Artificial neural networks - Exercise session 3

Unsupervised learning: PCA and SOM

Cristina Luna

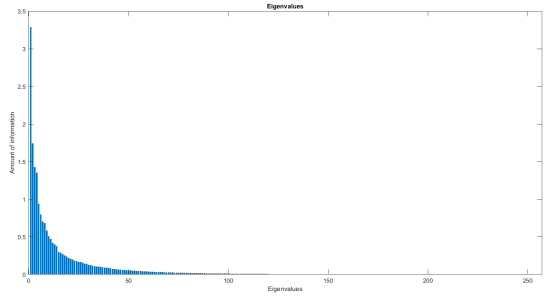
May 6, 2018

1 Principal Components Analysis

PCA is a technique frequently used for mapping data into a lower dimension in order to reduce complexity and maintain the maximum amount of information. For this purpose, the highest eigenvalues are detected and used for extracting their eigenvectors that map the original data into a smaller dimension, keeping most of the original information.

From the experiments done in this project, we can say that it is a really useful tool when the dimensions of the data maintains some correlation because in few eigenvalues we have most of the information, so it is easier to reduce the number of parameters that define each data sample and reconstruct the sample almost without errors using just few eigenvectors associated to these eigenvalues. On the contrary, if data doesn't have correlation between their components, as in random numbers, all the eigenvalues keep a large amount of information, so removing one means increasing a lot the reconstruction error, so PCA is not an effective method for these situations.

The second task of this project is the application of PCA for the reconstruction of digits images using less parameters than the number of pixels that conforms the image, in our case, 256 pixels.



In the left image, we can see the information that eigenvalues contain.

If we divide these values by the total cumulated information, with 45 of them we would have the 90% of the information and with 69, the 95%, so using higher number of eigenvalues doesn't give much more information, as we can see in the figure too.

The first 4 eigenvalues maintain almost the 40% of the total information, for this reason, in the reconstructed images it is easy to distinguish the number even using just 1 of these components. The pictures have been extracted from the 2nd,10th and 50th digit images in the datasets

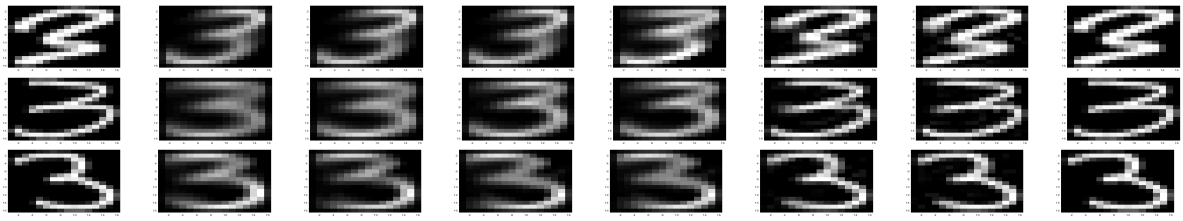
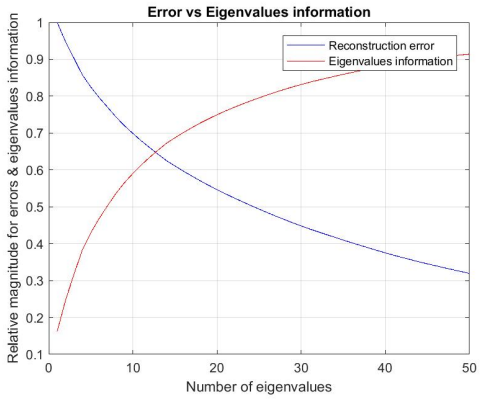


Figure 1: Original image and reconstructed images for k=1,k=2,k=3,k=4,k=45, k=70, k=256

For the case of using all the pixels, the expected error value should be 0, however it is $4.44 * 10^{-16}$, so the most probable explanation is that during the calculation of the covariance matrix some numbers were truncated and some decimals were lost due to these floating points operations.

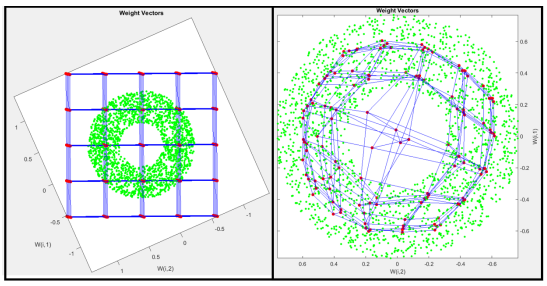


Finally, when we compare the reconstruction error with the number of eigenvalues selected, there is a clear inverse relationship between these 2 values.

In the following plot, we have normalized eigenvalue information and RMSE error dividing them by their maximum values, respectively, in order to see the clear correlation between these 2 parameters and how getting higher number of eigenvalues reduces the error.

2 Self-Organizing Maps

SOM are a technique employed for unsupervised learning that tries to copy the distribution of some data points in the space by using some prototype vectors that have lower dimensions than the initial data, and thanks to these prototypes vectors, it is able to split the space into regions or clusters. Typically, the projections are in a 2 or 3-dimensional planes, thus we can analyze the training in an easy way.



At the beginning, SOM algorithm distribute the prototype vectors uniformly according to a specific topology (grid, hexagons or random) having all the data points inside of this topology and when we perform the training, some areas get a higher number of vectors accordingly to the number of data points in that part, so areas with a higher density of dataset points will have more prototype vectors.

For the second part, we used Iris dataset for clustering. This dataset contains as parameters the length and width of sepals and petals of 3 different Iris species, so we are dealing with a 4-D dataset.

In order to study the optimal combination of parameters, we performed 3 tests, the first one based on all the possible combinations of distances and grids for 10,50,100,200,300 and 400 epochs using as grid size [3 1] due to its good results in previous experiments and because the number of classes of the dataset is 3 and is the more evident first reduction of dimensions.

The second experiment will be based on the grid size trying the following combinations: [2 1],[1 2], [3 1],[1 3], [4 1], [1 4] , [5 1],[1 5], [2 2], [3 2] and [2 3] for the best distance-topology combination found previously in the test1 because it could be interesting to se the performance of the algorithms when we use higher dimensions and lower ones.

Finally, the third experiment was done using the best results obtained from these other 2 experiments. As usual, the number of running of the same configuration was 10.

From the experiment1, the most of the combinations gave similar results except for the case of using randtop grid whose ARIs values were lower than with the others one. If we had to choose one of the configurations as the best one it would be the combination of hextop grid with mandist distance (ARI=0.7288). Regarding the number of epochs, the best values obtained were for 200 epochs, if we use more, results are almost constants but the training time increases.

From experiments 2, the results are the expected ones, when we use 3 neurons (grid sizes of [1 3] or [3 1] the best ARI values are reached because they are the best creating the 3 clusters that represents the 3 different classes of the dataset. Other results show that not bad ARIs are got with 4 neurons configurations(ARI=0.64) although when we increase this number, the ARI decreases because we start to have more clusters than the needed for this problem.

In the following picture we can see how the configuration that uses [1 3] as grid, and hextop-linkdist as topology gives the best vectors positions and a comparison with the classes of the dataset. (This figure is just for illustrative purposes because in an unsupervised learning problem we won't have the labels of the classes)