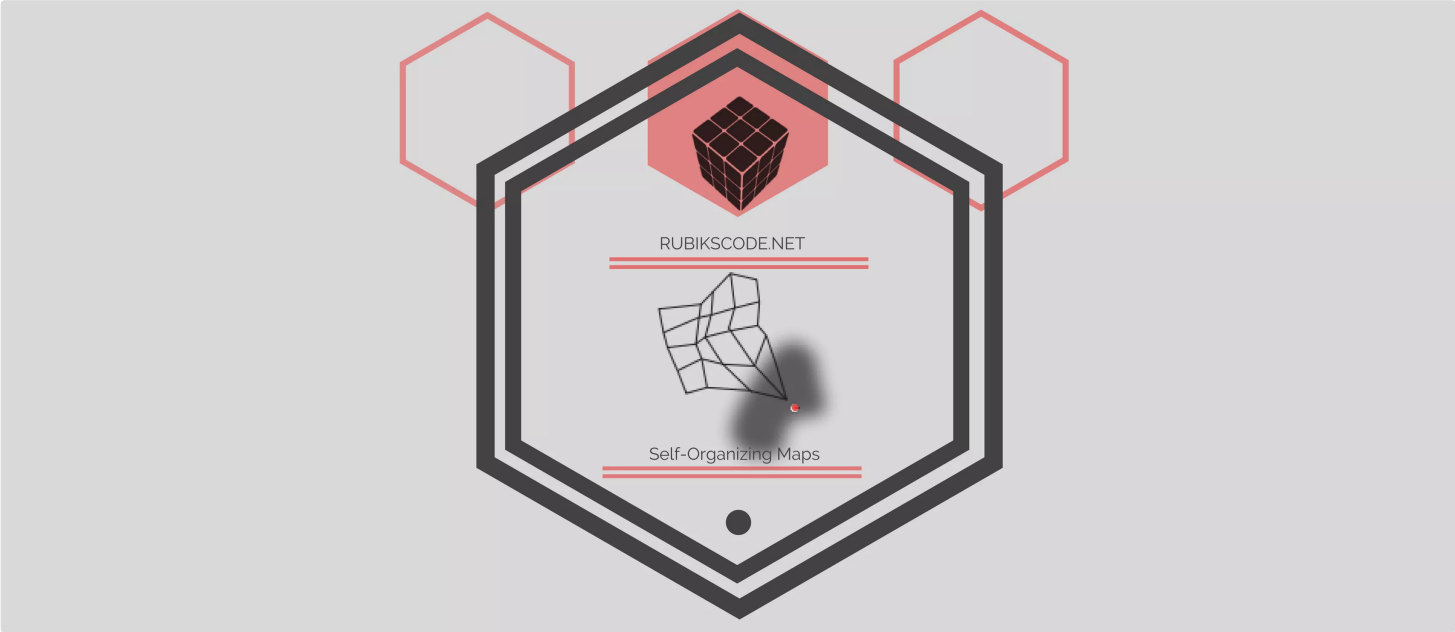
RUBIK'S CODE

Freedom.
Wisdom.
Excellence.

