Visual Perception

Human Psychophysics Project Report

Master in Computer Vision

UNIVERSITE DE BOURGOGNE

Centre Universitaire Condorcet - UB, Le Creusot

14 April 2017

Submitted to: Prof. Elizabeth Thomas

Submitted by: Mohit Kumar Ahuja

Project 1:

1) Construct a Kohonen network in order to carry out the classification of the vectors

(1 1 0 0), (1 0 0 0), (0 0 0 1), (0 0 1 1)

Solution: For this I made a function named kohnen_proj_1 that you can find in the attachment. This function takes training input (W) as well as test input (Z). From training input it automatically makes one cluster of vectors (1 1 0 0) and (1 0 0 0) another cluster of (0 0 0 1) and (0 0 1 1).

{To know more about this function you can also write "help kohnen_proj_1" in command window}

The Weight vector is initialized by taking mode of random numbers (Note: mode is taken to make all the values positive).

```
weights = mod(randn(Output_Vectors,Input_Vectors), 1)
```

Where `Input_Vectors = size(W,1)` to make it flexible in terms of the size of the input vector.

The number of iterations I used are 50 because of high learning rate, it converged in less number of iterations. Then I calculated Euclidian Distance [d(k)] of both weight vectors i.e. [d(1) and d(2)]. And update them in every iteration by updating the closest weight vector while leaving other unchanged.

And also one major issue I faced was the d(1) and d(2) vector keeps shifting i.e. sometimes d(1) corresponds to cluster 1 and sometimes d(2) represents cluster 1 and same for cluster 2. This problem I faced because I used random numbers for initialization of weight vector which results in shifting of d(1) and d(2) multiple run's. So, to solve this problem in project 1 I took the mean of First Cluster and then compared it with both the Weight vectors then which one is closer is assigned the value of that cluster. By doing this we can identify and assign the values of which weight vector represents which cluster.

2) Once the training is completed carry out a test with the vectors

(0, 0, 0, 0.9), (0, 0, 0.8, 0.9), (0.7, 0, 0, 0), (0.7, 0.9, 0, 0)

Solution: As expected, the vectors (0, 0, 0, 0.9) and (0, 0, 0.8, 0.9) falls in one class while the vectors (0.7, 0, 0, 0) and (0.7, 0.9, 0, 0) falls in second class by calculating the Euclidian Distance of each test vector with weight vectors. And the minimum distance vector wins and assigns its class.

The Results are shown in Figure (1).

```
Learned Weight matrix is:

weights =

         0         0    0.5000    1.0000
    1.0000    0.5000         0         0

Test Data for project 1:

Z =

         0         0         0    0.9000
         0         0    0.8000    0.9000
    0.7000         0         0         0
    0.7000    0.9000         0         0

Results:
[0          0          0        0.9 ] This Vector Belongs to Class 2
[0          0        0.8        0.9 ] This Vector Belongs to Class 2
[0.7          0          0          0 ] This Vector Belongs to Class 1
[0.7        0.9          0          0 ] This Vector Belongs to Class 1
```

Figure (1): Result of Test data in Project 1

Project 2:

1) Train your Kohonen networks using the training data set that I am sending. As to be expected the 'control.txt' file contains control data and the 'patient.txt' contains patient data! Each line corresponds to the data (time series) coming from one subject. The time series is made up of the displacements of markers placed on the joints of subjects. There are ten subjects in each file. Of course, the same markers are used for all subjects.

Solution: For project 2 and 3 I made a function named `kohnen_proj_2and3` that you can find in the attachment. This function takes training input (W, Z) as well as test input(Y) (Note: W represents Control Data and Z represents Patient Data). So, I concatenated matrix W and matrix Z so as to make a single matrix X. And from training input X it automatically makes one cluster for control data and one cluster for patient data.

{To know more about this function you can also write "help kohnen_proj_2and3" in command window}

2) The dimensions of your input and weight vectors for the Kohonen network then have to be adjusted accordingly.

