Artificial Neural Networks

Kohonen Model

Homework #3

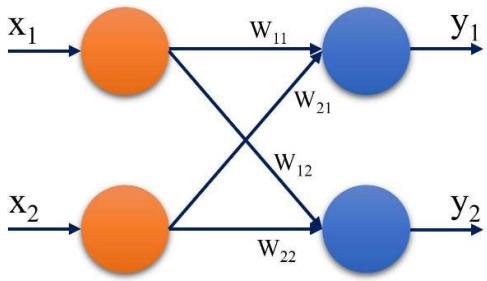
Poorya MohammadiNasab

(400722138)

Contents

Problem 1

Consider the following inputs: $I_1 = [0\ \ 1]^T$, $I_2 = [0\ \ 1]^T$, and $I_3 = [1\ \ 1]^T$. Using the Kohonen learning algorithm, train a network for a single epoch ($\alpha = 0.2$).



Initialize random weights:

$$W_{ij} = \begin{bmatrix} 0.2 & 0.1 \\ 0.3 & 0.4 \end{bmatrix}$$

* First input Vector: $I_1 = [0\ \ 1]^T$

Calculate Euclidian distance: $D(i) = \sum (W_{ij} - x_i)^2$

$$D(1) = (0.2 - 1)^2 + (0.3 - 0)^2 = 0.73$$

$$D(2) = (0.1 - 1)^2 + (0.4 - 0)^2 = 0.97$$

D(1) is the winner. Apply weight update on D(1).

$$W_{ij}(new) = W_{ij}(old) + \alpha \left(x_i - W_{ij}(old)\right)$$

$$W_{11}(new) = 0.2 + 0.2(1 - 0.2) = 0.36$$

$$W_{21}(new) = 0.3 + 0.2(0 - 0.3) = 0.24$$

$$W_{ij} = \begin{bmatrix} 0.36 & 0.1 \\ 0.24 & 0.4 \end{bmatrix}$$

* Second input vector: $I_2 = [0\ \ 1]^T$

Calculate Euclidian distance: $D(i) = \sum (W_{ij} - x_i)^2$

$$D(1) = (0.36 - 1)^2 + (0.24 - 0)^2 = 0.4672$$

$$D(2) = (0.1 - 1)^2 + (0.4 - 0)^2 = 0.97$$

D(1) is the winner. Apply weight update on D(1).

$$W_{ij}(new) = W_{ij}(old) + \alpha\left(x_i - W_{ij}(old)\right)$$

$$W_{11}(new) = 0.36 + 0.2(1 - 0.36) = 0.488$$

$$W_{21}(new) = 0.24 + 0.2(0 - 0.24) = 0.192$$

$$W_{ij} = \begin{bmatrix} 0.488 & 0.1 \\ 0.192 & 0.4 \end{bmatrix}$$

* Third input vector: $I_3 = [1 \quad 1]^T$

Calculate Euclidian distance: $D(i) = \sum\left(W_{ij} - x_i\right)^2$

$D(1) = (0.488 - 1)^2 + (0.192 - 1)^2 = 0.915008$

$D(2) = (0.1 - 1)^2 + (0.4 - 1)^2 = 1.17$

D(1) is the winner. Apply weight update on D(1).