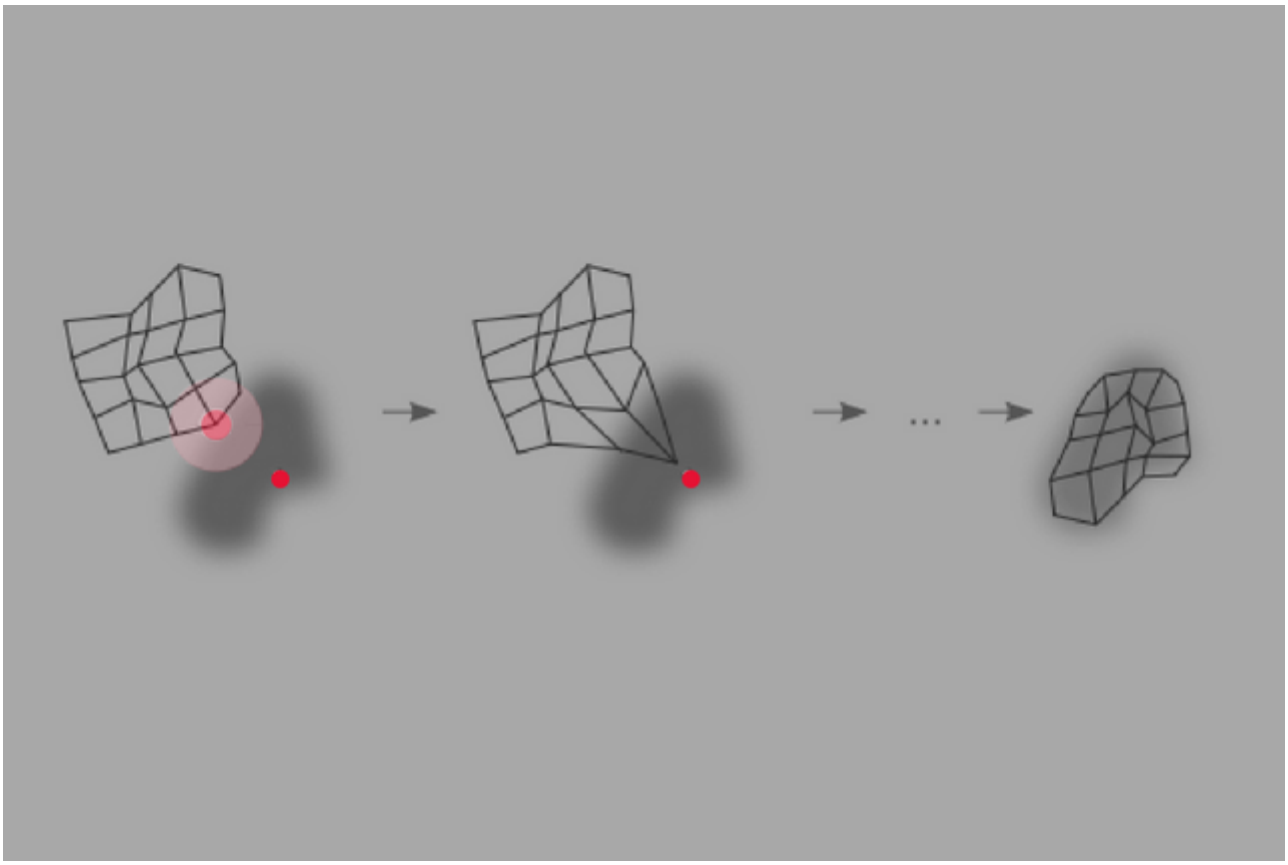
Introduction to Self-Organizing Maps

AUGUST 20, 2018 — 3 COMMENTS



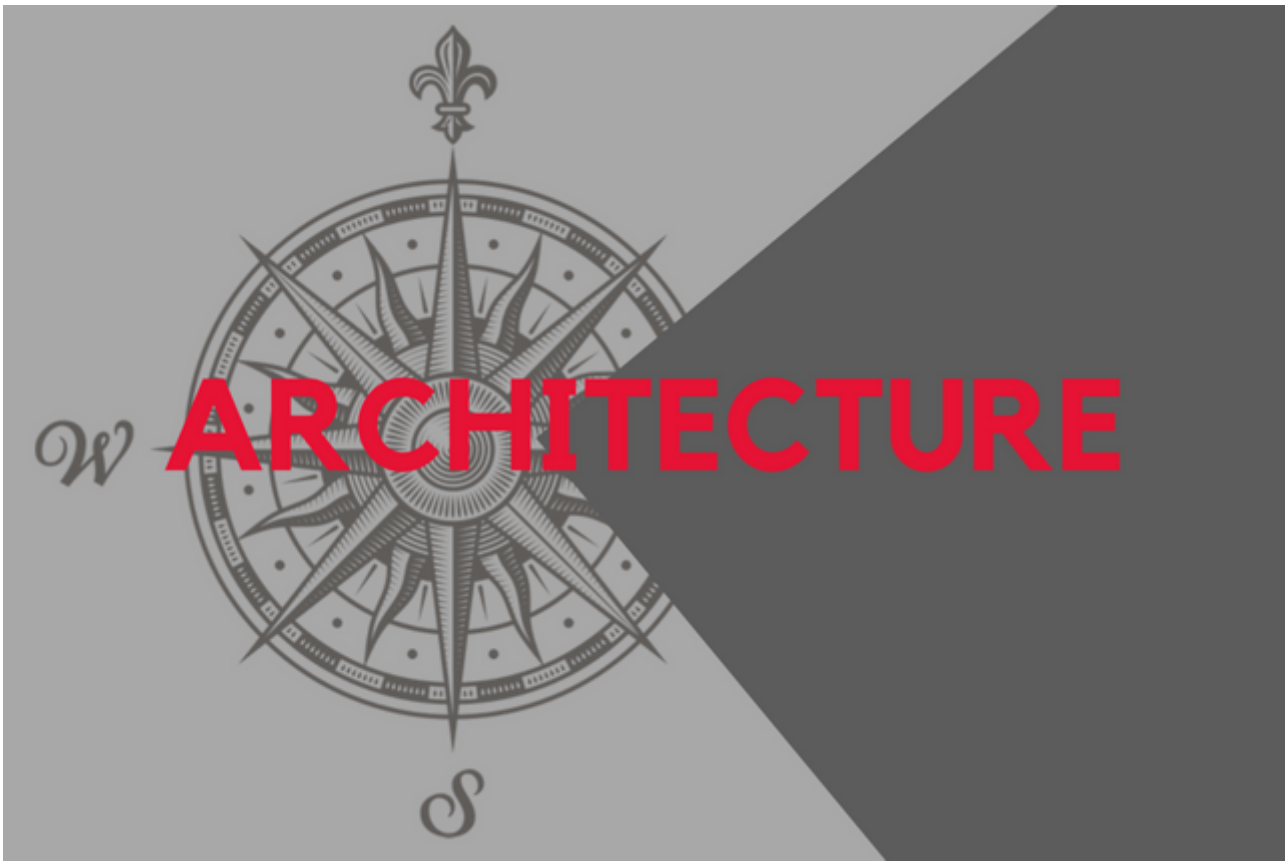
So far in our artificial neural network series, we have covered only neural networks that are using supervised learning. To be more precise, we only explored neural networks that have input and output data available to them during the learning process. Based on this information, this kind of neural networks change their weights and are able to learn how to solve a certain problem. However, there are other types of learning and we are going to explore neural networks that are using these other approaches as well.

Namely, we are going to get familiar with unsupervised learning. Neural networks that use this type of learning get only input data and based on that they generate some form of output. The correct answers are not known during the learning process and neural networks try to figure out patterns in the data on their own. A result of this approach is that we usually have some kind of clustering or classification of data. Self Organizing Maps, or SOMs for short, are using this approach.

