Kohonen's Learning Rule and Adaptive Resonance Theory

Sayeda Suaiba Anwar
Lecturer, Department of CSE
East Delta University

Kohonen’s Self Organizing Systems

Self Organization refers to the ability of some networks to learn without being given the corresponding output for an input pattern.

Self Organizing Networks modify their connection strength based only on the characteristics of the input patterns.

Kohonen Feature Map:

- Simplest self organizing system
- consists of a single layer of neurons (called the Kohonen layer) which are highly interconnected within the Kohonen layer as well as to the outside world through an input buffer layer that is fully connected to the neurons in the Kohonen layer through adjustable weights.

Lateral Inhibition

Kohonen networks utilize lateral inhibition (connections between neurons within a layer) to provide -

- Positive or excitatory connections to neurons in the immediate vicinity
- Negative or inhibitory connections to neurons that are further away.

The strengths of the connections vary inversely with distance between the neurons.

The inputs from lower levels and outputs from higher levels are the same as for other networks.

Generally, there is no feedback from higher to lower layers.

Lateral Inhibition

Lateral connections moderate competition between neurons in the Kohonen layer. When an input pattern is presented to the Kohonen layer, each neuron receives a complete copy of the input pattern modified by the connecting weights, and the varying responses establish a competition that flows over the intralayer connections. The purpose of the competition is to determine which neuron has the strongest response to the input. Each neuron in the layer tries to enhance its output and the output of its immediate neighbors and inhibit the output of the remaining neurons that are further away. Lateral connections can cause oscillations in networks, but the output eventually stabilizes with the output of the neuron, with the strongest response being declared the winner and being transmitted to the next layer if there is one. The activity of all other neurons is squashed, as the network determines for itself which neuron has the greatest response to the input pattern. The relative impact of a neuron's interlayer inhibition is also permitted to decrease with training. Initially, it starts fairly large and is slowly reduced to include only the winner and possibly its immediate neighbors. It has been shown that similar systems exist in the brain with regard to vision.

Lateral Inhibition

The complexity of intralayer connections makes lateral inhibition and excitation hard to implement. An alternative which is much easier to implement is to use a "max" function to determine the neuron with the greatest response to the input and then assign this neuron a +1 value to the output while assigning a zero to all other neurons in that layer. The winning neuron represents the category to which the input pattern belongs. This is not a true implementation of lateral inhibition, but it generally gives the same result as a true implementation, and it is far more efficient when implemented using serial computers. An even simpler alternative is to merely compute the dot product of each of the weight vectors with the input and then select the winner from this list.

In training, the Kohonen network classifies the input vector components into groups that are similar. This is accomplished by adjusting the Kohonen layer weights, so that similar inputs activate the same Kohonen neurons. Preprocessing the input vectors is very helpful. This involves normalizing all inputs before applying them to the network—that is, divide each component of the input vector by the vector's length:

$$x_i' = \frac{x_i}{\left[x_1^2 + x_2^2 + x_3^2 + \cdots + x_N^2\right]^{1/2}} \qquad (9.4\text{-}1)$$

Lateral Inhibition

When building a Kohonen layer, two new things are required:

1. Weight vectors must be properly initialized. Generally, this means the weight vectors point in random directions.
2. Weight vectors and input vectors must be normalized to a constant fixed length, usually “unity”.

Kohonen Learning Rule

Determining the winner is the key to training a Kohonen network. Only the winner and its immediate neighbors modify the weights on their connections. The remaining neurons experience no training. The training law used is

$$\Delta w_i = \eta\left[x_i - w_i^{\text{old}}\right] \tag{9.4-2}$$

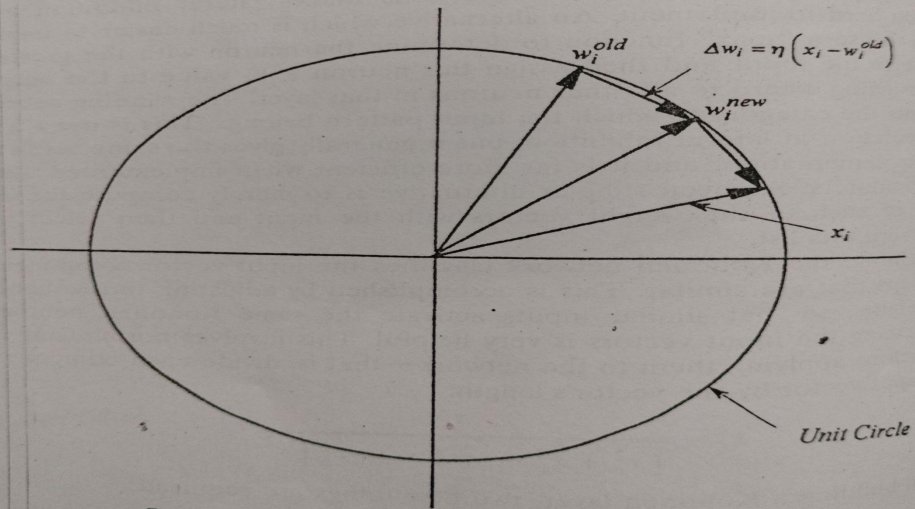

Figure 9.6 Learning in a Kohonen neural network.

