“Neural Networks”

Shervin Halat

98131018

Homework 2

“First Part”

1.

Generally, a SOM is used as an unsupervised learning approach to produce a discretized low-dimensional representation of the input space of the input data. This is done by forcing the network neurons to respond to each input vector based on its distance to the input. This procedure is implemented below :

.

.

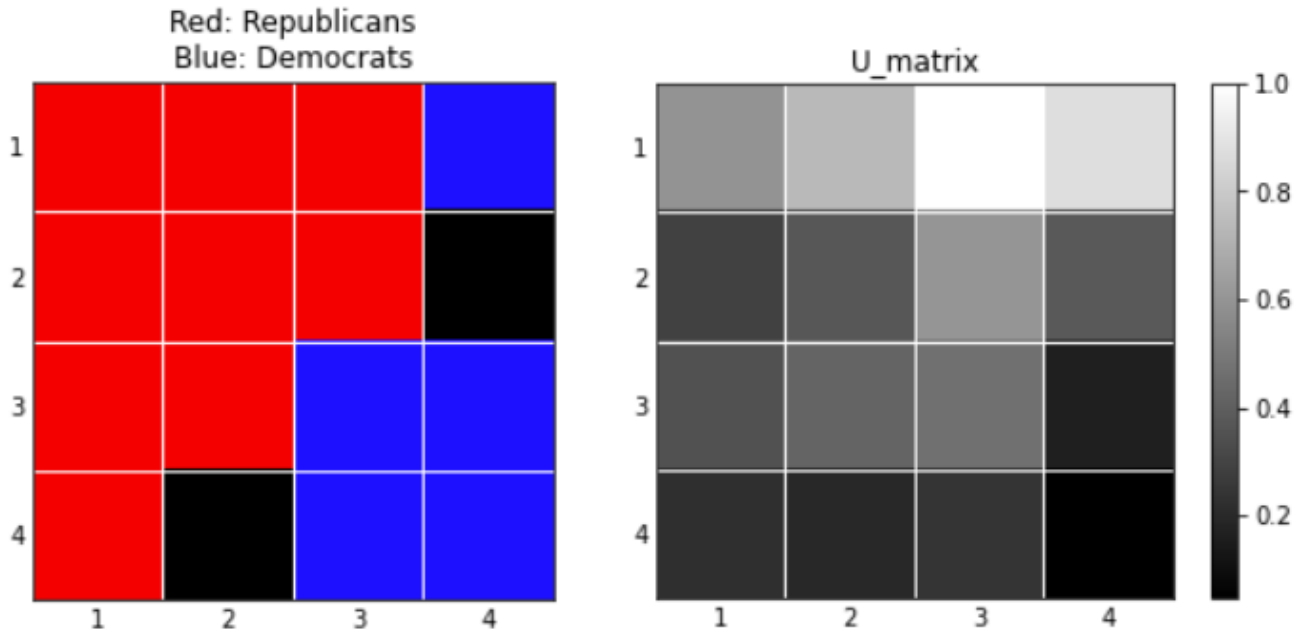
.

2.

Implemented algorithm has the following attributes:

Gaussian neighborhood strength with exponentially decreasing sigma and learning rate values. Initial value of learning rate is 0.1 which doesn't fall below 0.01 with epochs. Also, overall, 3000 epochs are considered.

By implementing the mentioned algorithm and considering parameters above, the following 2D SOM map of size 4x4 will be generated: (left figure colors denote most repeated mapped record party for each cell)



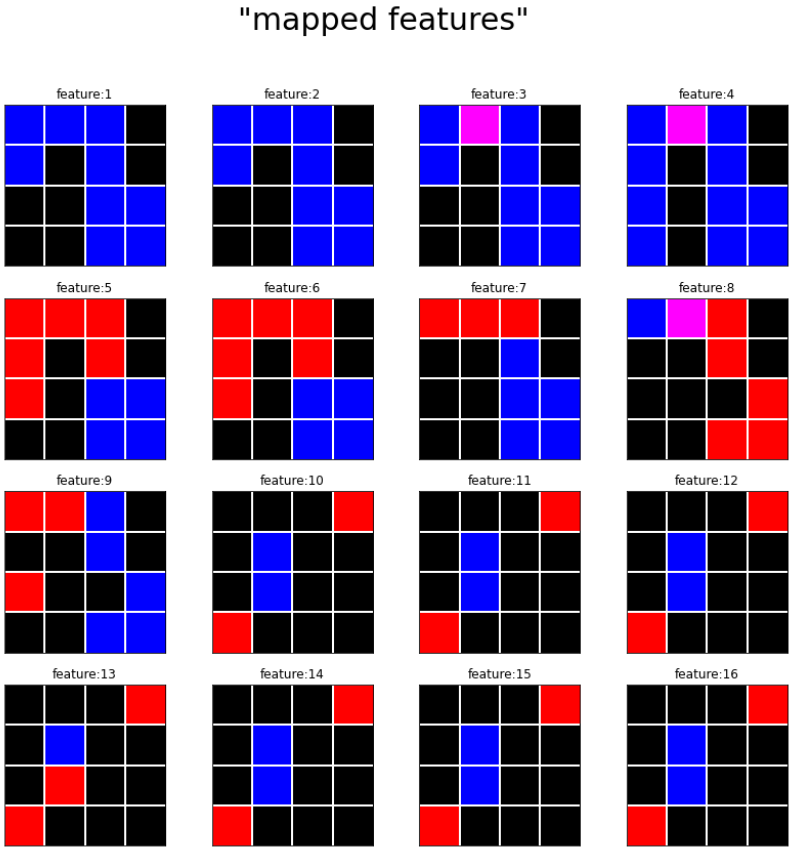
3.

