PROGRAM DESCRIPTION

In this computer experiment we will design an RBF network

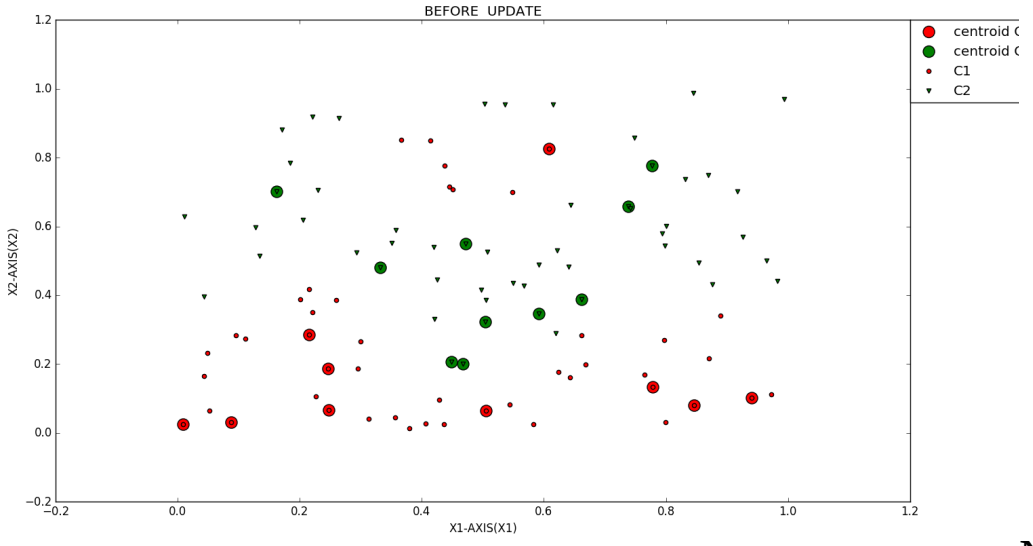
We start creating 100 points x1….x100 independently and uniformly at random on $[0,1]^2$. These 100 point,represented by 2 components x1 and x2 will be our input pattern.

We can now calculate the labels for each point xi using the following formula:

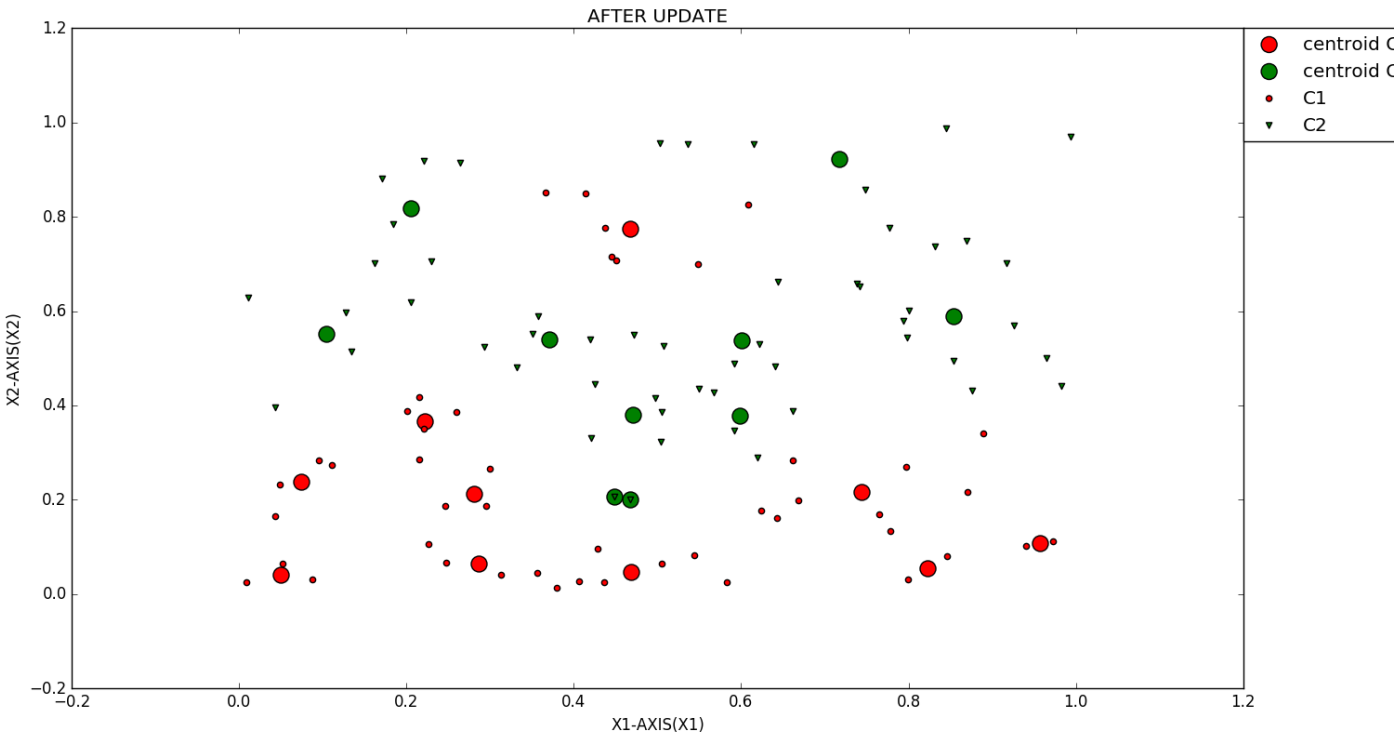
$$d_i = \begin{cases} 1, & x_{i2} < \frac{1}{5}\sin(10x_{i1}) + 0.3 \text{ or } (x_{i2}-0.8)^2 + (x_{i1}-0.5)^2 < 0.15^2 \\ -1, & \text{otherwise} \end{cases}$$