Kohonen Learning Rule

where η is the learning constant whose value may vary between 0 and 1, with a typical value of about 0.2, and x_i is the input along the ith connection. It can be shown that this learning rule is a variation of the Woodrow–Hoff learning rule. This is shown graphically in Figure 9.6 for the two-dimensional case. Learning is illustrated for the case of an input x_i and a weight w_i^{old}. The difference between these two unit vectors is a vector from the tip of w_i^{old} to the tip of x_i. In Figure 9.6, this vector is broken into two parts (cords of the unit circle) so that the w_i^{new} will have a unit length. The vector from the tip of w_i^{old} to w_i^{new} represents the change in the weight vector due to learning and is equal to $\eta(x_i - w_i^{\text{old}})$.

Kohonen Learning Rule

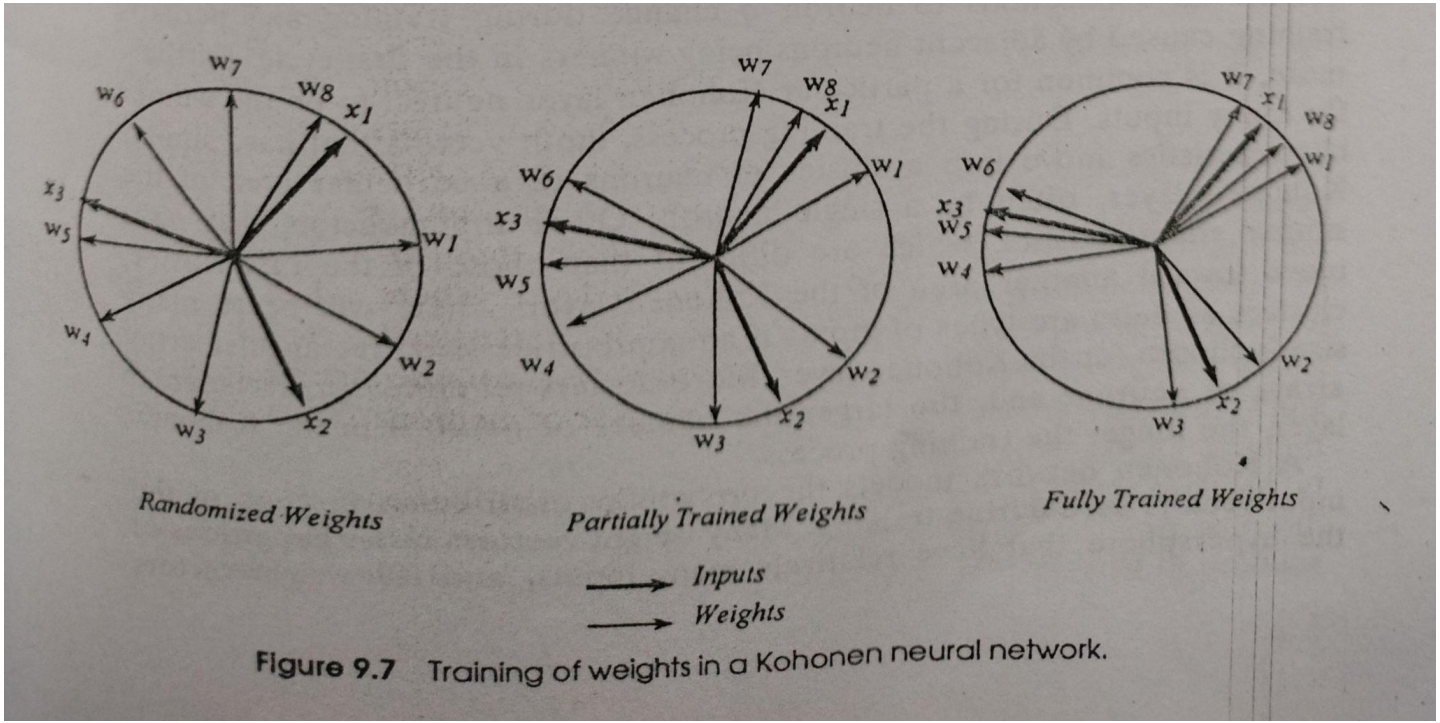

Figure 9.7 Training of weights in a Kohonen neural network.