The left most SOM map above, illustrates the way our model distributes our records in a 2D space which in our case has perfectly classified our input data based on the party to which each of our records belongs to. Actually, each cell of the SOM maps represents the center of each cluster by a vector. The right most figure, which is called U-matrix, denotes overall differentiation between each cell vector with its neighbor cells. Therefore, we expect that cells within mutual clusters be darker in the U-matrix and the whiter cells specify the boundary between clusters (or classes).

By comparing the generated SOM map by each of the mapped features which are shown below, it can be figured out that features of 5 and 6 are the best features to distinguish party of each member of congress. Features of 5 and 6 represent the followings respectively:

5: House Vote #223 2020-11-19T14:05:00 - On Ordering the Previous Question: H.Res. 1224: Providing for consideration of the bill (H.R. 8294) to amend the National Apprenticeship ...

6: House Vote #224 2020-11-19T14:58:00 - H.Res. 1224: Providing for consideration of the bill (H.R. 8294) to amend the National Apprenticeship Act and expand the national apprenticeship system to include apprenticeships



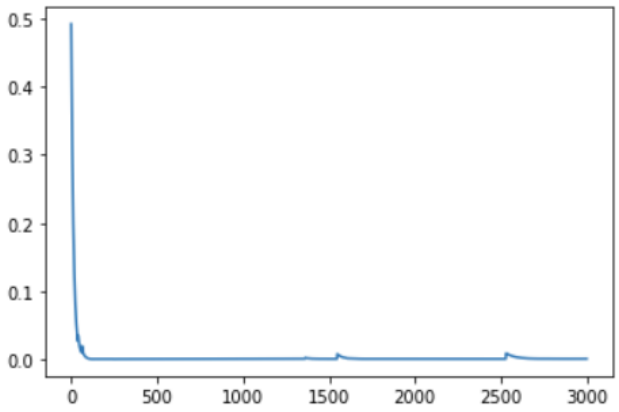
This selection is rooted in the fact that these two features maps (5 & 6) perfectly overlap with the SOM map and therefore, have the most sharing information with SOM map compared to other features' maps. Each one of the feature maps above illustrate that by only considering the corresponding feature, how does our SOM model classify (or cluster) our input data. Therefore, the best match with SOM map determines the most informative feature.

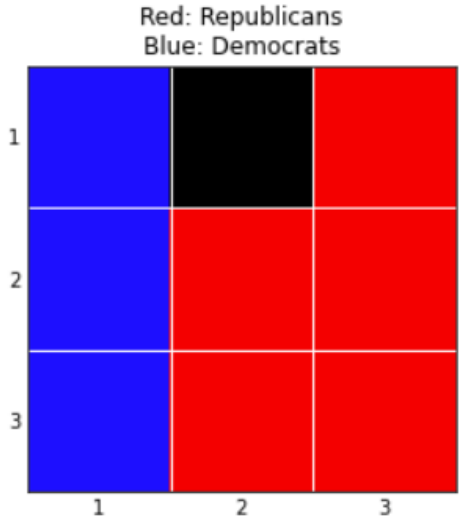
4.

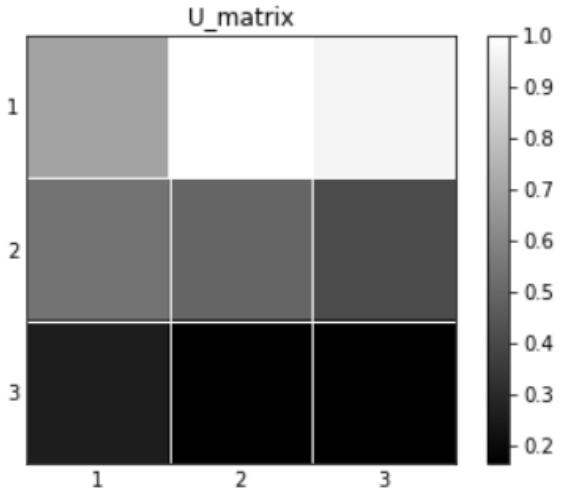
Dimension of 4x4 was considered as the best dimension for the visualization of the data as this dimension results in both small number of dead neurons and also has the most compacted clusters compared to the higher dimensions. In the following, obtained maps for the dimensions of 3, 4, 5, 6, and 8 are illustrated:

3x3:

```
Iteration: 0, LR: 0.1, Sigma: 2.0999999999999996, J: 0.4919279141397212
Iteration: 300, LR: 0.09048374180359596, Sigma: 1.9001585778755146, J: 0.0010094051416158726
Iteration: 600, LR: 0.0818730753077982, Sigma: 1.7193345814637615, J: 0.0010999951201524757
Iteration: 900, LR: 0.0740818220681718, Sigma: 1.5557182634316074, J: 0.001253706087243122
Iteration: 1200, LR: 0.06703200460356394, Sigma: 1.4076720966748424, J: 0.0011889408236388035
Iteration: 1500, LR: 0.06065306597126335, Sigma: 1.27371438539653, J: 0.0012191809100563407
Iteration: 1800, LR: 0.05488116360940265, Sigma: 1.1525044357974554, J: 0.0012414765771023304
Iteration: 2100, LR: 0.04965853037914096, Sigma: 1.0428291379619599, J: 0.0013216442794543363
Iteration: 2400, LR: 0.044932896411722156, Sigma: 0.9435908246461652, J: 0.0012534441233717727
Iteration: 2700, LR: 0.04065696597405991, Sigma: 0.853796285455258, J: 0.0015782598904204017
```

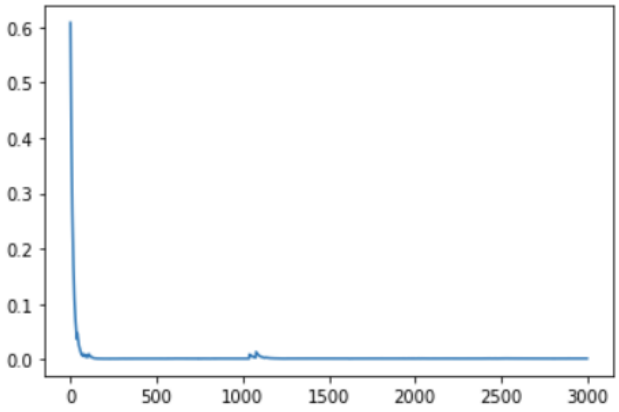


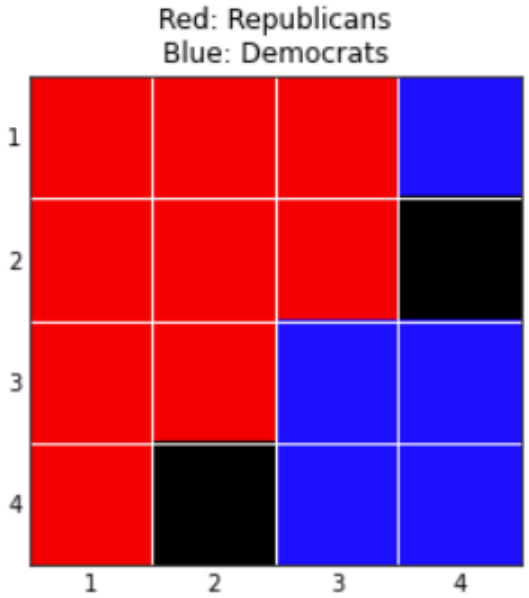


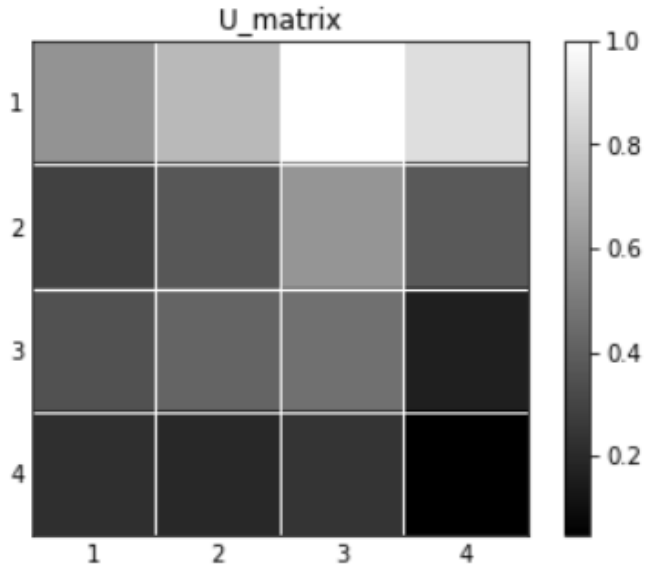


4x4:

```
Iteration: 0, LR: 0.1, Sigma: 2.8, J: 0.6082342105625288
Iteration: 300, LR: 0.09048374180359596, Sigma: 2.5335447705006864, J: 0.0014857941668576316
Iteration: 600, LR: 0.0818730753077982, Sigma: 2.292446108618349, J: 0.001675649199222926
Iteration: 900, LR: 0.0740818220681718, Sigma: 2.07429101790881, J: 0.001562945344898719
Iteration: 1200, LR: 0.06703200460356394, Sigma: 1.87689612889979, J: 0.0018563702858195783
Iteration: 1500, LR: 0.06065306597126335, Sigma: 1.6982858471953735, J: 0.00173624211944784
Iteration: 1800, LR: 0.05488116360940265, Sigma: 1.5366725810632742, J: 0.0016973252620246697
Iteration: 2100, LR: 0.04965853037914096, Sigma: 1.3904388506159469, J: 0.001617138855540611
Iteration: 2400, LR: 0.044932896411722156, Sigma: 1.2581210995282204, J: 0.0017624991573727082
Iteration: 2700, LR: 0.04065696597405991, Sigma: 1.1383950472736775, J: 0.00166099405488651
```