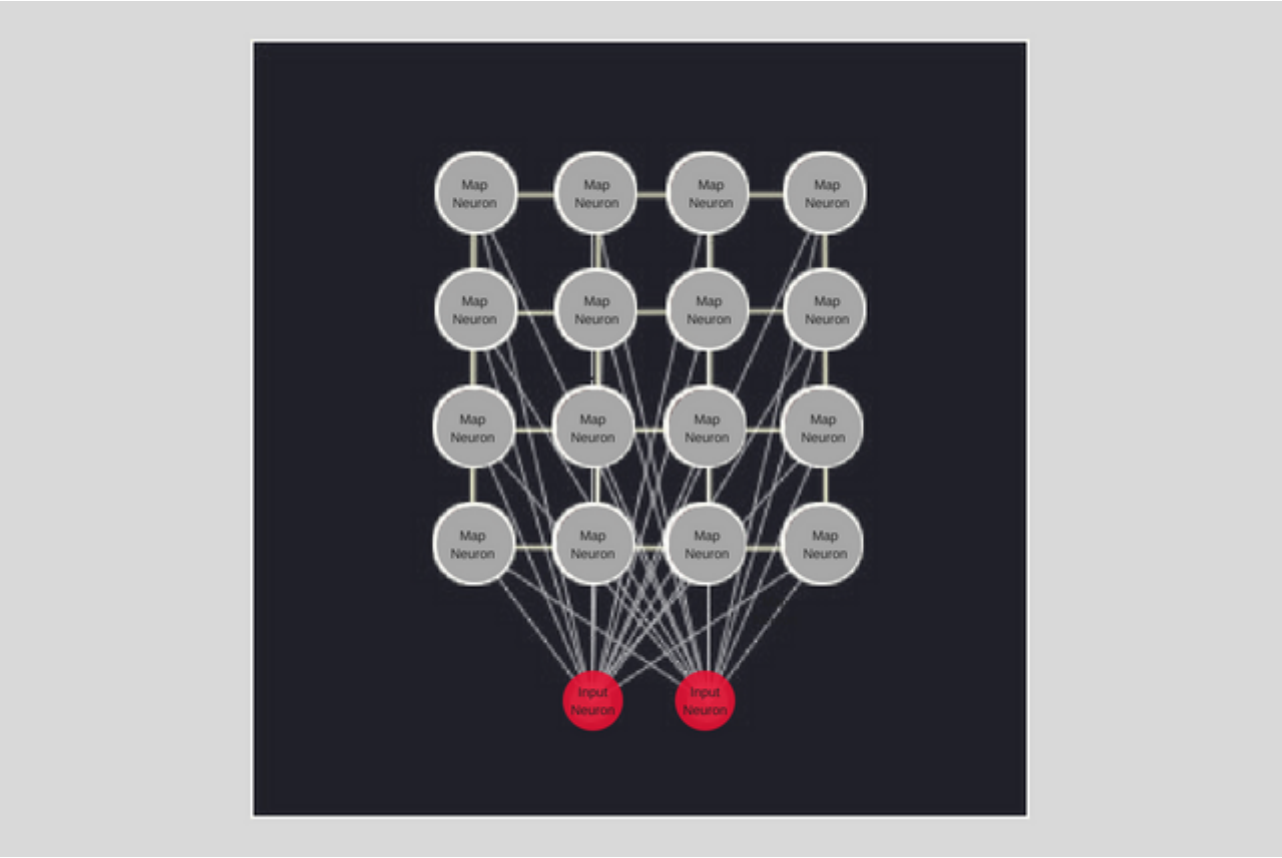


Even though the early concepts for this type of networks can be traced back to 1981, they were developed and formalized in 1992 by Teuvo Kohonen, a professor of the Academy of Finland. In an essence, they are using vector quantization to detect patterns in multidimensional data and represent it in much lower dimensional spaces – usually one or two dimensions, but we will get into more details later. However, it is important to take a fresh perspective on these networks and forget standard neuron/connection and weights concepts. These networks are using the same terms but they have a different meaning in their world. Let's take a look.

Architecture



There is a reason why these networks are called maps. They are sheet-like neural networks, whose neurons are activated by various patterns or classes of patterns in input signals. Take a look at the picture below:



Here we can see a simple self-organizing map structure. We are having two input neurons, which essentially present features in our dataset. This also means that our input data can be represented by three-dimensional vectors. Above them, we can see so-called map neurons. The goal of these neurons is to present data received on input neurons as two-dimensional data. Meaning, that in this example self-organizing map uses unsupervised learning to cluster that three-dimensional data into a two-dimensional representation.

Of course, we can have any number of dimensions in our input data and any number of dimensions for our output (mapping) data. It is important to notice that each map neuron is connected to each input neuron and that map neurons are not connected to each other. This makes every map neuron oblivious to what values their neighbors have.

Each connection still has a weight attached to it, but they are not used in the same way as in feedforward networks. Basically, you can see that weights are now representing the relationship of the mapping neuron with the input vector. Every map neuron can be identified by unique *i, j* coordinate and weights on their connections are updated based on the values on the input data, but more on that later.