In this way we obtain two classes C1, that contains the points whose output is 1 and C2 that contains the points whose output -1.

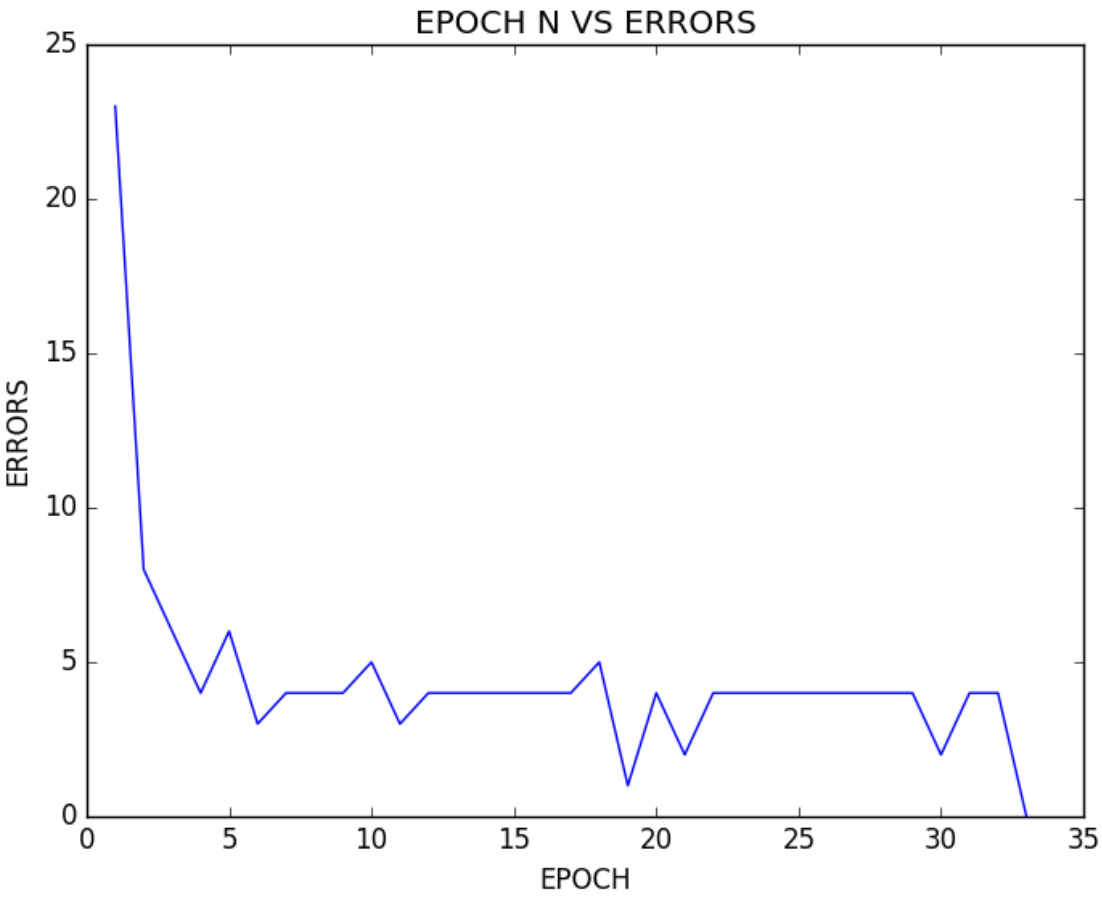
Now the goal is to design an RBF network g(x) with m=20 centers.
We start by running the K-Means algorithm for 10 centers for class C1 and we do the same for class C2,with 10 centers again.
At the beginning of the K-Means algorithm we initialize 10 centroids for each class, choosen at random between the points inside 2 classes.
The result of the centroids initialization is the following:



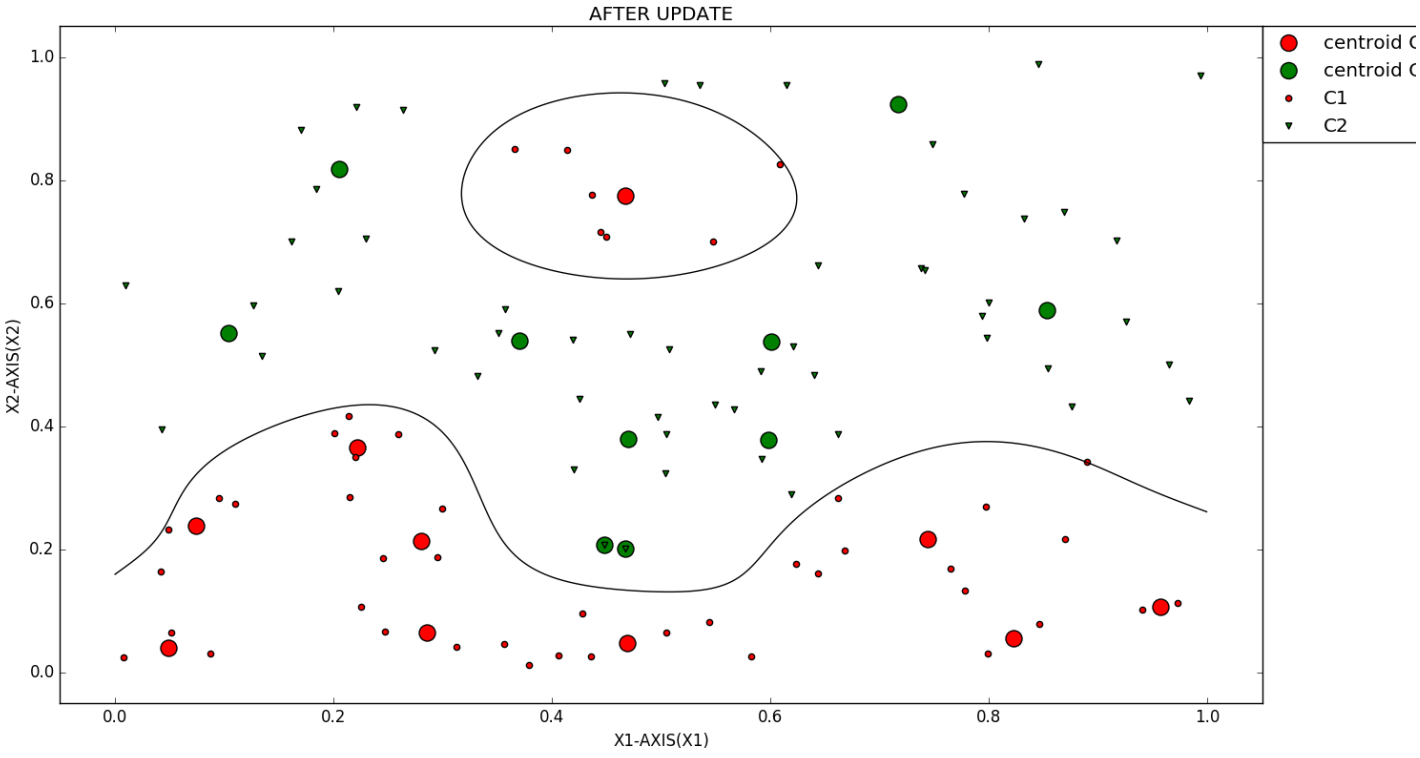
Now we run the kmeans algorithm, and at the end we obtain the following centroids:



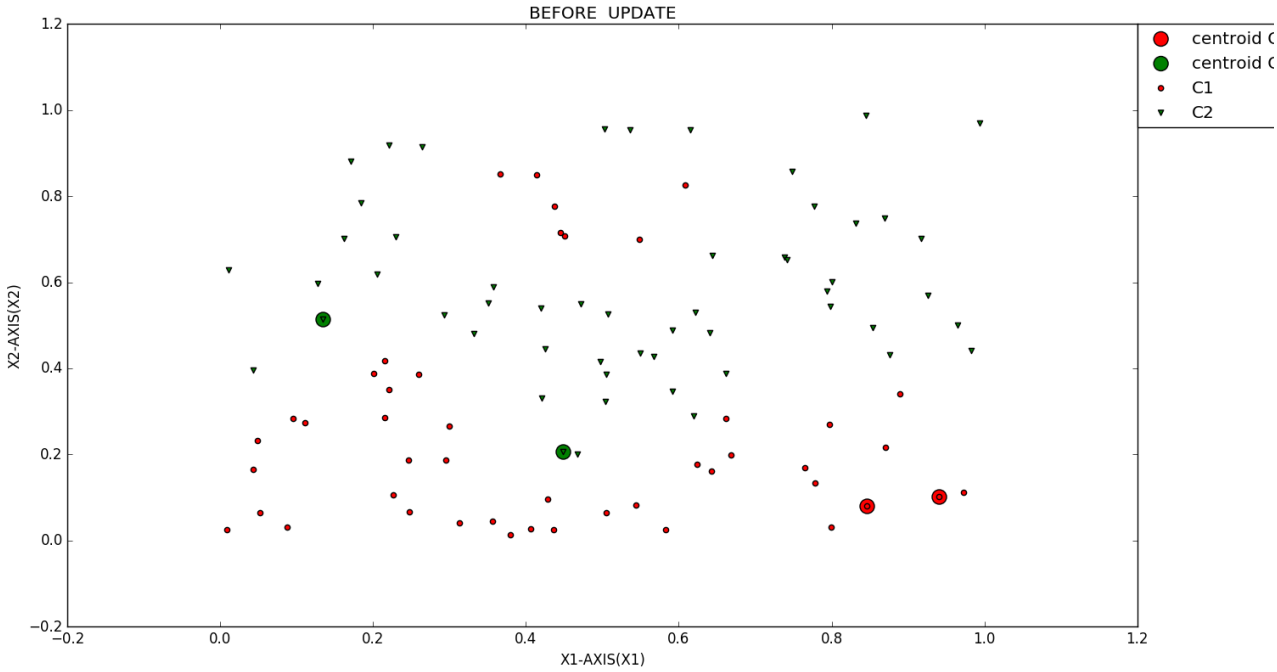
We can now run the PTA algorithm to find the weights $w_1,\ldots w_{20}$ and the bias. In order to do that we initialize the weights unifromly at random betweenn -1 and 1,the bias to 1 and we switch the label of all the points with class -1 to 0.
We then choose an appropriate eta, in this case i choosed eta=1 since it allows a quite fast convergence.
The input of the PTA algorithm, so the “x”, will be the output of the RBF function that we have chosed, in particular a gaussian RBF function with sigma equal to 0.1.
For every point x passed to the function we will obtain as output a vector of 20 elements that we will use as input for the PTA. We also add to this vector another element equal to 1 that will represent the bias.
We now choose an activation function that we can use to predict the label y of each point, in particular we choose a step activation function that will return 1 when the argument passed to it is >=0 and will return 0 otherwise.
Now we can run the PTA until convergence.
The result in terms of number of epoch and number of errors using eta=1 is the following:



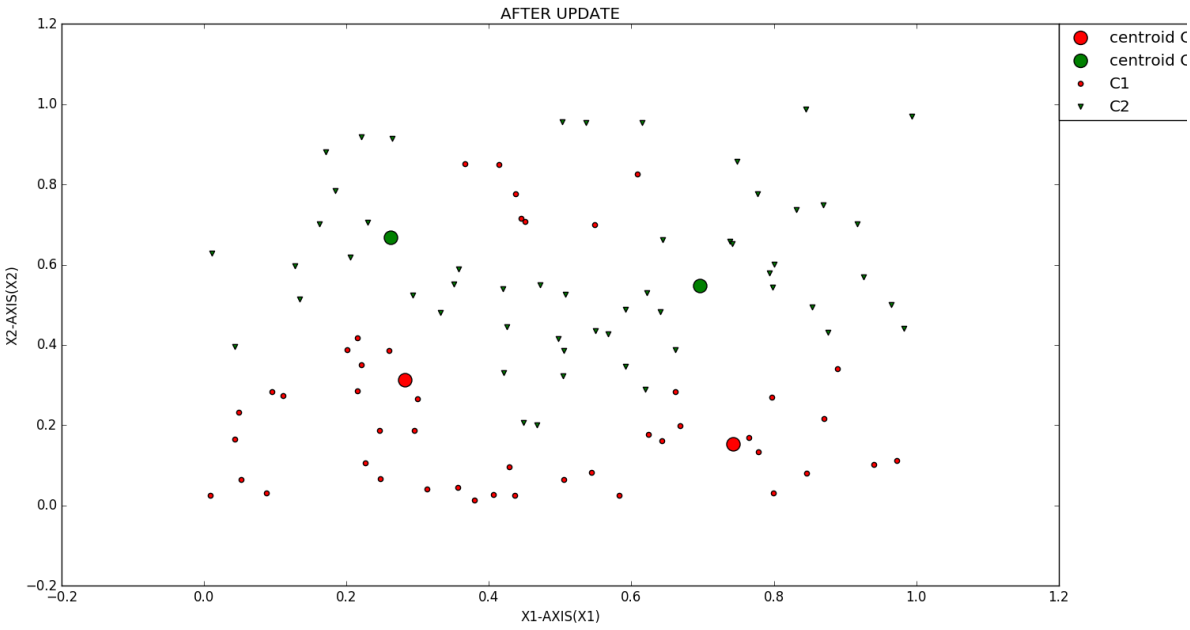
Now that we have found the weights that can perfectly separate the two classes we can use them to calculate the g(x) function.
We can now plot the decision boundary {x: g(x)=0}:



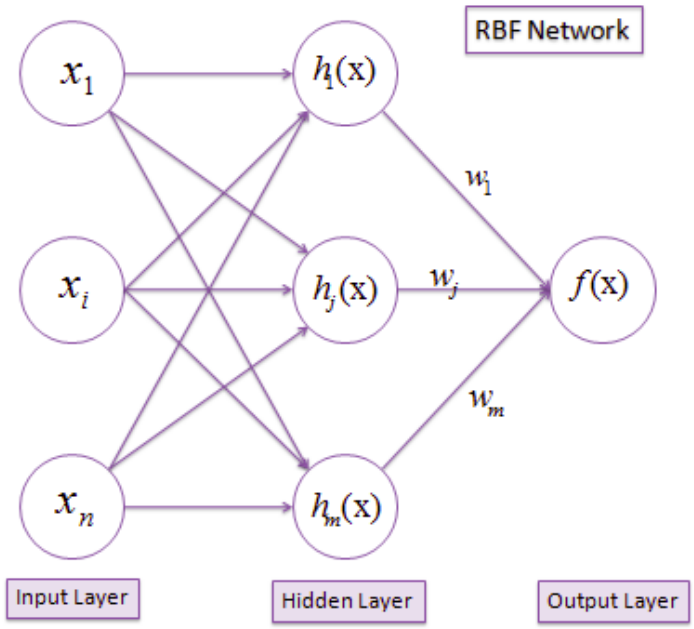
Now we repeat the previous step using a different number of clusters, in particular using a total of 4 clusters. Again we use half of the centers for one class and the the other half for the other.
When we initialize the centroids of kmean algorithm the rsult is the following:



And after the end of the K-means we have:



In this case the PTA algorithm will not converge,since it is not possible to separate the 2 classes perfectly. This happens because higher values of centroids means an higher number of neurons in the hiddden layer, which enables a more complex decision boundary but also means more computations to evaluate the network. In this exercise we can see that with a total of 20 centroids we can build a decision boundary that is complex enough to separate perfectly the 2 classes, while with 4 this boundary is not precise,and for this reason the number of missclassified example will not be zero.
This is the a general RBF network:



$$f(x) = \sum_{j=1}^{m} w_j h_j(x)$$

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

Where h(x)is the Gaussian RBF function that we have used, f(x) is what we have called g(x) and the weights w1,...wm are the weights that we have determined through the PTA algorithm.