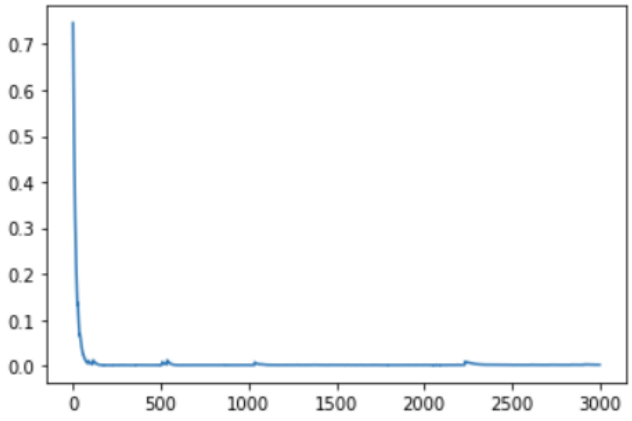


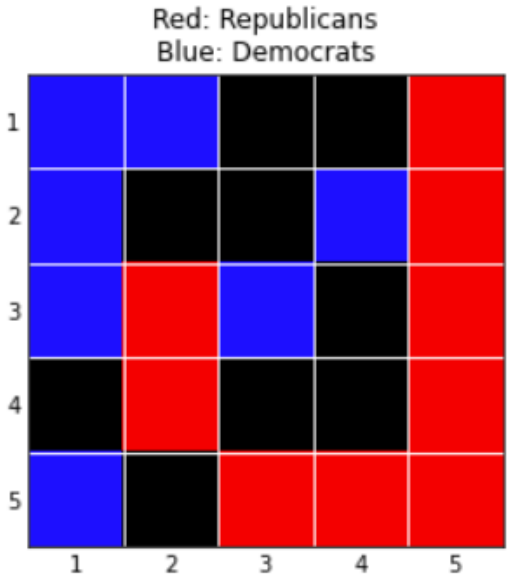


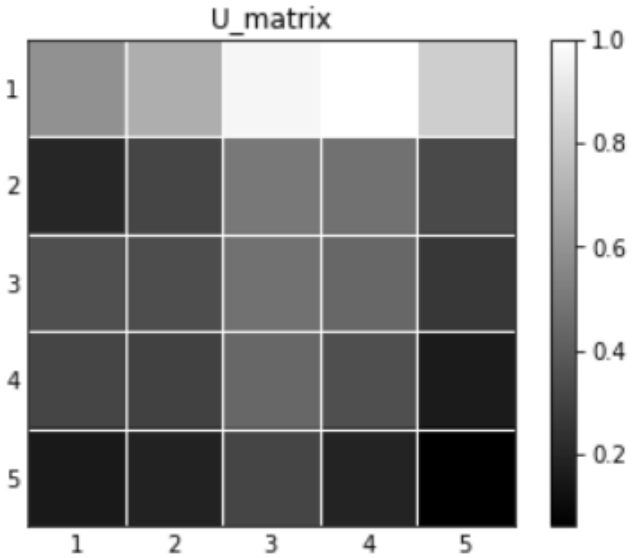


5x5:

```
Iteration: 0, LR: 0.1, Sigma: 3.5, J: 0.7457576487537241
Iteration: 300, LR: 0.09048374180359596, Sigma: 3.1669309631258584, J: 0.0019101532154120766
Iteration: 600, LR: 0.0818730753077982, Sigma: 2.8655576357729364, J: 0.0018135996556388552
Iteration: 900, LR: 0.0740818220681718, Sigma: 2.5928637723860124, J: 0.0019459014775243327
Iteration: 1200, LR: 0.06703200460356394, Sigma: 2.3461201611247375, J: 0.002257726440385889
Iteration: 1500, LR: 0.06065306597126335, Sigma: 2.122857308994217, J: 0.0021390676066261865
Iteration: 1800, LR: 0.05488116360940265, Sigma: 1.9208407263290928, J: 0.001930412374600325
Iteration: 2100, LR: 0.04965853037914096, Sigma: 1.7380485632699336, J: 0.0019679524191102363
Iteration: 2400, LR: 0.044932896411722156, Sigma: 1.5726513744102755, J: 0.0024185701784333142
Iteration: 2700, LR: 0.04065696597405991, Sigma: 1.4229938090920968, J: 0.002207044227774351
```