$$W_{ij}(new) = W_{ij}(old) + \alpha\left(x_i - W_{ij}(old)\right)$$

$$W_{11}(new) = 0.488 + 0.2(1 - 0.488) = 0.5904$$

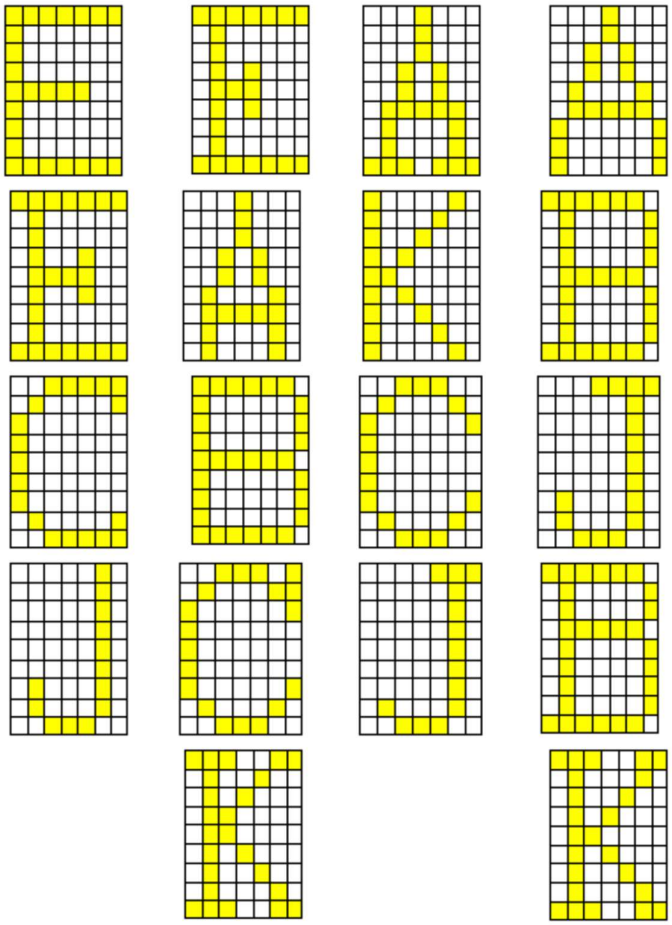
$$W_{21}(new) = 0.192 + 0.2(1 - 0.192) = 0.3536$$

$$W_{ij} = \begin{bmatrix} 0.5904 & 0.1 \\ 0.3536 & 0.4 \end{bmatrix}$$

Problem 2

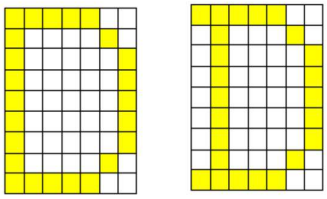
2 - a

Train a Kohonen grid with the dimensions of 9 * 7 using the following patterns. Then, indicate how these patterns will be categorized after training.



2 - b

How will the trained network predict the category of the following patterns?

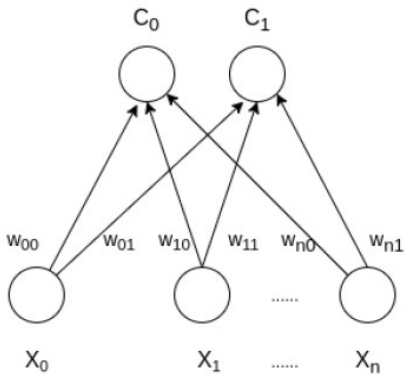


Let's convert each sample to an array with length = 63. In this question number of training samples is 18(m), each sample has 63 (9*7) features (n). According to training set we have 6 different clusters (c). First, it initializes the weights of size (n, C) where C is the number of clusters. Then iterating

over the input data, for each training example, it updates the winning vector (weight vector with the shortest distance (e.g. Euclidean distance) from training example). Weight update rule is given by:

$$W_{ij}(new) = W_{ij}(old) + \alpha\left(x_i - W_{ij}(old)\right)$$

Where alpha is a learning rate (in this question equals to 0.5) at time t, j denotes the winning vector, i denotes the i^{th} feature of training example and k denotes the k^{th} training example from the input data. After training the SOM network, trained weights are used for clustering new examples. A new example falls in the cluster of winning vector. The following figure illustrates the network topology.



