The Hopfield Discrete Recurrent Neural Network

(Commonly known as the Hopfield NN)

Given:

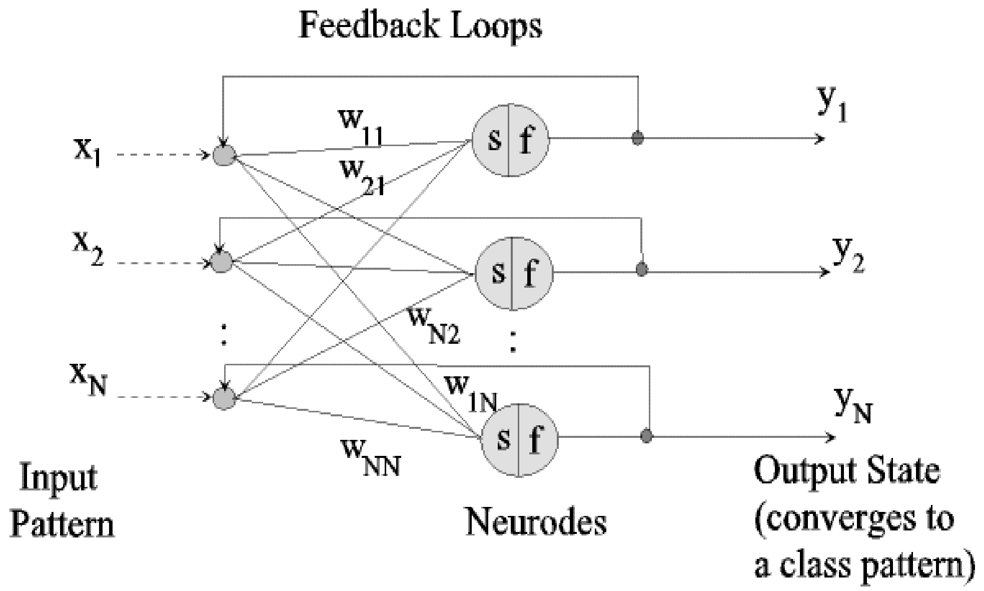
1) a set of *class patterns* (also called *output identifiers*) $\{\mathbf{Y}^{(q)} : q = 1,...,Q\}$ of N dimensions, where $\mathbf{Y}^{(q)} = (Y_1^{(q)},...,Y_N^{(q)})$. Each component value of the class patterns is either *1* or *-1* so that it is a binary codeword.

2) a distorted, partial, or approximate *input pattern* $\mathbf{x} = (x_1,...,x_N)$.

The Problem:

use a feedback neural network to accept the input pattern $\mathbf{x}$ and iteratively feedback the outputs until the output identifier converges to a class pattern $\mathbf{Y}^{(q)} = (Y_1^{(q)},...,Y_N^{(q)})$ for some q. The output must be associated with the input pattern in that the network converges to the correct class pattern.

The Solution (the Hopfield NN):



1. The NxN weight matrix is determined by the Q class patterns $y_n(t+1)$, $q = 1,...,Q$.

$W = [w_{ij}] = (1/N) \sum_{(q=1,Q)} \mathbf{Y}^{(q)T}\mathbf{Y}^{(q)}$, where T denotes the transpose, and put $w_{nn} = 0$ for all n.

This matrix is symmetric, which is sufficient to guarantee that the transient output patterns (states) $\mathbf{y}^{(q)} = (y_1^{(q)},...,y_N^{(q)})$ converge to a class pattern $\mathbf{Y}^{(q)} = (Y_1^{(q)},...,Y_N^{(q)})$. Note that this is a sum of NxN matrices with the factor (1/N) because $\mathbf{Y}^{(q)T}\mathbf{Y}^{(q)}$ yields an NxN matrix, e.g.,

$$(1/3)(1, -1, -1)^T (-1, -1, 1) = (1/3)\begin{bmatrix}1\\-1\\-1\end{bmatrix}[-1\ \ -1\ \ 1] = (1/3)\begin{bmatrix}-1 & -1 & 1\\ 1 & 1 & -1\\ 1 & 1 & -1\end{bmatrix}$$