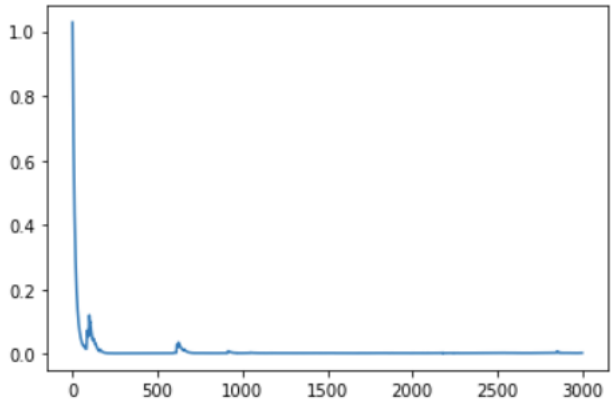


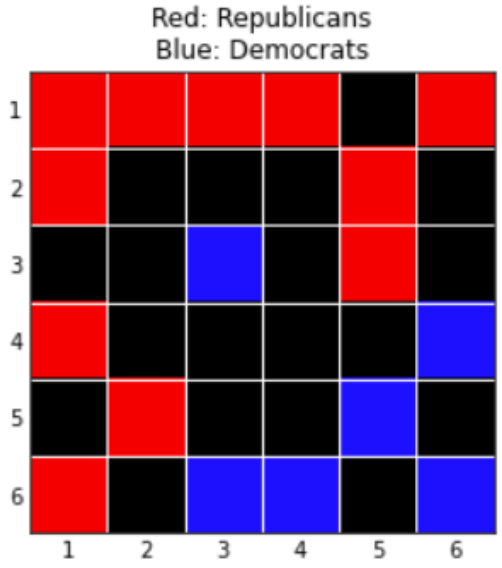


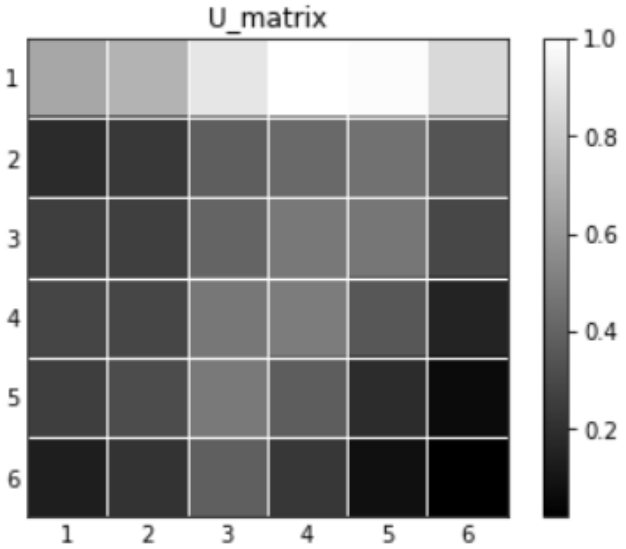


6x6:

```
Iteration: 0, LR: 0.1, Sigma: 4.199999999999999, J: 1.0281100664494336
Iteration: 300, LR: 0.09048374180359596, Sigma: 3.800317155751029, J: 0.0018940910536982838
Iteration: 600, LR: 0.0818730753077982, Sigma: 3.438669162927523, J: 0.0034537298510313942
Iteration: 900, LR: 0.0740818220681718, Sigma: 3.1114365268632147, J: 0.0023698995215183875
Iteration: 1200, LR: 0.06703200460356394, Sigma: 2.8153441933496848, J: 0.0025926256929620925
Iteration: 1500, LR: 0.06065306597126335, Sigma: 2.54742877079306, J: 0.002574102974778355
Iteration: 1800, LR: 0.05488116360940265, Sigma: 2.305008871594911, J: 0.0024138463649108466
Iteration: 2100, LR: 0.04965853037914096, Sigma: 2.0856582759239197, J: 0.0025531734112128643
Iteration: 2400, LR: 0.044932896411722156, Sigma: 1.8871816492923303, J: 0.0030764432964208374
Iteration: 2700, LR: 0.04065696597405991, Sigma: 1.707592570910516, J: 0.002577476255092044
```