Data and parameters:

```
# training data
Train = [
    [1,1,1,1,1,1,1,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,1,1,1,1,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,1,1,1,1,1,1],
    [1,1,1,1,1,1,1,  0,1,0,0,0,0,0,  0,1,0,0,0,0,0,  0,1,0,1,0,0,0,  0,1,1,1,0,0,0,  0,1,0,1,0,0,0,  0,1,0,0,0,0,0,  0,1,0,0,0,0,0,  1,1,1,1,1,1,1],
    [0,0,0,1,0,0,0,  0,0,0,1,0,0,0,  0,0,0,1,0,0,0,  0,0,1,0,1,0,0,  0,0,1,0,1,0,0,  0,1,1,1,1,1,0,  0,1,0,0,0,1,0,  0,1,0,0,0,1,0,  1,1,1,0,1,1,1],
    [0,0,0,1,0,0,0,  0,0,0,1,0,0,0,  0,0,1,0,1,0,0,  0,0,1,0,1,0,0,  0,1,0,0,0,1,0,  0,1,1,1,1,1,0,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1],
    [1,1,1,1,1,1,1,  0,1,0,0,0,0,0,  0,1,0,0,0,0,0,  0,1,0,1,0,0,0,  0,1,1,1,1,1,0,0,  0,1,0,0,1,0,0,  0,1,0,0,0,0,0,  0,1,0,0,0,0,0,  1,1,1,1,1,1,1],
    [0,0,0,1,0,0,0,  0,0,0,1,0,0,0,  0,0,0,1,0,0,0,  0,0,1,0,1,0,0,  0,0,1,0,1,0,0,  0,1,0,0,0,0,1,0,  0,1,1,1,1,1,0,  0,1,0,0,0,1,0,  0,1,0,0,0,1,0],
    [1,0,0,0,0,1,0,  1,0,0,0,1,0,0,  1,0,0,1,0,0,0,  1,0,1,0,0,0,0,  1,1,0,0,0,0,0,  1,0,1,0,0,0,0,  1,0,0,1,0,0,0,  1,0,0,0,1,0,0,  1,0,0,0,0,1,0],
    [1,1,1,1,1,1,0,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,1,1,1,1,0,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  1,1,1,1,1,1,0],
    [0,0,1,1,1,1,1,  0,1,0,0,0,0,1,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  0,1,0,0,0,0,1,  0,0,1,1,1,1,1],
    [1,1,1,1,1,1,0,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,1,1,1,1,1,0,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,1,1,1,1,1,0],
    [0,0,1,1,1,0,0,  0,1,0,0,0,1,0,  1,0,0,0,0,0,1,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,1,  0,1,0,0,0,1,0,  0,0,1,1,1,0,0],
    [0,0,0,1,1,1,1,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,1,0,0,0,1,0,  0,1,0,0,0,1,0,  0,0,1,1,1,0,0],
    [0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,1,0,0,0,1,0,  0,1,0,0,0,1,0,  0,0,1,1,1,0,0],
    [0,0,1,1,1,0,1,  0,1,0,0,0,1,1,  1,0,0,0,0,0,1,  1,0,0,0,0,0,0,  1,0,0,0,0,0,0,  1,0,0,0,0,0,1,  0,1,0,0,0,1,0,  0,0,1,1,1,0,0],
    [0,0,0,0,1,1,1,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,0,0,0,0,1,0,  0,1,0,0,0,1,0,  0,0,1,1,1,0,0],
    [1,1,1,1,1,1,0,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,1,1,1,1,0,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  1,1,1,1,1,1,0],
    [1,1,1,0,0,1,1,  0,1,0,0,1,0,0,  0,1,0,1,0,0,0,  0,1,1,0,0,0,0,  0,1,1,0,0,0,0,  0,1,0,1,0,0,0,  0,1,0,0,1,0,0,  0,1,0,0,0,1,0,  1,1,1,0,0,1,1],
    [1,1,1,0,0,1,1,  0,1,0,0,0,1,0,  0,1,0,0,1,0,0,  0,1,0,1,0,0,0,  0,1,1,0,0,0,0,  0,1,0,1,0,0,0,  0,1,0,0,1,0,0,  0,1,0,0,0,1,0,  1,1,1,0,0,1,1]
   ]

# test data
Test1 = [1,1,1,1,1,0,0,  1,0,0,0,0,1,0,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,0,0,0,0,0,1,  1,0,0,0,0,1,0,  1,1,1,1,1,0,0]
Test2 = [1,1,1,1,1,0,0,  0,1,0,0,0,1,0,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,0,1,  0,1,0,0,0,1,0,  1,1,1,1,1,0,0]


# input data size (m,n)
m = 18 # number of training examples
n = 63 # number of features 9 * 7
c = 6 # number of clusters
```