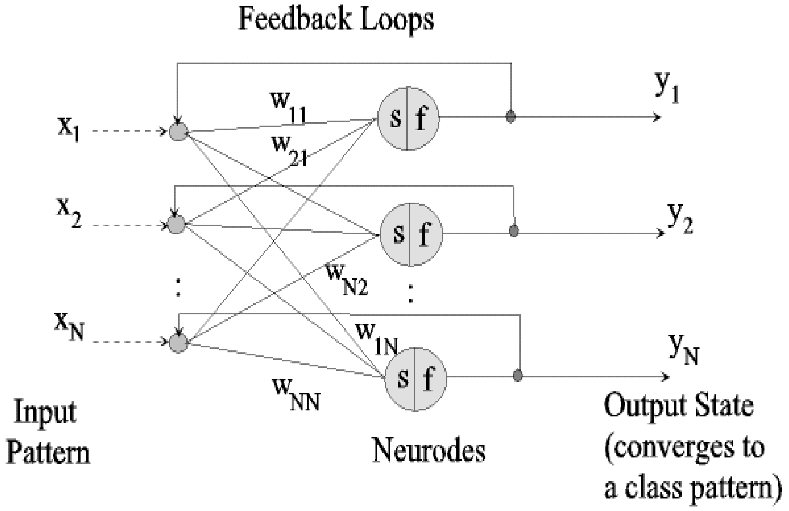
2. $\boldsymbol{x} = (x_1,...,x_N)$ is initially submitted to the HNN on the left above and used as the first feedback $\boldsymbol{y}^{(0)} = \boldsymbol{x}$ at time $t = 0$

3. Iterate:

select n at random from 1 to N with a random number generator

compute sum at time $t = 1$ via
$s_n = w_{n1}y_1(t) + ... + w_{nN}y_n(t)$

$y_j(t+1) = sgn(s_j - \tau_j)$, for all j = 1,...,N



where τ_n is the threshold (taken to be 0 for bipolar values of patterns). Note that $w_{jj} = 0$, so $y_n(t+1)$ is not updated for this selection of n.

4. If no value $y_j(t+1)$ has changed, $j = 1,...,N$, then none will change on the next iteration either (everything remains the same in the computations), so we put

$$Y = (Y_1,...,Y_N) = \boldsymbol{y}(t+1) = (y_1(t+1),...,y_N(t+1))$$

or else put $t \Leftarrow t + 1$ and go to Step 3 above.

When the process stops the output will be one of the class patterns, unless there are not enough neurodes for the number of classes. The limit is about 0.15N for the number of classes, where N is the number of of neurodes. However, for safety, we should use 0.12 as a guide to avoid convergence to spurious patterns.

Why it Works:

An energy principle from physics holds. The *energy* here is a sum analogous to energy in physics, which is

$$E = -(1/2)\sum_{(n<j)} w_{nj}\, s_n\, s_j + \sum_{(n=1,N)}\tau_n s_n$$

This is considered to be a Lyapunov function, so if the update feedback is chosen randomly one at a time, the convergent state is a where the energyis at a local minimum. There are multiple local minima determined by N and the number of patterns.

A local minima is called an *attractor* and the energy function will decrease toward the attractor whose *basin of attraction* the initial pattern is in. The way the weights are chosen, each local minima is determined by a class pattern. The initial energy for an input pattern is

$$E(x) = -\mathbf{x}W\mathbf{x}^T$$

and the feedback iterations moves downhill in the basin of attraction to the local minimum.

References:

J. J. Hopfield, "Neural networks and physical systems with emergent collective comutational abilities," Proc. Nat. Acad. Sciences U.S.A, vol. 79, no. 8, 2554-2558, April, 1982.