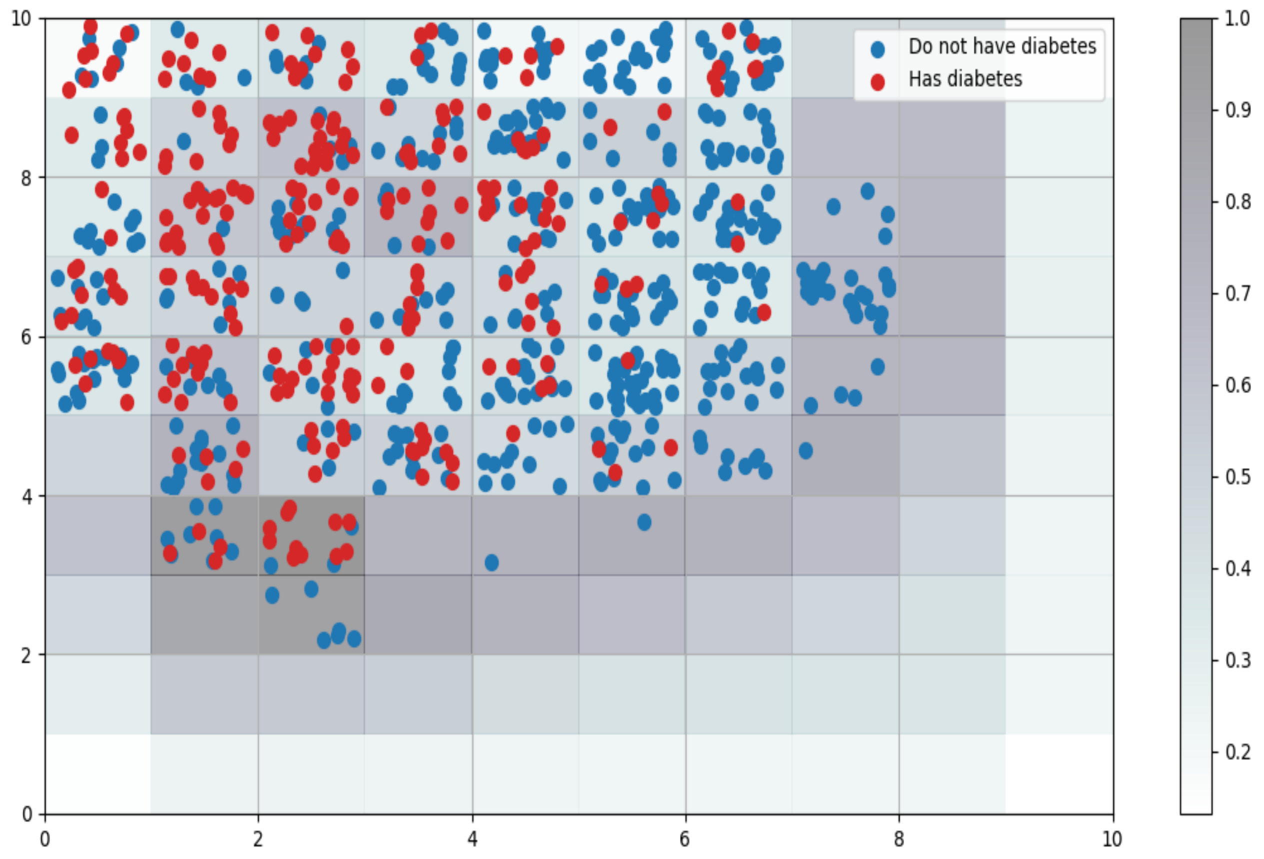


7. **Generate another grid to indicate how often each neuron is activated using, MiniSom.activation response.**



8. **Visualize the proportion of samples per class falling in a specific neuron using the, matplotlib.gridspec.GridSpec and matplotlib.patches.Patcha**





Do not have diabetes
Has diabetes
0
2
4
6
8
10
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0