Implemented Model:

```
class SOM :

    # Function here computes the winning vector
    # by Euclidean distance
    def winner( self, weights, sample ) :

        D0 = 0
        D1 = 0
        D2 = 0
        D3 = 0
        D4 = 0
        D5 = 0

        for i  in range( len( sample ) ) :

            D0 = D0 + math.pow( ( sample[i] - weights[0][i] ), 2 )
            D1 = D1 + math.pow( ( sample[i] - weights[1][i] ), 2 )
            D2 = D2 + math.pow( ( sample[i] - weights[2][i] ), 2 )
            D3 = D3 + math.pow( ( sample[i] - weights[3][i] ), 2 )
            D4 = D4 + math.pow( ( sample[i] - weights[4][i] ), 2 )
            D5 = D5 + math.pow( ( sample[i] - weights[5][i] ), 2 )

            #if D0 > D1 :
                #return 0
            #else :
                #return 1
            print('Distances: {}'.format(np.array([D0,D1,D2,D3,D4,D5])))
            return np.argmax(np.array([D0,D1,D2,D3,D4,D5]))

    # Function here updates the winning vector
    def update( self, weights, sample, J, alpha ) :

        for i in range( len ( weights[0] ) ) :
            weights[J][i] = weights[J][i] + alpha * ( sample[i] - weights[J][i] )

        return weights
```

Training procedure:

```
weights = np.random.rand(n,c)

# training
ob = SOM()

epochs = 3
alpha = 0.5

for i in range( epochs ) :
    for j in range( m ) :

        # training sample
        sample = Train[j]

        # Compute winner vector
        J = ob.winner( weights, sample )
        print('Winner cluster for sample {0} is : {1}'.format(m,J))

        # Update winning vector
        weights = ob.update( weights, sample, J, alpha )


J = ob.winner(weights, Test1)

print( "\n\nTest1 Sample s belongs to Cluster : ", J )


J = ob.winner(weights, Test2)

print( "Test2 Sample s belongs to Cluster : ", J )

print( "\n\nTrained weights :\n\n", weights )
```

Training log:

```
Distances: [0.44444461 0.44444486 0.11129654 0.44444477 0.1112725  0.11122868]
Winner cluster for sample 18 is : 1

Distances: [0.44444461 0.11111121 0.11129654 0.44444477 0.1112725  0.11122868]
Winner cluster for sample 18 is : 3

Distances: [0.11111103 0.44444424 0.44407382 0.44444428 0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 3

Distances: [0.11111103 0.44444424 0.44407382 0.11111107 0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 1

Distances: [0.44444461 0.44444455 0.11129654 0.44444453 0.1112725  0.11122868]
Winner cluster for sample 18 is : 0

Distances: [0.44444436 0.11111106 0.44407382 0.11111107 0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 0

Distances: [0.44444449 0.44444455 0.11129654 0.44444453 0.1112725  0.11122868]
Winner cluster for sample 18 is : 1

Distances: [0.44444449 0.11111114 0.11129654 0.44444453 0.1112725  0.11122868]
Winner cluster for sample 18 is : 3

Distances: [0.11111109 0.44444439 0.44407382 0.4444444  0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 3

Distances: [0.44444449 0.11111114 0.11129654 0.44444446 0.1112725  0.11122868]
Winner cluster for sample 18 is : 0

Distances: [0.44444442 0.44444439 0.44407382 0.1111111  0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 0

Distances: [0.11111111 0.44444439 0.44407382 0.1111111  0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 1

Distances: [0.11111111 0.1111111  0.44407382 0.1111111  0.44412185 0.4442094 ]
Winner cluster for sample 18 is : 5

Distances: [0.11111111 0.1111111  0.44407382 0.1111111  0.44412185 0.11105235]
Winner cluster for sample 18 is : 4

Distances: [0.11111111 0.1111111  0.44407382 0.1111111  0.11103046 0.11105235]
Winner cluster for sample 18 is : 2

Distances: [0.44444446 0.44444447 0.44462981 0.44444446 0.44460578 0.44456199]
Winner cluster for sample 18 is : 2

Distances: [0.44444446 0.44444447 0.11115745 0.44444446 0.44460578 0.44456199]
Winner cluster for sample 18 is : 4

Distances: [0.44444446 0.44444447 0.11115745 0.44444446 0.11115145 0.44456199]
Winner cluster for sample 18 is : 5
```

Output:

```
Distances: [0.44444446 0.44444447 0.11115745 0.44444446 0.11115145 0.1111405 ]
Test1 Sample s belongs to Cluster :  1
Distances: [0.44444446 0.44444447 0.11115745 0.44444446 0.11115145 0.1111405 ]
Test2 Sample s belongs to Cluster :  1
```

