Kohonen Self Organizing Map- Project : 4

Atsushi oba (s1230128), Hitoshi Ikuta (m5221106)
Chowdhury Md Intisar (m5212106), Yunosuke Teshima (m5221151)

May 15, 2018

1 Self Organizing Network

When a self-organizing network is used, an input vector is presented at each step. The network tries to represent the input vector of higher dimention to a lower dimentional space. Thus each new input tries to learn the parameter to adapt. The best-known and most popular model of self-organizing networks is the topology-preserving map proposed by Teuvo Kohonen, hence Kohonen self organizing map. If an input space is to be processed by a neural network, the first issue of importance is the structure of this space. Kohonen networks learn to create maps of the input space in a self-organizing way. An image below depicts the network learning the function f to map the input space to output space.

2 Learning Algorithm

Let us consider, an n-dimensional space which will be mapped with the help of m-kohonen units or prototype vectors. Each unit becomes tne n-dimentional input x and computes the exciation or firing rate. The n-dimentional weights $w_1, w_2, .., w_m$ are the computational neurons. The objective of each unit or neuron is that, it learns to specilize on different regions of input space.Thus when an input from such a region is fed into the network, the corresponding unit should fire with maximum excitation. During the training the unit which

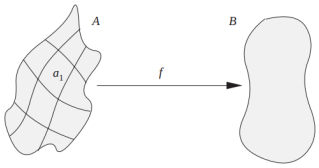

Figure 1: Mapping of function from input space A to output space B. Here the region a is selected.

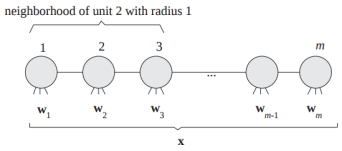

Figure 2: Learning of the prototype

fires the most is selected using the winner takes it all algorithm. Then, the winner weight is updated as follows till it converges to the input.

$$w_m^{k+1} = w_m + \alpha(x - w_m^k) \tag{1}$$

$$w_i = w_i + \eta\phi(i,k)(\xi - w_i) \tag{2}$$

Both of the equation 1 and equation 2 can be used to update the weights. Equation 2 takes into amount its neighbour neurons. Thus a few neighborhood function $\xi(i,k$ have been introduced. In both equation $\alpha and \eta$ represents learning rate.

3 Output of the Program

The given source code have been modified to take 150 samples of input from the ULC machine learning repository. THe changes in the source code is attached as image. The iris data set was fed into the program without any training label. The program have been run with different setting such as with different number of neurons, varied dimention of the input. Output depicted some interesting result showed in Table 1.

Table 1: The output of the program for different setting of M where M is the number of Neurons and is the dimention

M	**Accuracy**
3	66.66%
4	66.66%
5	94%
6	66.66%

It is to be noted that, for each setting the program have been run for 5times and the result have been averaged. Afer few iteration it was suprisingly discovered that the program succesfully finds 3 clusters with high accuracy for the value of $M = 5$. For the other value of M the program results in only 2 clusters which result is accuracy of 66.66%. Also, the weight initialization part has been changed for convergance which is depicted in the Figure 3 and Figure 4. After

```
107
108   for(p=0; p<P; p++) {
109       s0 = 0;
110       for(m=0; m<M; m++) {
111           s = 0;
112           for(i=0; i<I; i++){
113               s += (w[m][i]-x[p][i])*((w[m][i]-x[p][i]));
114           //    printf(" %5f ", x[p][i]);
115               }
116               s = sqrt(s);
117           if(s > s0) {
118               s0 = s;
119               m0 = m;
120           }
121       }
```

Figure 3: Changed the distance metrics to Euclidean

```
64    }
65    for(m=0; m<M; m++) {
66        norm=0;
67        for(i=0; i<I; i++) {
68            w[m][i] = (double)(rand()%100);
69            //w[m][i] = (double)(rand()%10001)/10000.0 - 0.5;
70            // norm += w[m][i]*w[m][i];
71        }
```

Figure 4: Changed weight initialization

changing the distance function from normalized version of the distance metrics we succesfully able to find 3 clusters.

After we have changed the weight initialization method it resulted in early convergence.

4 Research Findings

- The reason for failing to converge to 3 differnt cluster is due to the initilization of the weight or prototype vector.
- A good initialization can converge to proper number of clusters easily.
- Another key factor is the dimention of prototype vector and the number of prototype vector.

Thus the above point greatly influence the outcome and convergence of the prototype vector to the solution.