Now you can see why it is important to take a fresh perspective on this type of networks. Even though they are using the same terms, like neurons, connections and weights their meaning is completely different. Apart from that, you can see that their structure is much simpler than the structure of the other networks that we have covered so far. This is causing the learning process to be different as well. So, let's see how these networks learn.

Learning Process

As we mentioned previously, self-organizing maps use unsupervised learning. This type of learning is also called competitive learning, and we will see in a second why. The first step in the learning process of self-organizing maps is the initialization of all weights on connections. After that, a random sample from the dataset is used as an input to the network. The network then calculates weights of which neuron are most like the input data (input vector). For this purpose, this formula is used:

$$Distance^2 = \sum_{i=0}^{n} (input_i - weight_i)^2$$

where *n* is the number of connection (weights). The map neuron with the best result is called Best Matching Unit or BMU. In an essence, this means that the input vector can be represented with this mapping neuron. Now, the self-organizing maps are not just calculating this point during the learning process, but they also try to make it "closer" to the received input data.

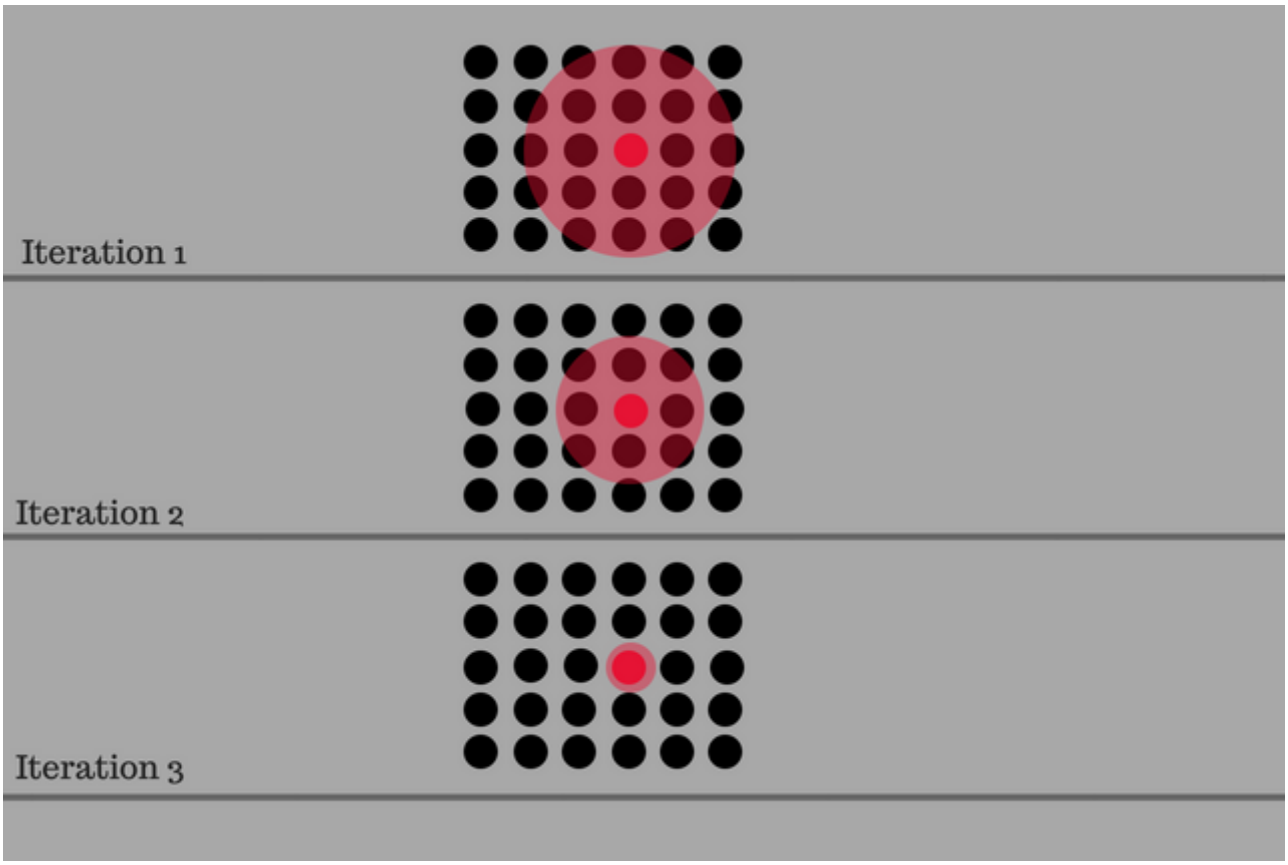
This means that weights on this connection are updated in a manner that calculated distance is even smaller. Still, that is not the only thing that it is done. The weights of neighbors of BMU are also modified so they are closer to this input vector too. This is how the whole map is 'pulled' toward this point. For this purpose, we have to know the *radius* of the neighbors that will be updated. This radius is initially large, but it is reduced in every iteration (epoch). So, the next step in training self-organizing maps is actually calculating mentioned radius value. Following formula is applied:

$$\sigma(t) = \sigma_0 e^{-\frac{t}{\lambda}}$$

where *t* is the current iteration, *σo* is the radius of the map. The *λ* in the formula is defined like this:

$$\lambda = k/\sigma_0$$

where *k* is the number of iterations. This formula utilizes exponential decay, making radius smaller as the training goes on, which was the initial goal. In a nutshell, this means that every iteration through the data will bring relevant points closer to the input data. The self-organizing map is fine-tuned in this way.



When the radius of the current iteration is calculated weights of all neurons within the radius are updated. The closer the neuron is to the BMU the more its weights are changed. This is achieved by using this formula:

$$weight(t+1) = weight(t) + \Theta(t)L(t)(input(t) - weight(t))$$

This is the main learning formula, and it has a few important points that should be discussed. The first one is *L(t)* which represents the learning rate. Similarly to the radius formula, it is utilizing exponential decay and it is getting smaller in every iteration:

$$L(t) = L_0 e^{-\frac{t}{\lambda}}$$

Apart from that, we mentioned that the weight of the neuron will be more modified if that neuron is closer to the BMU. In the formula, that is handled with the *Θ(t).* This value is calculated like this:

$$\Theta(t) = e^{-distBMU/2\sigma(t)^2}$$

Apparently, if the neuron is closer to the BMU, *distBMU* is smaller, and with that *Θ(t)* value is closer to 1. This means that the value of the weight of such neuron will be more changed. This whole procedure is repeated several times.

To sum it up, these are the most important steps in the self-organizing map learning process:

1. Weight initialization
2. The input vector is selected from the dataset and used as an input for the network
3. BMU is calculated
4. The radius of neighbors that will be updated is calculated
5. Each weight of the neurons within the radius are adjusted to make them more like the input vector
6. Steps from 2 to 5 are repeated for each input vector of the dataset

Of course, there are a lot of variations of the equations presented used in the learning process of self-organizing maps. In fact, a lot of research has been done trying to get to the optimal values for the number of iterations, the learning rate, and the neighborhood radius. The inventor, Teuvo Kohonen, suggested that this learning process should be split into two phases. During the first phase, the learning rate would be reduced from 0.9 to 0.1 and the neighborhood radius from half the diameter of the lattice to the immediately surrounding nodes.

In the second phase, the learning rate would be further reduced from 0.1 to 0.0. However, there would be double or more iterations in the second phase and the neighborhood radius value should remain fixed at 1, meaning the BMU only. This means that the first phase would be used for learning and the second phase would be used for fine-tuning.

Conclusion

Self-organizing maps are one very fun concept and very different from the rest of the neural network world. They use the unsupervised learning to create a map or a mask for the input data. They provide an elegant solution for large or difficult to interpret data sets. Because of this high adaptivity, they found application in many fields and are in general mostly used for classification. Initially, Kohonen used them for speech recognition, but today they are also used in Bibliographic classification, Image browsing systems and Image classification, Medical Diagnosis, Data compression and so on.

Thanks for reading!

This article is a part of Artificial Neural Networks Series, which you can check out **here**.

Read more posts from the author at **Rubik's Code**.