Trained weights transposed for better display :

```
[[0.333 0.333 0.667 0.333 0.667 0.667 0.568 0.877 0.687 0.101 0.53  0.028
 0.367 0.761 0.689 0.073 0.594 0.428 0.827 0.409 0.571 0.403 0.497 0.039
 0.776 0.875 0.453 0.457 0.77  0.412 0.13  0.128 0.247 0.492 0.014 0.491
 0.99  0.923 0.528 0.979 0.409 0.313 0.076 0.032 0.237 0.362 0.71  0.98
 0.953 0.826 0.829 0.518 0.174 0.824 0.577 0.306 0.107 0.596 0.408 0.534
 0.659 0.631 0.104]
[0.333 0.067 0.667 0.333 0.667 0.667 0.666 0.888 0.42  0.551 0.179 0.832
 0.151 0.009 0.104 0.415 0.853 0.825 0.478 0.911 0.696 0.111 0.073 0.792
 0.073 0.604 0.974 0.845 0.744 0.418 0.659 0.201 0.114 0.88  0.073 0.323
 0.387 0.102 0.9   0.651 0.037 0.141 0.87  0.524 0.818 1.    0.349 0.436
 0.898 0.42  0.55  0.778 0.454 0.28  0.33  0.648 0.535 0.069 0.296 0.44
 0.893 0.205 0.097]
[0.867 0.067 0.667 0.867 1.    0.667 0.409 0.597 0.937 0.812 0.232 0.283
 0.487 0.45  0.525 0.726 0.7   0.982 0.467 0.583 0.917 0.253 0.199 0.48
 0.212 0.571 0.953 0.309 0.227 0.025 0.047 0.342 0.798 0.768 0.57  0.963
 0.519 0.712 0.007 0.562 0.736 0.708 0.947 0.667 0.201 0.591 0.302 0.702
 0.203 0.569 0.836 0.778 0.424 0.527 0.461 0.895 0.866 0.008 0.386 0.254
 0.111 0.042 0.992]
[1.    0.733 0.667 1.    0.333 0.    0.538 0.441 0.542 0.553 0.386 0.14
 0.668 0.41  0.859 0.06  0.94  0.764 0.839 0.759 0.932 0.853 0.477 0.969
 0.589 0.416 0.732 0.482 0.192 0.252 0.937 0.592 0.195 0.89  0.746 0.646
 0.586 0.929 0.999 0.989 0.795 0.223 0.153 0.052 0.088 0.998 0.422 0.538
 0.34  0.905 0.695 0.618 0.091 0.232 0.876 0.021 0.004 0.962 0.91  0.34
 0.73  0.816 0.447]
[0.867 0.6   1.    0.867 0.333 0.    0.956 0.164 0.785 0.087 0.652 0.864
 0.131 0.854 0.358 0.85  0.677 0.746 0.581 0.811 0.509 0.766 0.133 0.371
 0.965 0.886 0.676 0.353 0.492 0.24  0.437 0.884 0.843 0.005 0.97  0.627
 0.803 0.241 0.306 0.776 0.065 0.115 0.294 0.353 0.301 0.745 0.729 0.923
 0.908 0.184 0.232 0.748 0.604 0.632 0.33  0.174 0.984 0.254 0.192 0.38
 0.395 0.723 0.214]
[0.333 0.867 1.    0.867 0.667 1.    0.688 0.121 0.161 0.719 0.94  0.306
 0.846 0.338 0.855 0.48  0.9   0.495 0.076 0.207 0.034 0.077 0.081 0.182
 0.506 0.303 0.182 0.029 0.469 0.191 0.952 0.962 0.056 0.202 0.42  0.737
 0.595 0.663 0.079 0.192 0.302 0.496 0.619 0.342 0.101 0.571 0.753 0.205
 0.909 0.68  0.665 0.643 0.85  0.697 0.783 0.955 0.895 0.215 0.743 0.295
 0.449 0.857 0.904]]
```

Problem 3

Explain the applications of an SOM network. Furthermore, state that why an SOM can be used in those areas.