Training of a Kohonen Neural Network

Let us consider a Kohonen neural network [sometimes called a self-organizing map (SOM)] with an input buffer layer (typically a linear array) and a Kohonen layer (typically a rectangular array or grid) that are fully connected. An input vector is applied to the buffer layer, and its component vectors are transmitted to each neuron in the Kohonen layer through randomized connecting weights. The neuron in the Kohonen layer with the strongest response (let's call it neuron q) is declared the winner and its value is set equal to 1. Then the weights connecting all component vectors from the buffer layer to the winning neuron undergo training in accordance with the process shown graphically in Figure 9.6.. Neurons immediately adjacent to the winner are also allowed to undergo training. Then a second input vector is applied to the buffer layer, another neuron in the Kohonen layer is declared the winner, its value is set equal to unity, and it and its neighbors are allowed to undergo training. This process continues until all the inputs in the epoch of data have been applied to the buffer input layer. In the training process, the weights tend to cluster around the input vectors as indicated in Figure 9.7. Training stops when a criterion relating the nearness of the weights in the clusters to the relevant input vector is satisfied. Kohonen neural networks train relatively rapidly compared to backpropagation neural networks. Often a single cycle through an epoch of data, especially if the data set is large, constitutes adequate training.

Training of a Kohonen Neural Network

It is important to note that just because the input vector selected neuron q as the most active in the first cycle of training does not mean that it will select neuron q in the second or subsequent cycles of training, because the weights on connections to neuron q change during training and perhaps training caused by adjacent neurons being winners in the first cycle. Furthermore, it is common for a particular Kohonen layer neuron to be the winner for many inputs. During the training process, input vectors that have similar characteristics move into a cluster of neurons in a particular area of the Kohonen layer, often to a single neuron. Other input vectors that have similar characteristics, which are different than those of the first cluster, move toward another area of the Kohonen layer. There will be as many clusters as there are types of inputs if an appropriate-sized rectangular array size is chosen for the Kohonen layer. More clusters require larger rectangular arrays of neurons and, the larger the number of neurons in the Kohonen layer, the longer the training process.

A Kohonen network models

Training of a Kohonen Neural Network

- Kohonen’s network works best when the network are very large. The smaller the network, the less accurate the statistical model will be.
- Kohonen’s networks are very fast, even while training. Thus, they have the potential for real-time application learning.
- These networks can learn continuously. Hence, if the statistical distribution of the input data changes over time, it can automatically adapt to those changes.
- Learning rate coefficient is always less than 1; usually starts at about 0.7 and gradually reduced during training.
- Weight vector must be set before training begins. It is common practice to randomize these weights to small values.

Adaptive Resonance Theory

- Adaptive Resonance Neural Networks are based on adaptive resonance theory (ART).
- Three general types of ART networks are used:
  - ART 1: which can handle only binary inputs
  - ART 2: which can handle gray-scale inputs
  - ART 3: can handle analog inputs better, is more complex and developed to overcome the limitations of ART 2.

General Operations of ART Neural Network

ART neural networks are two-layer neural networks fully connected with inputs going to the bottom layer, from which they are transmitted through adjustable weights to the top or storage layer. This is a bottom-up or "trial" pattern that is presented to stored patterns of the upper storage layer. The input pattern is modified during its transmission through the "bottom-up" weights to the upper layer, where it tries to stimulate a response pattern in the storage layer that contains several possible responses. Training takes place after every pass of the pattern, up or down. Since the training rule does not matter, it is common to use Hebbian learning for convenience. If this "bottom-up" pattern is selected, then "resonance" occurs, and the input is put into the matching pattern category. If it is not selected, the resulting activity in the "top-down" layer (called the "expectation" pattern or "first guess" pattern) is usually different from the bottom-up pattern because the top-down pattern is presented through the top-down weights to the bottom layer. Then the weights are adjusted, and the process is repeated. After a number of trials, the process is stopped, and a new category of pattern is created in the storage layer. This ability of ART to create new categories is its most important characteristic.

General Operations of ART Neural Network

When a pattern fails to produce a match, a new pattern of nodes (from the storage layer) is now free to attempt to reach resonance with the input layer's pattern. In effect, when the trial patterns do not match, a reset subsystem signals the storage layer that a particular guess was wrong. Then that guess is "turned off," allowing another pattern from storage to take its place. This cycle repeats as many times as necessary. When resonance is reached and the guess is deemed acceptable, the search automatically terminates. This is not the only way a search can terminate; the system can terminate its search by learning the unfamiliar pattern being presented. As each trial of the search occurs, small weight changes occur in the weights of both the bottom-up and top-down pathways. These weight changes mean that the next time the trial pattern is passed up to the storage layer, a slightly different activity pattern is received, providing a mechanism for the storage layer to change its guess. If the system cannot find a match and if the input pattern persists long enough, the weights eventually are modified enough that an uncommitted node in the storage layer learns to respond to the new pattern. These changes in weights also explain why the storage layer's second or third guess may prove to be a better choice than the original one. The small weight changes ensure that the activity generated by the bottom-up pattern in the second pass is somewhat different from the activity generated in the first pass. If the input is a slightly noisy version of a stored pattern, it may require a few weight changes before the truly best guess can be matched.

Properties of ART - 1

1. It learns constantly but learns only significant information and does not have to be told what information is significant.
2. New knowledge does not destroy information already learned.
3. Rapidly recalls an input pattern it has already learned.
4. It functions as an autonomous associative memory.
5. It can handle only binary inputs.
6. Its ability to create new categories is its most important attribute.