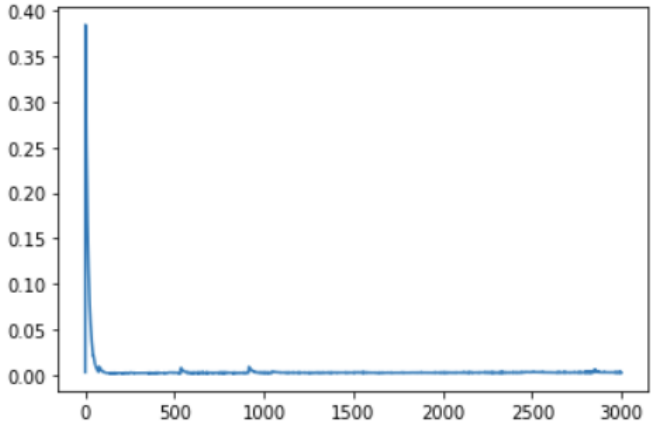


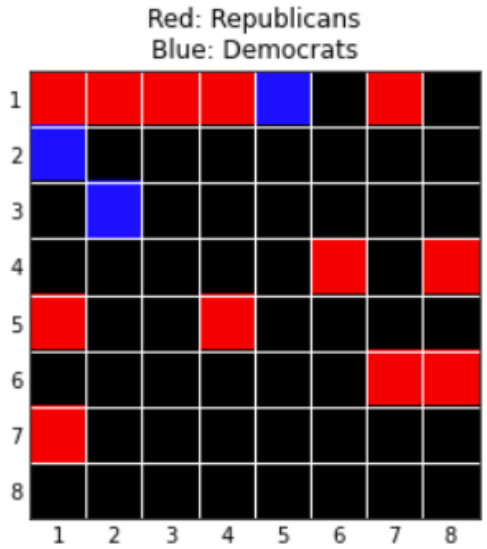


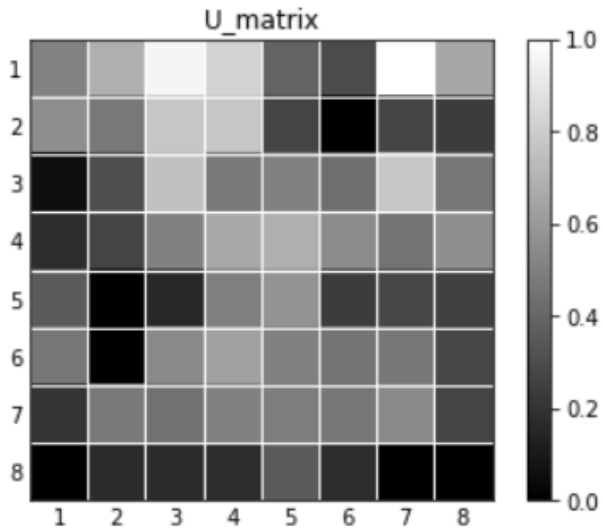


8x8:

```
Iteration: 0, LR: 0.1, Sigma: 4.199999999999999, J: 0.003006401778373877
Iteration: 300, LR: 0.09048374180359596, Sigma: 3.800317155751029, J: 0.002179231328952467
Iteration: 600, LR: 0.0818730753077982, Sigma: 3.438669162927523, J: 0.002147004513905461
Iteration: 900, LR: 0.0740818220681718, Sigma: 3.1114365268632147, J: 0.002546770990586835
Iteration: 1200, LR: 0.06703200460356394, Sigma: 2.8153441933496848, J: 0.0026451066345786066
Iteration: 1500, LR: 0.06065306597126335, Sigma: 2.54742877079306, J: 0.002583187240433443
Iteration: 1800, LR: 0.05488116360940265, Sigma: 2.305008871594911, J: 0.002670334885967059
Iteration: 2100, LR: 0.04965853037914096, Sigma: 2.0856582759239197, J: 0.002739731563743474 7
Iteration: 2400, LR: 0.044932896411722156, Sigma: 1.8871816492923303, J: 0.0027401845926194246
Iteration: 2700, LR: 0.04065696597405991, Sigma: 1.707592570910516, J: 0.00274019995000854
```







“Second Part”

1.

As it was mentioned in the previous part, SOM map contributes in clustering the data and separating data points of each class by mapping similar ones to the closer neurons. Thereby, a classifier may be defined in order to classify datapoints based on the distance of the winner neuron from other neurons. Therefore, the network learns to classify each data point to a class which has approximately the least mean distance between the winner neuron and neurons which represent that class. Hence, for this part, a vector which stores the distances of each neuron from the input vector will be considered as the alternative feature for the initial data.

2.

A MPL model with two hidden layers was defined with 15 and 8 nodes in first and second hidden layers respectively. Although models with only one hidden layer converged to high accuracies, a model with two hidden layers seemed to converge faster.

This may be because of the fact that as we saw in the previous part, generated SOM maps seemed to be separable with a convex, hence, extracted features would be separable with only one hidden layer; Whereas, In order to avoid sudden changes in number of nodes in neighbor layers, two hidden layers were considered to solve this problem.

The obtained optimized MLP model has reached the validation accuracy of 100% after 55 epochs by early stopping callback. The configuration of the obtained model is as follows:

```
model4.summary()
```

```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_33 (Flatten)         (None, 16)                0
_________________________________________________________________
dense_93 (Dense)             (None, 15)                255
_________________________________________________________________
dense_94 (Dense)             (None, 8)                 128
_________________________________________________________________
dense_95 (Dense)             (None, 1)                 9
=================================================================
Total params: 392
Trainable params: 392
Non-trainable params: 0
_________________________________________________________________
```

```
log_dir = "../logs/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
cb2 = keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
model4 = keras.models.Sequential([
                                  keras.layers.Flatten(input_shape=(som_net4.map.shape[0],som_net4.map.shape[1])),
                                  keras.layers.Dense(units = 15 , activation = 'relu' ),
                                  keras.layers.Dense(units = 8 , activation = 'relu' ),
                                  keras.layers.Dense(units = 1 , activation = 'sigmoid'),
])

model4.compile(optimizer = 'Adam' , loss = 'binary_crossentropy' , metrics = ['accuracy'])
model4.fit(x_train2, y_train2 , epochs=500, validation_split = 2/9, callbacks = [cb2,cb1])
```