Solution: The Weight vector is initialized by taking mode of random numbers (Note: mode is taken to make all the values positive).

```
weights = mod(randn(Output_Vectors,Input_Vectors), 1)
```

Where `Input_Vectors = size(X,1)` to make it flexible in terms of the size of the input vector. As explained X is concatenated matrix of control data and patient data.

The number of iterations I used are 50 because of high learning rate, it converged in less number of iterations. Then I calculated Euclidian Distance [d(k)] of both weight vectors i.e. [d(1) and d(2)]. And update them in every iteration by updating the closest weight vector while leaving other unchanged.

The same problem I faced in this project was the shifting of d(1) and d(2) vector i.e. sometimes d(1) corresponds to cluster 1(patient) and sometimes d(2) corresponds to cluster 1(patient) and same for cluster 2. This problem was faced because of random numbers initialization of weight vector which results in shifting of d(1) and d(2) multiple run's. So, I solved this problem in project 2 by two methods:

First Method: In first method I calculated the row mean vector of the first training cluster i.e. control data. And by calculating the Euclidian Distance between both weight vectors, the minimum among both wins and then I assigned the matched weight vector with cluster (control), so by doing this the second cluster has been automatically assigned the second weight vector.

Second Method: In second method I calculated the row mean values (not vectors) of first weight vector [d(1)] and second weight vector[d(2)] which comes to be 1.25 for patient weight vector and 1.36 for control weight vector in every iteration. Then by comparing in every run if the first row of weight vector shifts to second one I make them shift again to make the weight vector constant every time.

That is in every run, if first the first weight vector corresponds to Control Data, nothing will change but if it corresponds to Patient data, the row weight vectors will shift to each other's place and again the first weight vector will correspond to Control Data. By using any of these methods we can compute an accurate result.

{Note: To check any of the method please uncomment the one you want to apply and comment the other one. You will get the same output.}

Project 3:

1) The text file (mohit.txt) contains the data from 4 subjects. You have to identify which ones are patients or controls based on the Kohonen network that you have already constructed.

Solution: The text file was loaded into 'Y' and then as we already got weight vectors from training. i.e: d(1) and d(2). So, by finding the Euclidian Distance between both the weight vectors, it was found that the test data of 4 subjects was from cluster **"CONTROL"**.

I have replaced the 3rd row vector of test data with patient row vector to cross check my code and it was showing the accurate result. i.e. for first two rows and forth row vector it was showing control and for third row vector it shown patient. For testing I saved that file as "Copy_of_mohit.txt". Please uncomment second 'Y' to see the output of that matrix.

The Results are shown in Figure (2).

```
                Project 2 and 3:

Control.txt and Patient.txt is given as training data

Test Data for Project 3:
mohit.txt with 4 row vectors

Results:
Test Vector 1 Belongs to Control
Test Vector 2 Belongs to Control
Test Vector 3 Belongs to Control
Test Vector 4 Belongs to Control
```

Figure (2): Result of Test data in Project 3

Bonus points: What are some differences if any between a bio-inspired algorithm like the Kohonen SOM and two other well-known similar clustering algorithms – the kmeans and k nearest neighbour algorithm?

Solution:

Sr. no:	Kohonen SOM	K_Means	K Nearest Neighbour
1	SOM's provide a more robust learning	k-means obviously is not the best choice, because it is sensitive to initialization	
2	In SOM clusters are formed geometrically	In k-means clusters are formed through centroid and cluster size	It saves a lot of time by using locality-sensitive hashing. It allow to compute k-nearest neighbors very efficiently without calculating lots of distances.
3	SOM are less sensitive to the noise present in the dataset	k-means is more sensitive to the noise present in the dataset	
4	SOMs fall more into the unsupervised learning category	k-means is either unsupervised or semi-supervised	is a subset of supervised learning
5	SOMs are better as you can always do a check on the output and remove redundant variables accordingly.	K-means is that you need to specify the value of k. Kmeans care about global	KNN cares about local
6	It is a classification algorithm	It is a Clustering algorithm	It is a classification algorithm