The Self Organizing Map is one of the most popular neural models. It belongs to the category of the competitive learning network. The SOM is based on unsupervised learning, which means that is no human intervention is needed during the training and those little needs to be known about characterized by the input data. We could, for example, use the SOM for clustering membership of the input data. The SOM can be used to detect features inherent to the problem and thus has also been called SOFM the Self Origination Feature Map. Despite de simplicity of the SOM algorithm, it can and has been used to perform many different tasks, the most common of which are:

1. **Clustering (k-means type clustering)** – This is probably the most common application of SOM, albeit probably not the best. In this context, the SOM is used as an alternative to k-means clustering, i.e., given a fixed number k of clusters, the SOM will partition the available data into k different groups. As an example, we may want to divide customers into 4 different groups according to their characteristics, for marketing purposes. The main advantage of o SOM in this case is that it is less prone to local minima than the traditional k-means clustering algorithm, and thus can act as a good initialization algorithm for that method. In fact, it can substitute k-means altogether, for as noted in, the final stages of the SOM training algorithm are exactly the same as the k-means algorithm. An extra bonus of the SOM algorithm is that the clusters obtained are topologically ordered, i.e., similar clusters are (usually) grouped together.
2. **Exploratory data analysis and visualization** – This is, arguably, the most important application of SOM. In this case, the SOM is used as a non-linear projection algorithm, mapping n-dimensional data onto a 1 or 2 dimensional grid. The SOM can thus be an alternative to PCA projections, Principal curves, or Multi-dimensional Scaling (MDS) algorithms such as Sammon Mappings. Different projection algorithms perform different trade-offs when mapping from high to low dimension, since in all but the most trivial cases some information will be lost. The main advantage of projecting multidimensional data onto 1 or 2 dimensions is that we can easily visualize the data in these dimensions. From this visualization we can identify outliers (data points that are far from other data), identify data that is similar to a given reference, or generally compare different data. If we project data onto 1 dimension, we may then plot histograms, and thus identify "natural" clusters of data. A similar result may be obtained with a technique closely related to SOM called U-Matrix that can be extended to visualize what can loosely be interpreted as a 2- dimensional histogram.
3. **Ordering of multidimensional data** – This type of application makes use of the topological ordering of the SOM to organize a given set of data vectors according to some criteria. As an example, a 1-dimensional SOM can be used to solve the well-known travelling salesman or related problems. Another interesting use of this ordering capacity of a SOM is to create color palettes from pictures.
4. **Supervised data classification** – The SOM is not meant to be a classifier, and a related technique called LVQ-Linear Vector Quantization is best suited for this task. However, just like the centroids obtained by a k-means algorithm, a SOM may also be used supervised classification by labeling the neurons (or units) with the classes of the data that are mapped to it.
5. **Sampling** – The units of a SOM have a probability distribution that is a function of the probability distribution of the data used for training. Generally, the SOM will over

represent regions of the input space that have a low density, but that is many times an advantage since it helps detect outliers and novel data patterns.

6. **Feature extraction** – Since the SOM performs a mapping from a high dimensional space to a low dimensional one, it may be used for feature extraction. In the simpler case, the new features are simply the coordinates of the mapped data point. This is one of the few cases where SOMs with a dimension greater than 2 are easy to use.
7. **Control and/or data sensitive processing** – A SOM can be used to select, based on available data, the best model, controller, or data processor for a given situation. The main idea behind this type of application is that instead of designing a rather complex controller, multiple simple controllers may be used, each one tuned to a particular type of situation. During the training of the SOM the input data is partitioned into various Voronoi regions, and each of these is used to train or define the parameters of a different controller.
8. **Data interpolation** – When using the SOM to interpolate data, the output space of the SOM will have the same dimension as the input space, but since the units are ordered on a regular grid, that grid provides a locally linear interpolator for the data.

References

1) Olawoyin, R., Nieto, A., Grayson, R. L., Hardisty, F., & Oyewole, S. (2013). Application of artificial neural network (ANN)–self-organizing map (SOM) for the categorization of water, soil and sediment quality in petrochemical regions. Expert Systems with Applications, 40(9), 3634-3648.

2) Behbahani, Soroor & Motie Nasrabadi, Ali. (2009). Application of SOM neural network in clustering. Journal of Biomedical Science and Engineering. 2. 637-643. 10.4236/jbise.2009.28093.

3) https://medium.com/machine-learning-researcher/self-organizing-map-som-c296561e2117

4) Lobo, V. J. (2009). Application of self-organizing maps to the maritime environment. In Information Fusion and Geographic Information Systems (pp. 19-36). Springer, Berlin, Heidelberg.

5) https://www.geeksforgeeks.org/self-organising-maps-kohonen-maps/