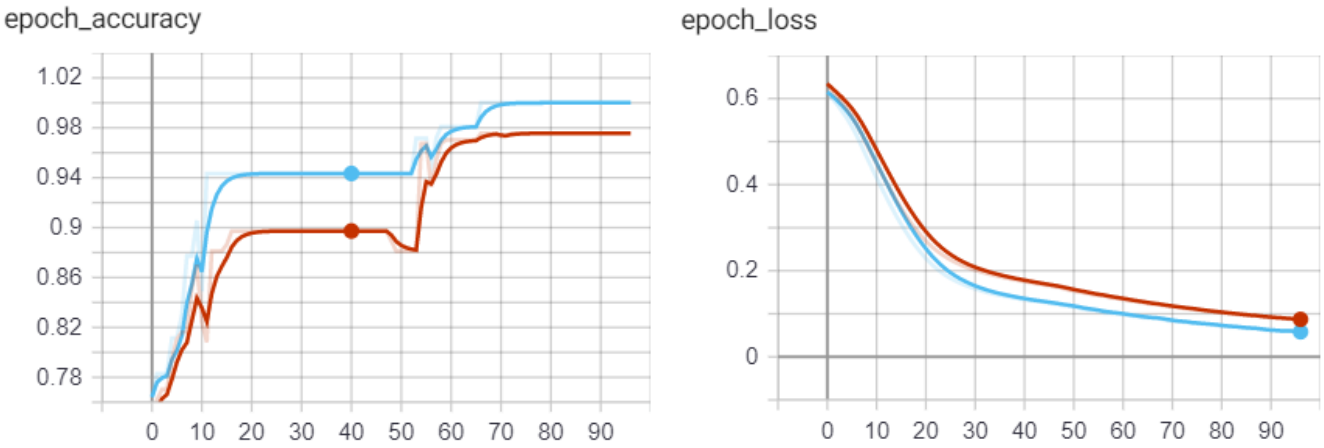
```
12/12 [==============================] - 0s 5ms/step - loss: 0.0691 - accuracy: 0.9892 - val_loss: 0.0604 - val_accuracy: 1.0000
Epoch 55/500
12/12 [==============================] - 0s 5ms/step - loss: 0.0675 - accuracy: 0.9892 - val_loss: 0.0583 - val_accuracy: 1.0000
Early stopping callback!
<tensorflow.python.keras.callbacks.History at 0x7feb667d4860>
```

The test set accuracy has reached 98%:

```
results2 = model4.evaluate(x_test2,y_test2)
print('test data accuracy = {0:.2f}'.format(results2[1]))
```

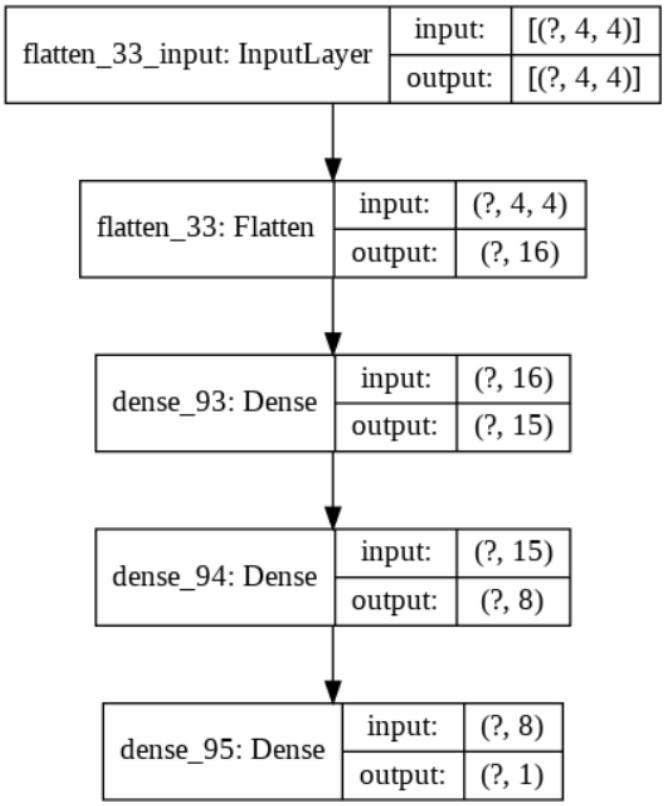
```
2/2 [==============================] - 0s 2ms/step - loss: 0.1826 - accuracy: 0.9811
test data accuracy = 0.98
```

The logs of the model are as follows:

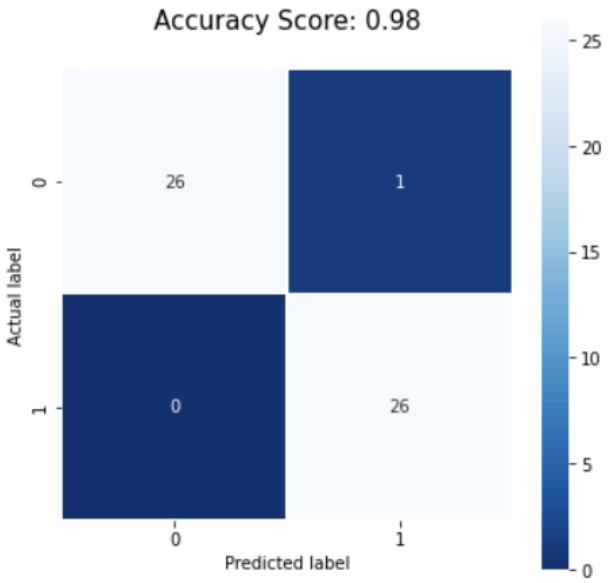


The visualization of the model is as follows:

```
1 keras.utils.plot_model(model4,show_shapes = True)
```



Lastly, the confusion matrix of the model is as follows:



It should be noted that, in order to simplify the model, three records of libertarian and independent parties where considered as democrats.

Results of conducted experiments for one hidden layer are as follows:

Hidden layer with 8 nodes:

More epochs, less test accuracy.

```
12/12 [==============================] - 0s 4ms/step - loss: 0.1118 - accuracy: 0.9865 - val_loss: 0.1065 - val_accuracy: 1.0000
Epoch 97/500
12/12 [==============================] - 0s 5ms/step - loss: 0.1111 - accuracy: 0.9865 - val_loss: 0.1058 - val_accuracy: 1.0000
Early stopping callback!
<tensorflow.python.keras.callbacks.History at 0x7feb6c747828>
```

```
1 results = model3.evaluate(x_test,y_test)
2 print('test data accuracy = {0:.2f}'.format(results[1]))
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.1597 - accuracy: 0.9623
test data accuracy = 0.96
```

Hidden layer with 5 nodes:

More epochs, less test accuracy.

```
12/12 [==============================] - 0s 5ms/step - loss: 0.1179 - accuracy: 0.9865 - val_loss: 0.1127 - val_accuracy: 1.0000
Epoch 129/500
12/12 [==============================] - 0s 5ms/step - loss: 0.1173 - accuracy: 0.9865 - val_loss: 0.1123 - val_accuracy: 1.0000
Early stopping callback!
<tensorflow.python.keras.callbacks.History at 0x7feb6c1cfeb8>
```

```
1 results = model3.evaluate(x_test,y_test)
2 print('test data accuracy = {0:.2f}'.format(results[1]))
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.1567 - accuracy: 0.9623
test data accuracy = 0.96
```

Also, another experiment was conducted in order to find out how change in SOM structure (i.e. Dimension) affects the proposed MLP classifier. The results are shown below for dimensions of 3, 5, 6, and 8:

3x3 SOM:

```
Epoch 86/500
12/12 [==============================] - 0s 2ms/step - loss: 0.0917 - accuracy: 0.9811 - val_loss: 0.0412 - val_accuracy: 1.0
000
Early stopping callback!

<tensorflow.python.keras.callbacks.History at 0x1dbacb270c8>
```

```
1 results3 = model3.evaluate(x_test,y_test)
2 print('test data accuracy = {0:.2f}'.format(results3[1]))
```

```
2/2 [==============================] - 0s 517us/step - loss: 0.1035 - accuracy: 1.0000
test data accuracy = 1.00
```

4x4 SOM:

```
Epoch 63/500
12/12 [==============================] - 0s 2ms/step - loss: 0.0768 - accuracy: 0.9892 - val_loss: 0.0558 - val_accuracy: 0.9
717
Early stopping callback!

<tensorflow.python.keras.callbacks.History at 0x1db90431488>
```

```
1 results4 = model4.evaluate(x_test,y_test)
2 print('test data accuracy = {0:.2f}'.format(results4[1]))
```

```
1/2 [==============>..............] - ETA: 0s - loss: 0.1329 - accuracy: 1.0000WARNING:tensorflow:Callbacks method `on_test_b
tch_end` is slow compared to the batch time (batch time: 0.0000s vs `on_test_batch_end` time: 0.0010s). Check your callbacks.
2/2 [==============================] - 0s 1ms/step - loss: 0.0970 - accuracy: 1.0000
test data accuracy = 1.00
```

5x5 SOM:

```
Epoch 62/500
12/12 [==============================] - 0s 2ms/step - loss: 0.0782 - accuracy: 0.9730 - val_loss: 0.0496 - val_accuracy: 0.9
906
Early stopping callback!

<tensorflow.python.keras.callbacks.History at 0x1dbacdc53c8>
```

```
results5 = model5.evaluate(x_test,y_test)
print('test data accuracy = {0:.2f}'.format(results5[1]))
```

```
2/2 [==============================] - 0s 997us/step - loss: 0.1665 - accuracy: 1.0000
test data accuracy = 1.00
```

6x6 SOM:

```
Epoch 45/500
12/12 [==============================] - 0s 2ms/step - loss: 0.0801 - accuracy: 0.9757 - val_loss: 0.0642 - val_accuracy: 0.9
623
Early stopping callback!

<tensorflow.python.keras.callbacks.History at 0x1dbadff5188>
```

```
results6 = model6.evaluate(x_test,y_test)
print('test data accuracy = {0:.2f}'.format(results6[1]))
```

```
2/2 [==============================] - 0s 499us/step - loss: 0.5641 - accuracy: 1.0000
test data accuracy = 1.00
```

8x8 SOM:

```
Epoch 74/500
12/12 [==============================] - 0s 2ms/step - loss: 0.0704 - accuracy: 0.9730 - val_loss: 0.0731 - val_accuracy: 0.9
717
Early stopping callback!

<tensorflow.python.keras.callbacks.History at 0x1dbaa1f0288>
```

```
results8 = model8.evaluate(x_test,y_test)
print('test data accuracy = {0:.2f}'.format(results8[1]))
```

```
2/2 [==============================] - 0s 498us/step - loss: 0.0871 - accuracy: 1.0000
test data accuracy = 1.00
```

3.

Considering obtained results above, there seems no significant changes occur in the performance of MLP classifiers with different SOM structures. But considering previous part figures, it can be figured out that as we increase SOM maps dimensions, more dead neurons will be appeared which adversely affects the maps visualization. Also, too small maps result in less informative U-matrixes. For example, SOM with dimension of 3 gives relatively no useful information. Therefore, in our case that there are only two classes, dimension of 4 seems to be the optimal choice.